IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

Emerging Trends in Virtual Memory

Dr. D. Thamaraiselvi[1] and Subhameera[2]
[1] Assistant Professor, [2] Student, (Computer Science & Engineering),
Sri Chandrasekharendra Saraswathi Viswa Maha Vidyalaya, Enathur, Kanchipuram

Abstract: *This paper reviews recent advances and innovations in a fundamental part of contemporary operating systems, namely virtual memory (VM), and aims to provide insights about VM, discuss its relevance in computing spaces, and examine futures that shape its ongoing development. As part of this review, we also present a case study that demonstrates real use cases, and conclude with suggested future work*

Keywords: Virtual memory, storage, memory, physical memory, RAM

I. INTRODUCTION

Virtual Memory is a prominent concept in computer architecture that abstracts physical memory to create an illusion to processes that there is large contiguous memory allocated exclusively to an application. As systems increase in scale and multitasking becomes prevalent, VM enables manageable memory access, process isolation, and process control. This paper presents a review of the fundamentals and explains why VM is more relevant than ever before.

II. VIRTUAL MEMORY

2.1 Definition

Virtual Memory is a memory management feature that presents processes as if they have contiguous and private memory, irrespective of whether the shared and fragmented physical memory exists.

2.2 Comparison: Physical Memory vs Virtual Memory

- Physical Memory (PM) is the amount of RAM modules installed in a machine.
- Virtual Memory (VM) simulates more memory than is physically available using hardware and/or software, usually with the aid of disk drives.
- VM allows systems to run larger programs and multitask efficiently by paging and swapping memory

2.3 Techniques in Virtual Memory

- Paging: It transforms memory into partitioned "pages" of fixed-size blocks, allowing for logical non-contiguous memory (multiple pages) in the physical memory.
- Segmentation: The physical memory is divided into segments by application logical divisions of memory usage such as code, stack, heap.
- Demand Paging: Adds pages to memory only when the page is required, in demand.
- Page Replacement Algorithms: It utilizes page replacement decisions when replacing memory (or swapping memory out to disk) pages such as LRU (Least Recently Used) and FIFO (First In First Out)

2.4 Virtual Memory in Modern Operating Systems

Modern operating systems like Windows, Linux, and Mac OS X utilize very sophisticated virtual memory model systems to manage applications, manage security, and multitask (Is it possible to effectively multitask without virtual memory?). With rapidly improving SSD, memory architectures and quickly faster disks, an effective virtual memory system is becoming even faster and more reliably.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 3, May 2025 Impact Factor: 7.67

III. NEED FOR INNOVATIONS IN VIRTUAL MEMORY

Applications are becoming larger and more complex and traditional VM systems are becoming more and more vexed with issues of performance, latency, and power consumption. What is needed is innovation to:

- Reduce page faults.
- Allow for better accessibility to memory.
- Improve memory in virtual and container environments.
- Use AI based memory prediction models.

IV. VIRTUAL MEMORY TRENDS ON THE RISE

As the computing demands shift, especially with the era of big data, AI, and cloud computing, VM systems must be smarter, quicker, and more efficient too. With the following list, some of the most important and recent trends are mentioned below:

4.1 Memory Compression

Memory compression is employed to cache more data in RAM by compressing less frequently accessed pages rather than swapping them out to disk. It accelerates performance and reduces use of slow disk I/O operations. Linux's zswap and zram are perfect examples of this technique. They allow compressed pages to be stored in RAM, significantly reducing latency.

4.2 Integration with Non-Volatile Memory (NVM)

Technologies like Intel Optane and 3D XPoint offer persistent memory that inherits RAM's performance and storage endurance. They can be mapped directly into virtual address space and provide faster access to large sets of data and support instant-on computing.

4.3 Hybrid Memory Systems

These systems now typically combine DRAM with other memory technologies such as NVM or SSDs to give a multi-level memory hierarchy. Operating systems nowadays are being engineered to manage such heterogeneity, dynamically moving memory pages between the fast and the slow levels based on usage.

4.4 AI-Supported Memory Management

Machine learning algorithms are used to forecast memory access patterns so that the system can pre-load or hold often accessed pages in RAM. The system reduces page faults and improves system responsiveness in general, especially in real-time and multitasking systems.

4.5 Virtualization and Cloud Optimization

With the introduction of virtual machines (VMs) and containers on cloud computing, memory overcommitment (use of more memory than is available physically) and VM live migration are now standard. To make these features work:
Ballooning is employed to reclaim memory from idle VMs.
Shared pages of memory (like Linux's Kernel SamePage Merging) reduce duplication. Memory hot-add dynamically adds available memory to VMs with zero downtime.
4.6 GPU and Specialized Memory Virtualization
With GPU processing now the standard, virtual memory is being used to handle unified memory across CPUs and GPUs. This enables developers to code as if there is shared memory, enhancing deep learning and data science operations.

V. CASE STUDY: LINUX ZSWAP – COMPRESSED CACHE FOR SWAP PAGES

5.1 Context

The Linux zswap functionality is a cache for compressed swap pages. It allows a system to be made more responsive when there is not sufficient memory. Rather than transferring pages to the slower disk-based swap space, zswap compresses the pages and keeps them in memory.

Operational Structure

When a page is evicted, zswap compresses the contents with a fast compression algorithm (e.g., LZ4 or ZSTD). The compressed page is cached in a memory pool (zpool). When RAM pressure builds or the compressed pool is full, older pages are decompressed and written to disk.

5.2 Advantages

- Better performance: It reduces the I/O operation and keeps the swapping activity within the RAM.
- Energy efficiency: Less disk access equals less power, and that is good news for mobile and embedded systems.
- Cost-effectiveness: Allowing systems with less RAM to perform bigger tasks.
- Real-World Application In low-end Chromebooks and Raspberry Pi setups, characterized by low RAM, the enabling of zswap has resulted in better multitasking performance and faster response times. Most Linux distributions have zswap as an enabled feature by default in desktop setups to improve user experience.

VI. FUTURE PLANS

- Upcoming VM systems will probably:
- Integrate more closely with AI/ML algorithms.
- Support large-scale distributed systems.
- Offer enhanced real-time processing capabilities.
- Leverage hardware-level capabilities such as memory disaggregation and intelligent memory controllers.
- In the future, VM can be extremely important in serverless computing and edge AI.In these, efficient use of resources is highly significant.

VII. CONCLUSION

Virtual Memory is a key aspect of operating system efficiency. As modern applications' demands continue to rise, there exists a need to enhance VM features to enhance the system's efficiency and speed. The trends highlighted in this paper indicate a promising future for VM, backed by future technologies and new concepts.

REFERENCES

[1] Ruan, H., Yang, Z., Zhu, T., & Lee, C. H. (2024). Enhancing Virtual Memory Management in Hybrid DRAM-NVM Systems Using AI Prediction. *IEEE Transactions on Computers*, 73(2), 312–324. https://doi.org/10.1109/TC.2023.3335678

[2] Liu, Y., Zhang, Q., & Chen, X. (2023). Unified Memory Virtualization for Heterogeneous Computing Systems. *ACM Transactions on Computer Systems*, 41(1), 1–29. https://doi.org/10.1145/3591120

[3] Wang, M., Liu, X., & Ma, H. (2022). A Memory-Aware Virtualization Framework for Cloud Computing. *IEEE Transactions on Cloud Computing*, 10(1), 54–65. https://doi.org/10.1109/TCC.2021.3061234

[4] Zhang, J., & Kim, H. (2021). Adaptive Memory Compression for Reducing Swap Overhead in Virtual Memory. *IEEE Transactions on Computers*, 70(12), 1961–1973. https://doi.org/10.1109/TC.2021.3088792

[5] Ramesh, D., & Ezhumalai, R. (2020). LSTM-based Emotion Detection with IoT Framework. *IEEE Access*, 8, 23578–23589. https://doi.org/10.1109/ACCESS.2020.9290364

[6] Dinesh, V., & Sridharan, R. (2019). AI-Driven Memory Access Prediction for Smart Virtual Memory Systems. *Future Generation Computer Systems*, 96, 168–179. https://doi.org/10.1016/j.future.2019.01.007

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 3, May 2025
Impact Factor: 7.67

[7] Tomar, A., & Sharma, N. (2018). Optimizing Virtual Memory for Real-Time Applications Using Machine Learning Techniques. *Procedia Computer Science*, 132, 786–793. https://doi.org/10.1016/j.procs.2018.05.123

[8] Kim, J., & Park, Y. (2017). Implementation of Memory Ballooning in KVM for Efficient VM Resource Allocation. *International Journal of Computer Applications*, 169(5), 22–27. https://doi.org/10.5120/ijca2017914350

[9] Chen, Y., & Wu, C. (2016). Improving Memory Utilization Using Compressed Swap Caches in Linux. *Journal of Systems and Software*, 119, 12–24. https://doi.org/10.1016/j.jss.2016.05.017

[10] Shafique, M., Garg, S., & Henkel, J. (2015). Hybrid Memory Management for Energy-Efficient Multicore Systems. *ACM Transactions on Embedded Computing Systems*, 14(3), 56:1–56:25. https://doi.org/10.1145/2714560