International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Evolving Operating System Strategies for Intelligent Scheduling and Adaptive Memory Management

Vishal Sharma[1], Varuna Gupta[2]

[1]Student, Department of Computer Science, Christ University, Bengaluru, India
Corresponding Author Email: *vishal.sharma[at]mca.christuniversity.in*

[2]Ph.D., Associate Professor, Department of Computer Science, Christ University, Bengaluru, India
Contributing Author Email: *varuna.gupta[at]christuniversity.in*

Abstract: ***Operating system design has moved beyond fixed scheduling rules and uniform memory policies as modern hardware and workloads have grown more diverse and demanding. Contemporary systems increasingly rely on learning driven schedulers that adjust decisions in real time, allowing CPU resources to be balanced more effectively under fluctuating loads while considering performance and energy use. At the same time, heterogeneous processor architectures require schedulers to reason not only about timing but also about task placement across cores with different capabilities. Memory management has followed a similar path, shaped by the rise of remote memory access, disaggregated memory pools, and persistent non-volatile memory technologies. These developments challenge traditional paging and eviction strategies, encouraging adaptive, workload-aware placement across multiple memory tiers. While many of these approaches remain experimental and introduce added system complexity, they reflect a clear shift toward operating systems that coordinate compute and memory resources in a more responsive and integrated manner, aligned with real-world demands across cloud, desktop, and edge environments.***

Keywords: Operating Systems, Intelligent Scheduling, Heterogeneous Processors, Adaptive Memory Management, Persistent Memory

1. Introduction

Traditionally, operating systems have been based on deterministic process scheduling algorithms such as round-robin and simple priority schemes, and on traditional memory-management techniques, in particular paging with least-recently-used replacement [10]. The introduction of more complex hardware capabilities and increasing workload requirements has forced OS designers to go beyond these traditional techniques, revealing their shortcomings in environments ranging from cloud data centres to mobile devices [10]. In response, researchers are developing new scheduling and memory-management techniques that are better suited for heterogeneous hardware, massive scale, and dynamic workloads [9]. This paper surveys some of these developments, with specific reference to how real-world demands are informing more adaptive and intelligent OS strategies.

Learning-Driven Scheduling

OS scheduling is increasingly incorporating machine-learning techniques. Traditional schedulers rely on fixed rules that struggle to adapt to complex and rapidly changing workloads [1]. In contrast, a learning-driven scheduler can adjust its decisions dynamically. For example, reinforcement-learning-based prototypes predict the next task to schedule based on current system state and historical behavior, optimizing throughput or energy efficiency [1]. Such mechanisms allow the OS to anticipate workload bursts and adapt scheduling policies in real time [2]. Early experimental results show that AI-based scheduling can improve CPU utilization and reduce latency under variable loads [1], although challenges remain in terms of model overhead and robustness. Nevertheless, these schedulers represent a significant step away from rigid policies toward more autonomous resource management in future kernels.

Non-Homogeneous Core Scheduling

Modern processors increasingly adopt heterogeneous designs that combine high-performance cores with energy-efficient cores within a single system [3]. While this architecture improves performance per watt, it complicates scheduling decisions. The OS must determine not only when to execute a task, but also on which type of core it should run [4]. Compute-intensive threads benefit from large cores, while background tasks can be assigned to smaller cores to save energy [3]. Hardware-aware schedulers analyze task characteristics and dynamically assign them to appropriate cores [4]. Such policies are now being integrated into modern smartphone and desktop operating systems, enabling better performance for demanding applications and improved energy efficiency for lighter workloads [4]. As a result, OS scheduling is no longer a one-size-fits-all strategy but an adaptive, context-aware mechanism.

Remote Memory and Memory Disaggregation

Memory disaggregation separates memory from individual servers and makes it accessible over high-speed networks using technologies such as RDMA and CXL [5], [6]. This allows multiple machines to share a common memory pool, effectively expanding available memory beyond local physical limits [5]. The primary cost is increased latency: remote memory access is slower than local DRAM but still significantly faster than disk [5]. The OS must manage remote page faults similarly to local faults, while minimizing network overhead through caching and prefetching techniques [5]. Experiments in data-centre environments show that disaggregated memory is feasible, but performance depends critically on data placement and network latency [6]. Memory disaggregation thus offers increased flexibility at the cost of greater OS complexity.

Durability and Hybrid Memory Systems

Operating systems must also support persistent memory technologies and tiered memory hierarchies. Emerging non-volatile memory (NVM) devices provide near-DRAM speeds while retaining data across power failures [7]. These devices blur the boundary between memory and storage, enabling large datasets to be accessed at memory speeds [8]. However, OS designs must address durability, crash consistency, and higher access latencies compared to DRAM [7]. Most systems now adopt hybrid memory architectures that combine small pools of fast DRAM with larger pools of slower NVM [8]. The OS must decide which data reside in each tier, as global eviction policies such as LRU perform poorly in such environments [8]. Adaptive placement mechanisms track access patterns and migrate hot pages to fast memory while pushing colder data to slower tiers [8]. Some designs employ machine-learning predictors to proactively migrate pages before performance degrades [1], [8]. Early studies show that intelligent tiered memory management can significantly improve throughput and efficiency [8].

2. Conclusion

Overall, operating systems are evolving rapidly in response to new hardware features and workload demands. Scheduling and memory management, two core pillars of OS design, are being redefined through learning-driven schedulers, heterogeneity-aware task placement, disaggregated memory systems, and multi-level memory hierarchies [1], [3], [5], [8]. While many of these techniques remain experimental, they point toward a future in which OS kernels coordinate CPU and memory resources in a more holistic and adaptive manner [10]. As computing diversifies from cloud servers to edge devices, these advanced scheduling and memory-management schemes will play a central role in achieving new levels of performance and efficiency.

References

[1] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource Management with Deep Reinforcement Learning,” *Proc. 15th ACM Workshop on Hot Topics in Networks (HotNets)*, 2016.
[2] C. Delimitrou and C. Kozyrakis, “Quasar: Resource-Efficient and QoS-Aware Cluster Management,” *Proc. ASPLOS*, 2014.
[3] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen, “Single-ISA Heterogeneous Multi-Core Architectures,” *Proc. ISCA*, 2004.
[4] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narváez, and J. S. Emer, “Scheduling Heterogeneous Multi-Cores through Performance Impact Estimation (PIE),” *Proc. ISCA*, 2012.
[5] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. Shin, “Efficient Memory Disaggregation with Infiniswap,” *USENIX NSDI*, 2017.
[6] D. Gouk et al., “Practical Memory Disaggregation using Compute Express Link,” *WORDS (Workshop on Resource Disaggregation)*, 2022.
[7] J. Yang et al., “NV-Tree: Reducing Consistency Cost for NVM-based Systems,” *Proc. FAST*, 2015.
[8] J. Xu, S. Swanson, and Y. Chen, “NOVA: A Log-Structured File System for Hybrid Volatile/Non-Volatile Main Memories,” *USENIX FAST, 2016*.
[9] P. E. McKenney and J. D. Slingwine, “Read-Copy Update: Using Execution History to Solve Concurrency Problems,” *PDCS*, 1998.
[10] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed., Wiley, 2020.