International Journal of Multidisciplinary Research and Publications

ISSN (Online): 2581-6187

Enhancing OS Memory Management Performance: A Review

Nihad Ramadhan Omar[1], Rezgar Hasan Saeed[2], Jihan Abdulazeez Ahmed[3], Shilan Bashar Muhammad[4], Zainab Salih Ageed[5], Zryan Najat Rashid[6]

[1]IT Management, Dohuk Polytechnic University, Duhok-Iraq, nihadro@yahoo.com
[2]Computer Science, Near East University, Cyprus, rezgarhasan1992@gmail.com
[3]Computer science, University of Duhok, Duhok-Iraq, drjihanrasool@uod.ac
[4]Mathmatic, University of Duhok, Duhok-Iraq, shilan.mohammed@uod.ac
[5]Translation, Nawroz University, Duhok-Iraq, zainab.ageed@nawroz.edu.krd
[6]Computer Network, Sulaimani Polytechnic University, Sulaimani-Iraq, zryan.rashid@spu.edu.iq

Abstract— *Memory management refers to all methods used in memory to store code and data, track use, and, where possible, retrieve memory space. This means that the physical chips and a logical address space are mapped through the memory map at a low level. Application programs could be used in higher-level virtual spaces with memory management unit (MMU) to create a contiguous memory impression. This paper proposed a review on operating system function and the rule of memory management unit in providing different techniques for various process in operating system. This paper shows the experiences of a group of researchers in operating systems development in many deferent techniques.*

Keywords— *Memory management, operating system, DRAM.*

I. Introduction

Applications have increased memory footprints, energy consumption and demand for output quickly in the Big Data and Cloud Computing age [1]. To meet these requirements, memory capacity is crucial, latency of memory access is reduced and energy efficiency of the memory improved [2]. During job performance, the bottleneck of a computer system is seriously affected [3]. Due to the data-intensive computational paradigm it has recently become more and more popular for large database processing, it usually has large data requirements in the memory subsystem of an inheritance OS (operating system) kernel of ten devices as the bottleneck of the system. As a matter of fact, the memory subsystem depends for its processing on the slow block storage of a device [4].

The DRAM power and bandwidth increases significantly in large data centers, with energy and consumption by the main memory systems [5]. Furthermore, the temporary exchange space created by OS on the storage device for the production of excessive, main memory pages tends to become an execution engine [6]. The performance engine of big data treatment was targeted mainly due to extensively accessing slow I/O block storage devices [7]. Solutions were sought. Significantly, all information in a system's main memory (DRAM) is widely used for fast processing in memory computation [2, 8]. Since modern computer systems are increasingly packaging core components on the processor chip, memory systems need to scale bandwidth proportional to deliver data to all core components [9].

The pin count for the mainframe chip is unfortunately the memory bandwidth dictated and this restricted memory bandwidth is one of the system performance bottlenecks [10]. compression of data is a promising way to increase the efficient memory system bandwidth. Prior compression work aims to achieve both compression capacity and bandwidth, by trying to fit the most pages in the main memory dependent on data compressibility [11, 12].

As these designs can change the effective memory capability during runtime, they require OS or hypervisor support to handle the memory capacity that change dynamically [13, 14]. Unfortunately, it means that memory compression solutions are not viable without coordination of the interface between both hardware vendors (for instance Intel, AMD) and OS vendors (Microsoft, Linux etc.) or limited solutions for systems that provide hardware and OS from the same vendor [15, 16].

Main memory is a part of the main components for recent computer systems, with a wider memory capability for various applications for handling increasingly explosive data [17]. DRAM is commonly used for several decades as a main memory, but due to physical restrictions, its mass and cost is not expected to increase further [18]. Nonvolatile memory in block storage devices, however, continues to grow due to technological improvements such as, the NAND flash memory and Intel® 3D Points memory used in SSDs [19]. Thus, hybrid memory systems that routine block devices as DRAM extensions, both in industry and in academia, are very popular as they can set up high-performance, large capacities and low-cost memories [20, 21].

II. Functions of Operating System

A. Management of Process

Operating system support for process management, process creation and removal. A mechanism for synchronization and communication between processes is also provided for process management [22]. The OS maintains the processor's tracking and process status [23]. A traffic controller is the software that does the job. It assigns the processor to perform the task of the processor, if the processor no longer needs a process [24].

B. Management of Memory

The main and secondary memory management is used for memory management. The module for memory management performs memory allocation and deallocation for the program [25]. The OS does different storage management tasks, it tracks the storage media of which memory part is being used and which memory part is not being used [26]. At the time of the process memory request, the operating system helps to allocate the memory [27]. If the process no longer requires memory, the memory will be deallocated [28]. The task of memory allocation is done with the help of the operating system in multi-programming [29].

C. Management of File

A directory is organized for fast or simple navigation and easy-to-use file system. These directories are made up of folders and other files [30]. It helps handle all file-related tasks, such as storage, recovery, sharing, naming and protection of files. It retains information tracking, location, user status, etc [31].

D. Management of Device

OS is responsible for the allocation and deallocation of the devices. It helps to monitor all devices [32]. The device communication through their respective drivers is carried out with the help of the operating system. It efficiently manages the device[33].

E. Management Secondary Storage

The secondary storage management is the responsibility of OS. The system has different storage levels that include primary, secondary and cache storage [34]. The instruction and data set are stored in primary memory or cache memory to reference the executed program [35].

F. Security

Security is the responsibility of the operating system, which prevents unauthorized access and threats from the data and information [36, 37].

G. Coordination between other software and user

The operating system co-ordinates other software and users. The operating system OS manages assemblies, translators, compilers and other software, as well as assigns them for various computer system users .

H. Networking

Distributed systems are a series of processors that do not share clock and memory hardware devices [38]. The processor communicates with one another with the help of the network.

I. Job accounting

The system functions to keep track of times and resources used by several jobs and users. Operating system provides a job accounting function [25].

J. Error detecting aids

The OS also performs the detection of errors. It continuously monitors or detects errors in the system and prevents errors in the system.

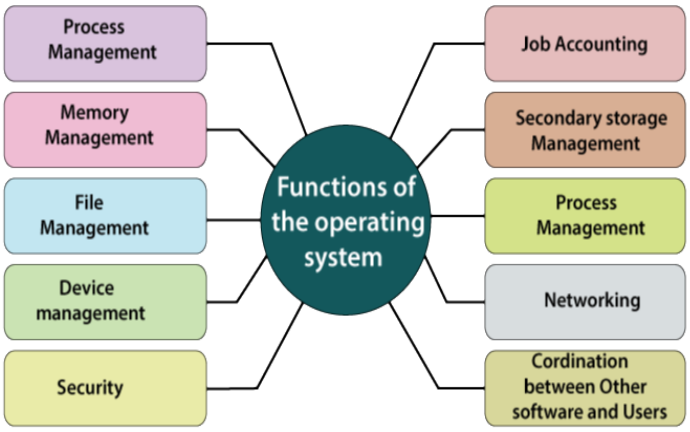

Fig. 1. Operating System Functions

III. MEMORY MANAGEMENT PERFORMANCE

Memory management offers different processes and threads for the allocation of memory and deallocation techniques [39]. OS offers two common memory allocation methods: static and dynamic . In static memory management OS assigns memory to a system that cannot be modified over time [40]. Dynamic management technology, however, offers flexibility in memory acquisition in runtime [41]. Static allocation cannot forecast the amount of memory needed, particularly in real-time scenarios [42]. Something like this can lead to over-supply of memory. If no allocation is made of the assigned memory, a dynamic allocation memory leak may result [23, 43]. Due to the size and costs of the devices, the memory size of sensors is restricted [44]. Static storage contains program code and dynamic storage contains runtime, buffer and stack variables [45].

In the Service Node (SN) system, the Memory Management System (MM) is controlled as distinct service or as part of the runtime management system in the Computer node memory assignment (CN) [25]. The following issues are addressed:

- selecting the best appropriate memory base on the allocated processing foundations.
- allowing synchronized, thread-safe memory allocation and deallocation though preventing fragmentation.
- carry out virtual to physical addresses, and vice versa.
- Runtime optimization performance. Picking the best appropriate memory In order to choose the best memory modules, the memory manager takes into account the following search criteria:
- The bandwidth between allocated processing elements and memory module.
- Memory module and its latency.
- Data transfer directions (input data/output data).
- The available space on the module.
- The load on routing and ports. Current criteria are defined according to the QoS requirements provided by the runtime resource manager to the memory manager. In addition, certain characteristics may change during runtime, depending on the target architecture. Generally, memory management systems are based on an algorithm

based on constantly updated data on the state of the resources, taking running decisions aiming to achieve.

- Assign a memory unit buffer next to the processing unit and release the memory units near unused processing units.
- leave free space to allocate high priority applications.
- The MMU performance is evaluated by various factors: memory, partial addresses, strategies, dynamism, speed versus fragmentation, location versus speed, fragmentation versus location etc. [46].

Main memory management is essential. Two problems are associated with complete system execution, with how much memory and memory management are provided in the work process [47]. The block should be distributed in a few languages when the program is ended with a memory block. The block is stamped unused in such conditions [48]. Space within Java is 'made accessible' because it is not available regularly [49]. This helps ensure that the Java waste product range is cleaned up in this area. The unused memory of the store can be verified.

The operator updates private information systems to reuse potential assignment criteria, thus showing that the block memory area is reused [50]. After transfer, the comparison continues to show the block. The system can never reach the point that is relocated [51]. The developer should make sure the specific does not try to follow the old reference to the block in the dealer in a word (such as C++) with unambiguous storage locations without a range of garbage [52].

The free space in the memory can be separated by store and delete memory operations in small components in the computer storage system [53]. The storage is currently used wastefully to reduce the limits and the framework implementation [54, 55]. Fragmentation circumstances hang on the system memory. Usually, memory space is gone. Moreover, because of its small size and memory squares, memory squares cannot be used. This is known as fragmentation [56]. During the stacking and exchange process there are numerous spaces left that are not able to stack a different procedure given their dimensions [57].

The concept storage is accessible but due to the dynamic allocation of special memory categories, its space is not accurate to stack another procedure [58]. One way to delegate small troughs is to make the memory allocation larger than the memory specified. Data fragmentation happens as a result of separating a series of memory into various non-neighboring portions [59].

IV. Literature Review

The new edge computer device memory resources management framework, TOMML, has been introduced by X LI et al. [60]. The observation is based upon the thought that the previously proposed MMBBT framework is not appropriate for existing users, that different optimization strategies cannot be integrated and that in various scenarios, users cannot change the optimization goals. TOMML tracks the microkernel's architectural patterns and uses all the MMBTB advantages. Real experimental results on Android systems show that the approach improves the efficiency of allocation from 12 to 20%. Moreover, a plugin is also made in edge computing to display the compatibility of the framed interface in the DRAM self-refreshing power issue. Experiments show, by using various mapping arrangements, that bank idleness can be improved by 6 to 25 per cent.

Co-operative Memorial Expansion (COMEX) was introduced by Srinuan et al. [2] to support disruption of fine grain kernel memory in networked systems. COMEX was developed and developed to dynamically expand its memory size to hold expelled pages in any networked computer, extending the OS kernel memory subsystem to accelerate execution on any machine. COMEX uses page tables based on Linux OS to control data transfer through low-latency RDMA links between remote page frames (memory nodes) and local page frames (in computer nodes). It achieves speeds with high runs that dwarf the memory size of the host, by evading much lighter disks like swap space (usually under a recent OS). COMEX is a lightweight kernel-level project and utilizes locality-aware kernel information, resulting in better kernel-prefetching functionality. It is fully transparent to applications and users for any commodity computing system connected to the RDMA-enabled fabric. Such a design approach is appropriate for any operating system that relies on page tables for virtual address mapping.

Ravi Kiran et al. [61] introduced the dual dedup system to enhance the read performance by eliminating unnecessary disk data duplicates from the cache. The duplication data is also deleted from the cache for increased storage efficiency. Dual-dedup is a very lightweight system as duplicate pages are not detected by themselves. The knowledge found by the disk deduplication subsystem is instead intelligently used. Real prototype system experiments show significant readability and latency improvements. Dual-dedup, for example, increases read output by 34 per cent with 25 percent duplication of data for FIO benchmarks.

While the considerable background power reduction using simple policies, further improved power efficiency would be achieved with more sophisticated policies to estimate memory use or reduce migration costs. OffDIMM with data-center workloads. Off-DIMM is a DRAM software-based DRAM PM based on online/off-line memory at OS levels. NS Kim et al. [62] have been proposed for operation of Off-DIMM. When an offline block is disconnected, a deep power down status is set for the subarray group. In the OS address space of a group or a DRAM subset the Off-DIMM maps a memory block. These experimental results show that Off-DIMM decreases background power by 24 percent on the basis of current memory utilization online-offline without significant overhead performance.

M Qureshi et al. [15] propose an easy design to achieve bandwidth advantages in memory compression while depending only on memory modules (NonECC), to make OS support more convenient. And the design uses a new inline mechanism for metadata, which allows the line to be compressed to be scanned by a specific marker word, eliminating the overhead access to metadata. Development of a low cost location forecast (LLP) which defines the line's position with 98 percent accuracy and a dynamic solution to disable compression when the advantages of compression are lower than overhead. These assessments show that PTMC provides a robust (no workload) acceleration of up to 73 percent with total overhead storage of less than 300 bytes.

L. Lio et al. [34] and S. Yang et al. [63] introduced Memos, A memory management framework that can list memory resources hierarchically throughout the entire memory hierarchy as well as cache, channels and main memory that consist of simultaneous DRAM and NVM. Memos can dynamically optimize memory hierarchy placement of information in response to memory access patterns, current resource use and memory medium features through the newly developed Kernel-level Monitoring Module that instances memory patterns through a combination of TLB monitoring with page walks and page migration engine. Channel scheduling is crucial in a hybrid DRAM-NVM system (e.g., MCHA), as multiple channels connect different memory types and offer different bandwidths. The overall system performance is improved by mapping data with appropriate memory kinds. Experimental results show that Memos is able to achieve high memory usage, improving system performance by approximately 20,0%, reducing the memory consumption by 82,5%, and improving the NVM life by up to 34X.

The gross-grain and fine grain delay models, along with use of Linux kernel changes and multiple runtime features, were implemented on a SoC-FPGA by Yu Omori et al. [64], In addition, the program variances between two models are evaluated by SPEC CPU programs. The fine grain model shows that the time of the program is run depends on the frequency of NVMM memory requests rather than on the cache hit ratio. Parallel bank levels and row buffer access points also affect memory access delays, and even when the former has a longer write-latency for four out of fourteen programs the fine grain model shows less runtime that ground grain.

H. Jang et al. [38] proposed a network-on-chip architecture that includes the MMU (NoC). By means of the approach proposed, NoC offers MMU functionality without changing processor design, making it easy for developers to leverage existing ULP lightweight processors and construct integrated systems that backing multi-processing. The design of Embedded NoC (MMNoC) is a prototype platform with MMNoC and dual RISC-V processors. The prototype platform is synthesized with the 28nm FD-SOI technology FPGA and Samsung to check the MMNoC's functionality and small capacity, scope and power overhead.

V. Comparision and Discusion

It is necessary to consider other researchers' efforts and experiments in the same area of the project to be undertaken in advance of any project and to move on from the point of completion. Therefore, as shown in the Table 1, some examples of the techniques used to improve maximum memory management performance such as noted by references [60], [64] and [38] in different key concepts like thread – oriented memory management layer (TOMML), nonvolatile main memory (NVMM) and memory management network on chip (MMNoC) respectively in order to achieve the maximum level of advantages. In other hand the rest researcher presented deferent key concepts for deferent approaches as shown in the table 1 from each references. The reader can note that each researcher proposed an example for memory management performance by using a deferent technology to achieve and maximize the benefit from the necessary goals.

TABLE I. Advantages of memory management

Ref.	Approaches	Key concepts	Advantages
[60]	Memory management	thread-oriented memory management layer (TOMML)	Inherits all the MMBTB advantages like using a thread on the Android platform to help optimize the thread and take full advantage of the thread behaviors
[2]	networked computing systems	cooperative memory expansion (COMEX)	support of memory disaggregation for executing diverse applications with large execution footprints
[61]	page cache management	Dual Deduplication-aware	Discloses to a page cache the redundancy knowledge detected by the deduplication level block layer, which may remove cache redundancy and avoid unnecessary reading requests
[62]	Power management	OffDIMM	Decreases background power by up to 24%, with low overhead performance
[15]	Transparent Memory-Compression (TMC)	Practical and Transparent Memory Compression (PTMC)	provides a robust (no slowdown of workload) speed up of up to 73 percent and can be executed with less than 300 byte overhead storage.
[34], [63]	hybrid memory management	Memos	can be deployed on systems equipped with Fast-Slow memories potentially
[64]	Memory management	nonvolatile main memory (NVMM)	Larger memory power and lower power consumption are achieved compared to traditional DRAM-based principal memory because NVMM requires no cooling processes
[38]	Memory management	Memory management network on chip (MMNoC)	It allows embedded hardware technologists to build a target platform to enable multiprocessing of existing lightweight processors (which normally don't have the MMU).

VI. Conclusion

The efficiency of several memory units has been identified as fundamental elements for improving the performance and scope for the application of computer technologies in memory units in existing data centers. A fundamental part of all systems is the memory management unit. Existing literary works have found that Memory is continuously reserved, evacuated, separated, reused and used by virtualization, and space usage needs to be improved. The study also focuses on various memory management strategies that can quickly set up frames and apply them. This paper looks at many operates on the MMU and the drawbacks. It clarifies every aspect of resource management. Their management mechanisms include process management, memory management, energy management, communication and file management. These approaches are further classified according to the formulations of their problems. An overview of the underlying idea and its advantages are discussed in each OS main approach.

References

[1] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, Z. N. Rashid, A. A. Salih, *et al.*, "A survey of data mining implementation in smart city applications," *Qubahan Academic Journal,* vol. 1, pp. 91-99, 2021.

[2] P. Srinuan, X. Yuan, and N.-F. Tzeng, "Cooperative memory expansion via OS kernel support for networked computing systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 31, pp. 2650-2667, 2020.

[3] H. R. Abdulqadir, S. R. Zeebaree, H. M. Shukur, M. M. Sadeeq, B. W. Salim, A. A. Salih, *et al.*, "A study of moving from cloud computing to fog computing," *Qubahan Academic Journal,* vol. 1, pp. 60-70, 2021.

[4] Z. S. Ageed, S. R. Zeebaree, M. A. Sadeeq, M. B. Abdulrazzaq, B. W. Salim, A. A. Salih, *et al.*, "A state of art survey for intelligent energy monitoring systems," *Asian Journal of Research in Computer Science,* pp. 46-61, 2021.

[5] B. T. Jijo, S. R. Zeebaree, R. R. Zebari, M. A. Sadeeq, A. B. Sallow, S. Mohsin, *et al.*, "A comprehensive survey of 5G mm-wave technology design challenges," *Asian Journal of Research in Computer Science,* pp. 1-20, 2021.

[6] F. Q. Kareem, S. R. Zeebaree, H. I. Dino, M. A. Sadeeq, Z. N. Rashid, D. A. Hasan, *et al.*, "A survey of optical fiber communications: challenges and processing time influences," *Asian Journal of Research in Computer Science,* pp. 48-58, 2021.

[7] S. M. S. A. Abdullah, S. Y. A. Ameen, M. A. Sadeeq, and S. Zeebaree, "Multimodal emotion recognition using deep learning," *Journal of Applied Science and Technology Trends,* vol. 2, pp. 52-58, 2021.

[8] M. A. Sadeeq and S. Zeebaree, "Energy management for internet of things via distributed systems," *Journal of Applied Science and Technology Trends,* vol. 2, pp. 59-71, 2021.

[9] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, H. S. Yahia, M. R. Mahmood, *et al.*, "Comprehensive survey of big data mining approaches in cloud systems," *Qubahan Academic Journal,* vol. 1, pp. 29-38, 2021.

[10] M. A. Omer, S. R. Zeebaree, M. A. Sadeeq, B. W. Salim, S. x Mohsin, Z. N. Rashid, *et al.*, "Efficiency of malware detection in android system: A survey," *Asian Journal of Research in Computer Science,* pp. 59-69, 2021.

[11] J. Y. Hur, "Representing contiguity in page table for memory management units," in *2017 IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC),* 2017, pp. 21-28.

[12] L. M. Abdulrahman, S. R. Zeebaree, S. F. Kak, M. A. Sadeeq, A.-Z. Adel, B. W. Salim, *et al.*, "A state of art for smart gateways issues and modification," *Asian Journal of Research in Computer Science,* pp. 1-13, 2021.

[13] I. M. Ibrahim, "Task scheduling algorithms in cloud computing: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* vol. 12, pp. 1041-1053, 2021.

[14] N. Omar, A. Sengur, and S. G. S. Al-Ali, "Cascaded deep learning-based efficient approach for license plate detection and recognition," *Expert Systems with Applications,* vol. 149, p. 113280, 2020.

[15] V. Young, S. Kariyappa, and M. K. Qureshi, "Enabling transparent memory-compression for commodity memory systems," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA),* 2019, pp. 570-581.

[16] A. S. Abdulraheem, A. A. Salih, A. I. Abdulla, M. A. Sadeeq, N. O. Salim, H. Abdullah, *et al.*, "Home automation system based on IoT," 2020.

[17] D. H. Maulud, S. R. Zeebaree, K. Jacksi, M. A. M. Sadeeq, and K. H. Sharif, "State of art for semantic analysis of natural language processing," *Qubahan Academic Journal,* vol. 1, pp. 21-28, 2021.

[18] A. A. Yazdeen, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, O. M. Ahmed, and R. R. Zebari, "FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review," *Qubahan Academic Journal,* vol. 1, pp. 8-16, 2021.

[19] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "IoT and Cloud computing issues, challenges and opportunities: A review," *Qubahan Academic Journal,* vol. 1, pp. 1-7, 2021.

[20] S. Imamura and E. Yoshida, "POSTER: AR-MMAP: Write Performance Improvement of Memory-Mapped File," in *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT),* 2019, pp. 493-494.

[21] S. R. Zebari and N. O. Yaseen, "Effects of Parallel Processing Implementation on Balanced Load-Division Depending on Distributed Memory Systems," *J. Univ. Anbar Pure Sci,* vol. 5, pp. 50-56, 2011.

[22] M. A. Sulaiman, M. Sadeeq, A. S. Abdulraheem, and A. I. Abdulla, "Analyzation Study for Gamification Examination Fields," *Technol. Rep. Kansai Univ,* vol. 62, pp. 2319-2328, 2020.

[23] A. Musaddiq, Y. B. Zikria, O. Hahm, H. Yu, A. K. Bashir, and S. W. Kim, "A survey on resource management in IoT operating systems," *IEEE Access,* vol. 6, pp. 8459-8482, 2018.

[24] M. Sadeeq, A. I. Abdulla, A. S. Abdulraheem, and Z. S. Ageed, "Impact of Electronic Commerce on Enterprise Business," *Technol. Rep. Kansai Univ,* vol. 62, pp. 2365-2378, 2020.

[25] A. Pupykina and G. Agosta, "Survey of memory management techniques for hpc and cloud computing," *IEEE Access,* vol. 7, pp. 167351-167373, 2019.

[26] Z. Ageed, M. R. Mahmood, M. Sadeeq, M. B. Abdulrazzaq, and H. Dino, "Cloud computing resources impacts on heavy-load parallel processing approaches," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 22, pp. 30-41, 2020.

[27] A. I. Abdulla, A. S. Abdulraheem, A. A. Salih, M. A. Sadeeq, A. J. Ahmed, B. M. Ferzor, *et al.*, "Internet of Things and Smart Home Security," *Technol. Rep. Kansai Univ,* vol. 62, pp. 2465-2476, 2020.

[28] A. Sallow, S. Zeebaree, R. Zebari, M. Mahmood, M. Abdulrazzaq, and M. Sadeeq, "Vaccine tracker," *SMS reminder system: Design and implementation,* 2020.

[29] A. A. Salih, S. R. Zeebaree, A. S. Abdulraheem, R. R. Zebari, M. A. Sadeeq, and O. M. Ahmed, "Evolution of Mobile Wireless Communication to 5G Revolution," *Technology Reports of Kansai University,* vol. 62, pp. 2139-2151, 2020.

[30] S. Giraddi, P. Kalwad, and S. Kanakareddi, "Teaching operating systems–programming assignments approach," *Journal of Engineering Education Transformations,* vol. 31, pp. 68-73, 2018.

[31] Z. S. Ageed, R. K. Ibrahim, and M. A. Sadeeq, "Unified Ontology Implementation of Cloud Computing for Distributed Systems," *Current Journal of Applied Science and Technology,* pp. 82-97, 2020.

[32] G. Aponso, "Effective memory management for mobile operating systems," *American Journal of Engineering Research (AJER),* vol. 246, 2017.

[33] N. Omar, A. M. Abdulazeez, A. Sengur, and S. G. S. Al-Ali, "Fused faster RCNNs for efficient detection of the license plates," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 19, pp. 974-982, 2020.

[34] L. Liu, S. Yang, L. Peng, and X. Li, "Hierarchical hybrid memory management in OS for tiered memory systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 30, pp. 2223-2236, 2019.

[35] A. B. Sallow, M. Sadeeq, R. R. Zebari, M. B. Abdulrazzaq, M. R. Mahmood, H. M. Shukur, *et al.*, "An Investigation for Mobile Malware Behavioral and Detection Techniques Based on Android Platform," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 22, pp. 14-20.

[36] Y. B. Zikria, S. W. Kim, O. Hahm, M. K. Afzal, and M. Y. Aalsalem, "Internet of Things (IoT) operating systems management: Opportunities, challenges, and solution," ed: Multidisciplinary Digital Publishing Institute, 2019.

[37] S. Zeebaree, S. Ameen, and M. Sadeeq, "Social media networks security threats, risks and recommendation: A case study in the kurdistan

region," *International Journal of Innovation, Creativity and Change,* vol. 13, pp. 349-365, 2020.
[38] H. Jang, K. Han, S. Lee, J.-J. Lee, and W. Lee, "MMNoC: Embedding Memory Management Units into Network-on-Chip for Lightweight Embedded Systems," *IEEE Access,* vol. 7, pp. 80011-80019, 2019.
[39] O. F. Mohammad, M. S. M. Rahim, S. R. M. Zeebaree, and F. Y. Ahmed, "A survey and analysis of the image encryption methods," *International Journal of Applied Engineering Research,* vol. 12, pp. 13265-13280, 2017.
[40] S. R. Zeebaree, K. Jacksi, and R. R. Zebari, "Impact analysis of SYN flood DDoS attack on HAProxy and NLB cluster-based web servers," *Indones. J. Electr. Eng. Comput. Sci,* vol. 19, pp. 510-517, 2020.
[41] H. M. Yasin, S. R. Zeebaree, M. A. Sadeeq, S. Y. Ameen, I. M. Ibrahim, R. R. Zebari, *et al.*, "IoT and ICT based Smart Water Management, Monitoring and Controlling System: A Review," *Asian Journal of Research in Computer Science,* pp. 42-56, 2021.
[42] S. Zeebaree and H. M. Yasin, "Arduino based remote controlling for home: power saving, security and protection," *International Journal of Scientific & Engineering Research,* vol. 5, pp. 266-272, 2014.
[43] K. Jacksi, N. Dimililer, and S. Zeebaree, "State of the art exploration systems for linked data: a review," *Int. J. Adv. Comput. Sci. Appl. IJACSA,* vol. 7, pp. 155-164, 2016.
[44] H. Malallah, S. R. Zeebaree, R. R. Zebari, M. A. Sadeeq, Z. S. Ageed, I. M. Ibrahim, *et al.*, "A Comprehensive Study of Kernel (Issues and Concepts) in Different Operating Systems," *Asian Journal of Research in Computer Science,* pp. 16-31, 2021.
[45] S. Zeebaree and I. Zebari, "Multilevel Client/Server Peer-to-Peer Video Broadcasting System," *International Journal of Scientific & Engineering Research,* vol. 5, 2014.
[46] Z. Yan, D. Lustig, D. Nellans, and A. Bhattacharjee, "Nimble page management for tiered memory systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 331-345.
[47] K. Bukkapatnam, C. K. Rekha, E. Kumaraswamy, and R. Vatti, "Smart Memory Management (SaMM) For Embedded Systems without MMU," in *IOP Conference Series: Materials Science and Engineering*, 2020, p. 032010.
[48] K. Jacksi, S. R. Zeebaree, and N. Dimililer, "LOD Explorer: Presenting the Web of Data," *Int. J. Adv. Comput. Sci. Appl. IJACSA,* vol. 9, 2018.
[49] D. Behera and U. R. Jena, "Detailed review on embedded MMU and their performance analysis on test benches," in *2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE)*, 2020, pp. 1-6.
[50] R. Ibrahim, S. Zeebaree, and K. Jacksi, "Survey on Semantic Similarity Based on Document Clustering," *Adv. sci. technol. eng. syst. j,* vol. 4, pp. 115-122, 2019.
[51] H. S. Yahia, S. R. Zeebaree, M. A. Sadeeq, N. O. Salim, S. F. Kak, A.-Z. Adel, *et al.*, "Comprehensive Survey for Cloud Computing Based Nature-Inspired Algorithms Optimization Scheduling," *Asian Journal of Research in Computer Science,* pp. 1-16, 2021.
[52] Z. N. Rashid, S. R. Zeebaree, and A. Shengul, "Design and analysis of proposed remote controlling distributed parallel computing system over the cloud," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019, pp. 118-123.
[53] I. M. Zebari, S. R. Zeebaree, and H. M. Yasin, "Real Time Video Streaming From Multi-Source Using Client-Server for Video Distribution," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 109-114.
[54] J. Zhang, S. H. Yeung, Y. Shu, B. He, and W. Wang, "Efficient memory management for gpu-based deep learning systems," *arXiv preprint arXiv:1903.06631,* 2019.
[55] H. M. Yasin, S. R. Zeebaree, and I. M. Zebari, "Arduino Based Automatic Irrigation System: Monitoring and SMS Controlling," in *2019 4th Scientific International Conference Najaf (SICN),* 2019, pp. 109-114.
[56] Y. Li, Y. Matsubara, and H. Takada, "A comparative analysis of RTOS and linux scalability on an embedded many-core processor," *Journal of Information Processing,* vol. 26, pp. 225-236, 2018.
[57] R. J. Hassan, S. R. Zeebaree, S. Y. Ameen, S. F. Kak, M. A. Sadeeq, Z. S. Ageed, *et al.*, "State of Art Survey for IoT Effects on Smart City Technology: Challenges, Opportunities, and Solutions," *Asian Journal of Research in Computer Science,* pp. 32-48, 2021.
[58] K. Jacksi, N. Dimililer, and S. R. Zeebaree, "A survey of exploratory search systems based on LOD resources," 2015.
[59] S. R. Zeebaree, H. M. Shukur, and B. K. Hussan, "Human resource management systems for enterprise organizations: A review," *Periodicals of Engineering and Natural Sciences (PEN),* vol. 7, pp. 660-669, 2019.
[60] Z. Zhu, F. Wu, J. Cao, X. Li, and G. Jia, "A thread-oriented memory resource management framework for mobile edge computing," *IEEE Access,* vol. 7, pp. 45881-45890, 2019.
[61] R. K. Boggavarapu and S. Jiang, "Deduplication-aware I/O Buffer Management in the Linux Kernel for Improved I/O Performance and Memory Utilization," in *2020 12th International Conference on Knowledge and Smart Technology (KST)*, pp. 70-74.
[62] S. Lee, N. S. Kim, and D. Kim, "Exploiting OS-Level Memory Offlining for DRAM Power Management," *IEEE Computer Architecture Letters,* vol. 18, pp. 141-144, 2019.
[63] L. Liu, M. Xie, and H. Yang, "Memos: revisiting hybrid memory management in modern operating system," *arXiv preprint arXiv:1703.07725,* 2017.
[64] Y. Omori and K. Kimura, "Performance Evaluation on NVMM Emulator Employing Fine-Grain Delay Injection," in *2019 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, 2019, pp. 1-6.