

A Survey on Memory Management Algorithms and Their Role in Minimizing Latency in Cloud Computing

Gauri Moghe

Department of Computer Science

California State University-Long Beach

GauriShashank.Moghe01@student.csulb.edu

Debarati Datta

Department of Computer Science

California State University-Long Beach

Rishwa Patel

Department of Computer Science

California State University-Long Beach

RishwaDharmeshkumar.Patel01@student.csulb.edu

Tapan Shah

Department of Computer Science

California State University-Long Beach

tapan.shah01@student.csulb.edu

Reema Pandya

Department of Computer Science

California State University-Long Beach

ReemabenGirishkumar.Pandya01@student.csulb.edu

Parikshith Sharma

Department of Computer Science

California State University-Long Beach

Parikshith.Sharma01@student.csulb.edu

Abstract—The integration of cloud in web applications is increasing day by day. Memory management is crucial in cloud computing to ensure efficient memory allocation and utilization of shared memory across virtual machines to enhance performance, scalability, and reduce latency. This survey paper examines memory management techniques that are useful to reduce latency specifically in cloud-based applications. We review a range of techniques, including Memory & Resource Allocation, Memory Partitioning and Scheduling, Page Management and several others highlighting their potential to reduce the memory latency. Through a detailed comparative analysis, we highlight algorithms such as Virtual Page Behavior Based Memory Management (VBPM), Task and Data Co-Scheduling (TDCS) and Spring Buddy which outperform with respect to latency reduction. These algorithms demonstrate the significance of adaptive and predictive techniques in balancing latency, scalability, and energy efficiency. Their performance emphasizes the value of strategic memory management in optimizing cloud workloads and ensuring high system efficiency.

Index Terms—Memory Latency, Cloud Computing, Memory Management, Energy Efficiency, Memory Allocation, Memory Partitioning, Resource Allocation, Task Scheduling, Page Management

I. INTRODUCTION

Cloud computing can be seen as the provision of computer system resources, particularly storage and processing power, by a third party without the end user having to oversee and control the operation [6]. It works on the principle of resource pooling in order to achieve efficiency. It has become crucial in the modern day for data processing along with the delivery of applications since it has made on demand and scalable resources possible. But of all the issues, latency is perhaps most aggravating—especially for real-time applications like IoT, AI, and financial analytics where each millisecond counts. Memory latency refers to the time delay between when a request for data is made and when those data are retrieved from memory [2]. Memory management is the key to solving this issue because if memory is poorly allocated, it

results into delays and suboptimal usage of resources. This survey aims to comprehensively analyze existing memory management algorithms, classify and assess them in regard to their effectiveness in minimizing the latency time. This survey provides insights into current methodologies while identifying areas for innovation.

II. METHODOLOGY

For the survey, several research papers and articles discussing memory management were gathered. Based on the techniques identified in these sources, additional articles and papers were reviewed to examine how those techniques were applied to develop new algorithms or mechanisms. Selected techniques included Memory Allocation, Resource Allocation, Dynamic Pooling, Page Management, Memory Partitioning, Task Scheduling, and Allocation. Key algorithms were identified based on their effectiveness in enhancing performance, reducing latency, and improving scalability in cloud environments. The performance of the analyzed algorithms was evaluated using key performance indicators such as latency, memory usage, scalability, and energy efficiency across diverse cloud environments. The algorithms were categorized and grouped based on the respective techniques they employed to systematically highlight their individual strengths, limitations, and trade-offs in addressing memory management challenges.

III. MEMORY MANAGEMENT TECHNIQUES

Memory management techniques are approaches used to allocate, organize, and optimize the utilization of memory resources in computing systems. These techniques play a pivotal role in ensuring efficient memory usage, reducing waste, and maintaining overall system performance by addressing challenges such as fragmentation, latency, and contention. They are particularly important in balancing resource availability and computational demands in environments like operating systems, cloud computing, and high-performance computing,

where dynamic workloads and scalability are critical factors [9] [10] [11].

A. Page Management

Page management techniques, such as demand paging and prediction-based allocation, optimize memory usage by loading necessary pages and discharging unused ones. These methods improve access speeds and reduce memory footprint, though inaccurate predictions can lead to performance issues during high demand [5] [15]. Table I compares key algorithms, such as CLOCK-DWF, which minimizes delays for predictable access patterns, and LRU-WPAM, which balances latency and energy efficiency. Advanced techniques like APP-LRU and VBPM further optimize memory by preemptively assigning pages to faster memory regions, reducing migration costs while maintaining low latency.

TABLE I
PAGE MANAGEMENT

Algorithm	Key Features	Impact
CLOCK-DWF [5]	Allocates frequently accessed read/write pages closer to processors to minimize access delays.	Reduces latency for predictable access patterns but relies heavily on DRAM.
LRU-WPAM [5]	Dynamically adjusts page allocation between PCM and DRAM based on access patterns.	Balances latency and energy efficiency but incurs overhead from frequent migrations.
MHR-LRU [5]	Allocates write-heavy pages to DRAM to prevent delays caused by PCM write operations.	Reduces PCM write delays but may underutilize PCM for read-heavy operations.
APP-LRU [5]	Predicts access patterns using metadata and preemptively assigns pages to faster memory regions.	Balances latency and energy efficiency but retains migration overhead.
VBPM [5]	Preemptively maps pages to DRAM/PCM based on simulated access behavior, avoiding migrations.	Achieves the lowest latency and eliminates migration costs but lacks runtime adaptability.

B. NUMA-Aware Memory Management

NUMA-Aware Memory Management aligns processes with local memory banks to guarantee the best possible memory placement in Non-Uniform Memory Access systems. Data transport speeds are increased and cross-node memory access overheads are decreased with this method. In large-scale and HPC systems, where memory locality is essential to sustaining high throughput, it has a particularly significant effect [15].

C. Machine Learning-Driven Memory Management

Predictive models are used by machine learning-driven memory management to anticipate memory usage trends and maximize allocation. Because they avoid memory bottlenecks and maximize resource efficiency, these strategies are very successful in workloads that are dynamic and unpredictable. This method is ideal for contemporary, adaptive systems since machine learning models can also adjust to changing workload patterns [13].

D. Task Scheduling and Allocation

Task Scheduling and Allocation uses memory task offloading and latency-aware scheduling to ensure jobs are distributed across memory and computation resources as efficiently as possible. These techniques are effective at reducing memory resource contention, optimizing utilization, and balancing workloads. Table II summarizes key algorithms used in task scheduling and allocation, highlighting their features and impacts. For example, the Greedy Algorithm minimizes task waiting time by assigning tasks to the earliest available core, while the CP Algorithm prioritizes tasks on the critical path to reduce execution time. Techniques like HEFT and BLTS excel in handling heterogeneous or dynamic workloads by balancing latency and scalability but may incur additional overhead. Optimization-based methods like PSO and ACO dynamically adjust task allocations to reduce contention and maintain energy efficiency, though they require significant computational resources [19] [12] [7].

TABLE II
TASK SCHEDULING AND ALLOCATION

Algorithm	Key Features	Impact
Greedy Algorithm [12]	Assigns tasks to the earliest available core, minimizing task waiting time.	Reduces latency and energy consumption but may lead to uneven task distribution.
CP Algorithm [12]	Prioritizes tasks on the critical path to minimize execution time.	Minimizes latency for critical tasks but may neglect non-critical tasks.
BL Algorithm [12]	Uses task b-levels (longest path to exit task) for scheduling priority.	Reduces latency for high-priority tasks but may under-utilize resources for lower-priority tasks.
TDCS [12]	Simultaneously optimizes task and data scheduling to improve efficiency.	Balances latency and energy use but introduces high computational complexity.
BLTS [12]	Dynamically redistributes tasks by task-stealing to fill idle cores and balance workloads.	Reduces latency and improves scalability but increases overhead from dynamic balancing.
HEFT [7]	Ranks tasks based on average processing and communication times for heterogeneous environments.	Reduces latency in heterogeneous systems but may have scalability challenges for larger systems.
Min-Min [7] & Max-Min	Allocates tasks based on minimum or maximum expected execution times to balance workloads.	Improves latency for either small or large tasks but may cause load imbalance in dynamic scenarios.
Round Robin (RR) and Weighted RR [7]	Cyclically assigns tasks to cores, ensuring fair and regular task execution.	Reduces latency and ensures fair task distribution but may not prioritize critical tasks.
PSO & ACO [7]	Uses optimization models to distribute tasks dynamically, reducing delays in resource contention.	Reduces latency while maintaining energy efficiency but computationally expensive.

E. Memory Deduplication

Memory Deduplication consolidates duplicate memory content across applications or virtual machines to free up memory

and reduce overhead. This technique optimizes memory utilization by eliminating redundant data, making it especially useful in virtualized cloud environments. Table III highlights the Group-based Deduplication algorithm, which removes redundant memory pages across VMs to improve latency and energy efficiency. However, this approach requires significant processing resources, which can introduce computational overhead and impact scalability [8] [4].

TABLE III
MEMORY DEDUPLICATION

Algorithm	Key Features	Impact
Group-based deduplication [8]	Eliminates redundant memory pages across VMs to optimize memory usage.	Improves latency and energy efficiency but requires significant processing resources.

F. Dynamic Pooling

Leveraging resource disaggregation or pooling techniques, dynamic pooling modifies memory resources in real time to satisfy workload demands. This method guarantees resource elasticity and works incredibly well with varying workloads, particularly in remote and large-scale cloud systems. Dynamic pooling increases system efficiency by reducing memory underutilization and avoiding allocation delays during demand spikes [9].

G. Memory Partitioning and Scheduling

Memory Partitioning and Scheduling reduce congestion and improve data locality by dividing memory and scheduling accesses. Table IV highlights key algorithms, including Multi-Dimensional Rotation Scheduling for reducing contention and First-Level Partition Scheduling for optimizing latency and scalability [18] [1].

TABLE IV
MEMORY PARTITIONING AND SCHEDULING

Algorithm	Key Features	Impact
Multi-Dimensional Rotation Scheduling [1]	Compacts execution of dependent tasks within a partition to reduce contention.	Reduces latency but limited by dependency constraints.
Prefetch and Keep Scheduling [1]	Prefetches and retains data in memory for availability during execution.	Improves latency and scalability but incurs prefetch overhead for dynamic workloads.
Partition Optimization [1]	Optimizes partition sizes to improve execution efficiency and data locality.	Balances latency and energy efficiency but is complex for irregular iteration patterns.
First-Level Partition Scheduling [18]	Divides iteration space into smaller partitions for better data locality.	Reduces latency in local memory but requires precise dependency analysis.
Second-Level Partition Scheduling [18]	Groups smaller partitions to optimize data movement across memory levels.	Balances latency and scalability but can introduce complexity in dynamic workloads.

H. Heap Management

Heap management focuses on efficiently allocating and deallocating memory in the heap while minimizing fragmentation. Techniques like the adaptive buddy system ensure prompt memory block allocation, reducing contention and sustaining steady performance, especially in concurrent workloads [10]. Table V highlights the Hermes Algorithm, which improves scalability and reduces latency by 54.4% through proactive memory reclamation, although it requires active monitoring [14].

TABLE V
HEAP MANAGEMENT

Algorithm	Key Features	Impact
Hermes Algorithm [14]	Gradually reserves memory and uses proactive reclamation to reduce allocation delays.	Reduces latency by 54.4% and improves scalability but requires active memory monitoring.

I. Resource Allocation

Resource Allocation dynamically allocates memory resources to workloads using techniques like hierarchical and predictive memory allocation. These methods anticipate and respond to memory demands, ensuring system stability under fluctuating workloads. They also reduce memory waste, improve energy efficiency, and balance operating costs with performance [9] [13]. Table VI highlights key algorithms, such as Spring Buddy, which dynamically splits and merges memory blocks to reduce allocation delays, and AIMING, which minimizes inter-region delays in federated cloud systems. Another approach, Dynamic Resource Allocation Using Skewness, balances server resource usage to reduce latency and improve energy efficiency, though it requires continuous monitoring [10] [19] [13].

TABLE VI
RESOURCE ALLOCATION

Algorithm	Key Features	Impact
Spring Buddy [10]	Dynamically splits and merges memory blocks to reduce allocation and deallocation delays.	Reduces latency by 71.47% but may temporarily increase fragmentation in dynamic scenarios.
AIMING [19]	Allocates resources regionally in federated-cloud systems to minimize delays in inter-region communication.	Reduces latency and improves scalability but is limited to federated environments.
Dynamic Resource Allocation Using Skewness [13]	Balances resource usage across servers to reduce delays caused by overloaded nodes.	Improves latency and energy efficiency but requires continuous monitoring and adjustments.

J. Caching Techniques

Near-data processing, which stores frequently accessed data closer to the processor, and multi-level caching are examples

of caching techniques. In data-intensive settings like edge computing and big data analytics, these methods are excellent at drastically cutting memory access times and enhancing system performance. Furthermore, in memory-intensive applications, energy-efficient caching techniques aid in lowering power consumption [11] [3].

IV. RESULTS & ANALYSIS

The evaluation of memory management algorithms in cloud computing environments relies on performance metrics that assess algorithm efficiency, scalability, and resource utilization. These metrics are essential for understanding algorithm performance in real-world scenarios, particularly for latency-sensitive workloads. Key metrics used in this analysis include memory access time, allocation latency, resource utilization, energy consumption, and the effectiveness of techniques like partitioning, deduplication, and resource pooling. Each algorithm was evaluated against these criteria to identify its strengths, weaknesses, and suitability for specific tasks.

This survey was guided by several critical metrics to ensure a comprehensive and fair comparison. Memory access time was one of the primary metrics, as it directly measures the speed at which data is retrieved from memory, a crucial factor for page management algorithms such as VBPM, which reduced access time by 24%. Allocation latency and deallocation latency were also significant metrics, particularly for dynamic resource allocation algorithms. These metrics represent the time taken to assign or reclaim memory resources, with Spring Buddy demonstrating outstanding performance by reducing allocation latency by 71.47% and deallocation latency by 93.20%. [9].

Another key metric was resource utilization, which evaluates how effectively memory resources are used. Group-Based Memory Deduplication excelled in this category by identifying and eliminating redundant memory pages across virtual machines, improving memory availability and reducing contention. [16].

The energy consumption metric was especially relevant for algorithms like the JCO Algorithm, which balances the need for efficient memory management with the goal of minimizing energy usage [5].

Task scheduling and allocation rely on minimizing execution delays and optimizing resource alignment. The Task-Data Co-Scheduling (TDCS) algorithm excelled in this category by ensuring that tasks and their associated data were scheduled together. This alignment reduced resource mismatch delays and improved workload efficiency, making TDCS superior to traditional memory-prioritized mapping techniques [17].

Memory deduplication focuses on optimizing resource utilization by identifying and eliminating redundancy across virtual machines. The Group-Based Memory Deduplication algorithm showed substantial improvements in memory utilization, reducing resource contention and ensuring availability for critical tasks. Its multi-tenant focus and efficiency set it apart from other deduplication methods [16].

TABLE VII
COMPARATIVE ANALYSIS OF MEMORY MANAGEMENT ALGORITHMS IN CLOUD COMPUTING

Technique	Best Algorithm	Key Results	Comparative Insights
Page Management	VBPM	Reduced memory access time by 24%, eliminated migration overhead [9].	Achieves significant latency reduction by avoiding runtime migrations, outperforming Prefetch and Keep Scheduling in hybrid memory systems.
Task Scheduling and Allocation	TDCS	Improved resource allocation and efficiency for latency-sensitive tasks [17].	Minimizes execution latency by aligning tasks with associated data, making it superior to traditional memory-prioritized mapping.
Memory Deduplication	Group-Based Deduplication	Enhanced memory utilization and reduced resource contention [5].	Reduces memory contention and task latency by optimizing resource availability in multi-tenant environments.
Memory Partitioning and Scheduling	Multi-Dimensional Rotation Scheduling	Compacts execution of dependent tasks within a partition to reduce contention [4].	Effectively reduces latency caused by dependency constraints, outperforming other partitioning techniques.
Heap Management	Hermes Algorithm	Proactively reclaimed unused memory, ensuring availability for critical tasks [8].	Improves latency by prioritizing memory availability for latency-sensitive workloads, outperforming standard heap management systems.
Resource Allocation	Spring Buddy	Reduced memory allocation latency by 71.47% and deallocation latency by 93.20% [10].	Demonstrates exceptional latency reduction by dynamically adjusting memory regions to meet workload demands.

Memory partitioning and scheduling prioritize reducing contention and dependency-related delays. The Multi-Dimensional Rotation Scheduling algorithm emerged as the best in this category, effectively compacting dependent task execution within partitions to minimize contention. While dependency constraints remain a limitation, its latency reduction capabilities make it superior to other partitioning techniques [4].

Heap management is evaluated on its ability to proactively reclaim unused memory and ensure availability for critical tasks. The Hermes Algorithm excelled in this domain by

prioritizing latency-critical tasks, outperforming standard heap management systems [8].

Resource allocation evaluates latency in assigning and reclaiming memory resources. Proactive memory management was highlighted by the Hermes Algorithm, which prioritized latency-critical workloads, and the Spring Buddy algorithm, which dynamically adapted to fluctuating workloads by splitting and merging memory regions. Each algorithm demonstrated specific strengths, offering insights into optimizing memory management for diverse cloud computing challenges. The Spring Buddy algorithm demonstrated exceptional adaptability by dynamically splitting and merging memory regions to match workload demands. It achieved a 71.47% reduction in allocation latency and a 93.20% reduction in deallocation latency, outperforming alternatives such as the Hermes Algorithm in environments with fluctuating workloads [8].

V. CONCLUSION

This survey provides a comprehensive analysis of memory management techniques and their effectiveness in minimizing latency and optimizing resource utilization in cloud environments. Latency reduction has emerged as a critical priority in cloud systems, driven by the increasing demand for real-time processing and high-performance applications across industries such as finance, healthcare, e-commerce, and entertainment. Cloud service providers face the challenge of balancing performance with resource efficiency while catering to diverse workloads, making effective memory management a cornerstone of modern cloud architecture. In this context, this survey fills a vital gap by systematically categorizing memory management algorithms based on core techniques such as Memory Allocation, Resource Allocation, Page Management, and Task Scheduling.

The necessity for such an analysis is emphasized by the increasing complexity of cloud workloads and the growing importance of achieving predictable performance in dynamic, multi-tenant environments. This study evaluates these algorithms against key performance metrics, including latency, memory usage, scalability, and energy efficiency, to provide a comprehensive understanding of their effectiveness. High-performing algorithms such as VBPM [5], which achieved a 24% reduction in memory access time, and Spring Buddy, which reduced allocation latency by 71.47%, stand out as practical solutions for diverse cloud workloads [10]. These findings highlight the importance of adopting well-designed memory management strategies to ensure seamless operation and optimal performance of cloud applications.

VI. FUTURE WORK

Future work in memory management for cloud computing systems should focus on developing hybrid techniques that integrate the strengths of behavior-based, predictive, and proactive approaches. These hybrid methods can dynamically adapt to workload patterns, ensuring efficient memory utilization and reducing latency under diverse operating conditions. Leveraging artificial intelligence (AI) and machine learning

(ML) for predictive memory allocation and real-time workload management also holds great promise. AI-driven models can analyze historical and live data to optimize resource allocation, providing enhanced responsiveness without significant computational overhead. Additionally, optimizing the trade-offs between energy efficiency and latency is a critical avenue for exploration, where multi-objective optimization techniques can ensure sustainable and low-latency operations in multi-tenant systems [19].

Scalability and resource pooling must also be enhanced to meet the growing demands of multi-tenant cloud environments. This includes designing algorithms capable of dynamically adjusting resources across distributed systems while maintaining performance and reliability. Security and standardization represent another essential focus area, where the integration of robust security measures into memory management algorithms can address data integrity and privacy concerns. Establishing standardized benchmarks and metrics will further enable consistent evaluation and comparison of proposed solutions, driving innovation, and fostering transparency. By following these research directions, future studies can significantly advance memory management practices, ensuring more efficient, secure, and sustainable cloud computing systems [12] [4].

REFERENCES

- [1] F. Chen and E. H.-M. Sha. Loop scheduling and partitions for hiding memory latencies. In *Proceedings 12th International Symposium on System Synthesis*, pages 64–70. IEEE, 1999.
- [2] M. Compute. How does memory latency impact overall system performance?, 2024. Accessed: 2024-12-09.
- [3] P. Das. Cache memory architectures for handling big data applications: A survey. In *Smart Computing Paradigms: New Progresses and Challenges: Proceedings of ICACNI 2018, Volume 2*, pages 211–220. Springer, 2020.
- [4] P. P. Deshmukh and S. Y. Amdani. Virtual memory optimization techniques in cloud computing. In *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*, RICE '18, pages 1–4, New York, NY, USA, 2018. IEEE.
- [5] J. Huang, X. Zhang, G. Han, G. Jia, H. Liyou, and J. Wan. Virtual page behavior based page management policy for hybrid main memory in cloud computing. In *2016 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 120–124. IEEE, 2016.
- [6] IBM. What is cloud computing?, 2024. Accessed: 2024-12-09.
- [7] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh. A comprehensive survey for scheduling techniques in cloud computing. *Journal of Network and Computer Applications*, 143:1–33, 2019.
- [8] P. Lakkadwala and P. Kanungo. Memory utilization techniques for cloud resource management in cloud computing environment: A survey. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pages 1–5. IEEE, 2018.
- [9] H. Li, D. S. Berger, S. Novakovic, L. Hsu, D. Ernst, P. Zardoshti, M. Shah, I. Agarwal, M. Hill, M. Fontoura, et al. First-generation memory disaggregation for cloud platforms. *arXiv preprint arXiv:2203.00241*, 2022.
- [10] Y. Lu, W. Liu, C. Wu, J. Wang, X. Gao, J. Li, and M. Guo. Spring buddy: A self-adaptive elastic memory management scheme for efficient concurrent allocation/deallocation in cloud computing systems. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 402–409. IEEE, 2021.
- [11] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani. Energy-efficient caching for mobile edge computing in 5g networks. *Applied Sciences*, 7(6):557, 2017.

- [12] T. Maqsood, N. Tziritas, T. Loukopoulos, S. A. Madani, S. U. Khan, and C.-Z. Xu. Leveraging on deep memory hierarchies to minimize energy consumption and data access latency on single-chip cloud computers. *IEEE Transactions on Sustainable Computing*, 2(2):154–166, 2017.
- [13] D. R. Patil and M. Sharma. Dynamic resource allocation and memory management using machine learning for cloud environments. *International Journal*, 9(4), 2020.
- [14] A. Pi, J. Zhao, S. Wang, and X. Zhou. Memory at your service: Fast memory allocation for latency-critical services. In *Proceedings of the 22nd International Middleware Conference*, pages 185–197. ACM, 2021.
- [15] A. Pupykina and G. Agosta. Survey of memory management techniques for hpc and cloud computing. *IEEE Access*, 7:167351–167373, 2019.
- [16] G. E. Shaikh and U. Shrawankar. Dynamic memory allocation technique for virtual machines. In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–6. IEEE, 2015.
- [17] X. Wang, C. Xu, K. Wang, F. Yan, and D. Zhao. Toward cost-effective memory scaling in clouds: Symbiosis of virtual and physical memory. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 33–40. IEEE, 2018.
- [18] Z. Wang, T. W. O’Neil, and E.-M. Sha. Optimal loop scheduling for hiding memory latency based on two-level partitioning and prefetching. *IEEE Transactions on Signal Processing*, 49(11):2853–2864, 2001.
- [19] J. Wei, A. Zhou, J. Yuan, and F. Yang. Aiming: Resource allocation with latency awareness for federated-cloud applications. *Wireless Communications and Mobile Computing*, 2018(1):4593208, 2018.