Survey Paper on Memory Allocation Systems

Kuldeep Vayadande, Aishwarya Pujari, Arvind Shelke, Sakshi Suryawanshi4,

Siddhant Deshpande, Suyog Savalkar

Vishwakarma Institute of Technology, Upper Indira Nagar, Bibwewadi, Pune, Maharashtra 411037, India

Abstract.

In the world of computation, we expect computers to perform various tasks at the same time, to achieve this goal multitasking and memory allocation must be done properly. Operating system performs a vital role in allocating memory to the processes. Memory Allocation techniques are primarily used for effective memory management. Moving processes between both the main memory and disc during process execution is another duty of the operating system. This paper discusses various techniques for memory allocation systems which helps users in allocating and deallocating memory as per their requirements.

Keywords. Memory, Allocation, Deallocation, Static, Dynamic, Contiguous

1. INTRODUCTION

A computer system's primary objective is to run applications. Both instruction and data are included in the programme. We can say that memory is a collection of instructions and data. A program that we are going to execute must be stored in the main memory. Memory is divided into large arrays. Each location has its address. Various memory allocation techniques are used for this purpose and the choice of technique depends upon the situation. Memory Allocation in an operating system is a method used to move in or move out of processes from the main memory to the disk at the time of process execution to achieve the aim of multiprogramming. The task of loading the process into the main memory is done by the loader. There are two types of loaders which are as follows:

- Static loading: In this type of loading, a complete program is loaded into the main memory before execution.
- Dynamic loading: In this type of loading, program load into the main memory as per demand.

Memory allocation can be done by using following methods:

- Contiguous Memory Allocation: Contiguous memory allocation is one of the oldest memory techniques used for the allocation of memory to the processes. In this type of allocation, memory is divided into consecutive memory blocks. Before execution, all the processes are in the waiting queue; these processes must be brought into the main memory for execution. Before this, size of the process is compared to the amount of contiguous memory available. If enough memory is found, then memory gets allocated to the process and the process starts executing. Multiple Partition Allocating: When a process needs memory in this sort of partitioning, the process is loaded into the free partition after being brought from the input queue.

- Manual Memory Allocation: This type of allocation is done by the programmer manually. A programming language such as C, C++, C# etc. still supports this type of memory allocation
- Fixed Partition Allocating: In this type of partitioning, the operating system maintains a table that shows the used memory and available memory by the processes. This available memory is known as a "Hole". When the process arrives in the waiting queue and requests memory we search for free available space. If enough space is found then the process gets memory and the execution of the process starts. While allocating memory to the process problem may arrive from the available free hole which free hole is used to fulfil the need of the process of size 'n'.

This problem is known as the dynamic storage allocation problem. The following are solutions to this problem:

- First Fit: In the first fit, the first free available hole which satisfies the memory request of the process is allocated to the process.
- Best Fit: In the best fit, the smallest available hole which is big as per the requirement of the process is allocated.
- Worst Fit: In the worst fit the biggest available hole is allocated to the process. This causes a waste of memory.

When a process terminates, the partition gets deallocated and now it is available for other processes. Another important terminology related to memory allocation is fragmentation. Fragmentation: After the execution of the process, it is removed from memory which creates a small hole in memory which is known as fragmentation. This causes a waste of memory. So, to achieve the goal of multiprogramming we must reduce memory wastage. There are two types of fragmentation:

- Internal Fragmentation: This fragmentation causes when memory is allocated to the processes more than their requirement.
- External Fragmentation: This fragmentation is caused when we have free memory blocks but we don't assign processes to that block.

2. LITERATURE REVIEW

In this paper [1], the author discusses operating system's memory management, along with the fundamentals of operating system segmentation and memory allocation. Additionally, the fundamentals of dynamic memory and virtual memory management are discussed in this work. The paper [2] is based on an examination of first-fit and dynamic equi-dimensional memory allocation arithmetic; this work offers a novel method of managing dynamic memory for embedded systems. The addition of dynamic equi-dimension memory tables in the first-fit arithmetic eliminates the effect of the length of the allocated linked table on the time to release memory blocks. The aim of the study [3] was to identify the contiguous memory allocation mechanism with the lowest fragmentation. According to the authors in paper [4], from a security standpoint, it is preferable to prevent code pointers (such as return addresses and function pointers) from being leaked. To protect code pointers, isolation methods have always been the preferred option. These methods incur significant

performance overhead because additional instructions must be instrumented for frequent authority switching or bound checking. The study in paper [5] offers three allocating strategies for dealing with external fragmentation. First-fit, Best-fit, and Worst-fit are these algorithms. On produced virtual traces, the author compared the performance of three methods and gave their implementations. This research study [6] discusses various memory allocation strategies and compares them in terms of the internal fragmentation they produce reaction time, memory footprint, and allocation and deallocation durations. Memory allocation has a significant impact on many computer systems, including multi-user systems, virtual machines, cloud-based services and many other systems. This paper [7] introduces a novel memory allocation strategy for on-demand (online) web applications, based on sequential fits and zoning. The study in paper [8] is to explore the magnitude and efficiency of artificial models of behaviour in programme allocation. If these models are sufficiently precise, offer a compelling substitute for algorithm assessment based on trace-driven simulation by real tracks. Memory access latency is significantly influenced by memory management and array binding methods. This study [9] presents a method for memory allocation and array binding that is efficient. The authors have also implemented buffer allocation for such arrays that have been used most frequently in an effort to reduce the number of requests to off-chip DRAMs. To manage memory layouts, several memory management systems use buddy memory allocation techniques. Using the interactive theorem prover Isabelle/HOL, the authors of this study [10] construct and formally validate a memory allocation model that preserves functional efficiency and security features.

TABLE 1 COMPARISON TABLE

Sr. No.	Authors	Title	Approach	Remark	Conclusion
1.	Durgesh Raguvanshi, 2018	Memory Management in the Operating System[1]	In this paper, the operating system's memory management is discussed, along with the fundamentals of operating system segmentation and memory allocation.	1.Space loss is caused by static memory allocation. 2.Longer execution time is necessary.	The fundamental idea of how an operating system works, how its memory is segmented, and what memory management is all about.

2.	Fuqing vu, Max Q-H, 2009	The study and improvement of memory management based on OS[2]	This paper presents a unique approach to managing dynamic memory for embedded systems.	The length of the allocated linked table is no longer a factor in how long it takes to release memory blocks	In this work, the memory allocation of the SOS OS is described briefly,
3.	A. K Mandal. Dilip Kumar Baruah, Jogamoh Medak, Neelutpol Gogoi, ParthaPratim Gogoi	Critical Scrutiny of Memory Allocation Algorithm[3]	This study's primary objective is to determine the contiguous memory allocation mechanism with the lowest fragmentation.	Internal fragmentation may or may not happen in fixed-sized partitions, but external fragmentation never happens since leftover space from a partition cannot be given to another process.	The worst performer in a variable-sized partition might not be the worst fit. The next fit algorithm, which looks for the first unoccupied division.
4.	Jiameng, YingRui, HouLutan, ZhaoFengkai, Yuan Pengh, ZhaoDan, Meng, 2022	A lightweight memory page management extension to prevent code pointer leakage[4]	According to the authors, from a security standpoint, it is preferable to prevent code pointers (such as return addresses and function pointers) from being leaked.	The solution does not require extending the cache and memory architecture as compared to typical tag architecture.	This investigation that employs conventional isolation solutions to stop code pointer leakage causes large performance overheads.

3. GRAPH OF COMPARISON

Table no. 2 discusses how each algorithm performs while allocating to different processes. Here three algorithms are listed. Which are first, best and worst fit memory algorithms. According to the total size of memory and total size of process in kb, amount of internal fragmentation is shown in table. Also, percentage of process allocation and internal

fragmentation are shown. Internal fragmentation value represents the difference between allocated memory and demanded memory. If we compare first and worst fit cases then it is seen that there is more percentile of internal fragmentation in case of worst fit memory algorithm. Whereas it is seen that best fit memory algorithms perform best as it shows less percentage of internal fragmentation as compared to other algorithms.

TABLE 2 MEMORY UTILISATION IN FIXED SIZED PARTITIONS AND INTERNAL FRAGMENTATION.

Algorithm	Total memory allocate (In KB)	Total size of process allocation (In KB)	% of process allocation	Internal fragmentation (In KB)	% of internal fragmentation
First Fit	1300	760	69.72	540	31.76
Best Fit	1600	1090	100	510	30.00
Worst Fit	1300	760	69.72	640	37.65

Fig. 1. Represents comparison graphs for first, best and worst fit memory algorithms. Here the x-axis represents the size of the process and the y-axis represents the number of blocks allocated to the process for different memory algorithms. By observing the graph, it is found that first fit technique sometimes performs as good as best fit technique whereas sometimes it allocates memory which is too small to hold the process. Worst fit algorithms give very less performance as compared to first fit and best fit. Best fit memory algorithm gives better performance as compared to both algorithms.

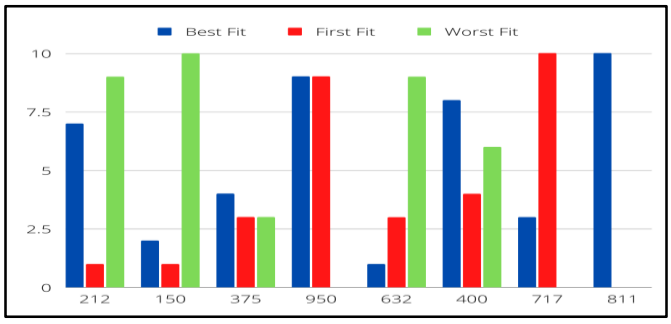

Figure 1 Analysis of Three Allocation Algorithms for Performance.

Table 3. Shows the comparison amongst other memory allocation algorithms based on three parameters which are fragmentation, response and memory footprint. Fragmentation is of type either external or internal in both conditions causing memory wastage. Memory footprint is the amount of main memory occupied by a process that uses any one of the algorithms in the table. So basically, good performing is the one who has smaller

fragmentation, faster response time and min value for memory footprint. It is found that TLSF is a good performer amongst all of the algorithms.

TABLE 3 FRAGMENTATION IT CREATES AND REACTION TIME.

Algorithm	Fragmentation	Response	Memory Footprint
Buddy System	Large	Fast	Max
Sequential Fit	Large	Slow	Max
Segregated Fit	Large	Fast	Max
Index Fit	Large	Fast	Max
Bit mapped Fit	Large	Fast	Max
TLSF	Smaller	Faster	Min
Hoard	Small	Faster	Min
Tertiary Buddy	Small	Fast	Min

4. SCOPE OF IMPLEMENTATION

In this survey paper, we have discussed the importance of memory allocation techniques in the operating system. Also, we have discussed how the processes get memory for execution. Process of swapping which is used for swapping lower priority processes with higher ones. Problems such as fragmentation and dynamic memory allocation occur while the execution and allocation of memory processes can be solved using multiple partition allocation. Algorithms such as first fit, best fit, and worst fit can be used to tackle the problem of dynamic memory allocation.

5. CONCLUSION

The main aim of this review is to discuss the various Memory allocation techniques. From loading the process into memory to the execution of the process there are many techniques used by operating systems. This paper also gives the brief information about various terminologies related to memory allocation such as static and dynamic loaders that load the process into main memory. After analysing various papers some problems get highlighted such as dynamic memory allocation. This paper also discusses solutions to these problems. Algorithms such as first fit algorithm, best fit algorithm, and worst fit algorithm can be used effectively to solve the problem of dynamic memory allocation. In the above discussion, the problem of fragmentation is also discussed. This causes wastage of resources. This problem

occurs when small holes are created in memory after the execution of the process. Methods such as multiple partition allocation can be used to tackle such problems. Our study also encourages that various algorithms and methods related to memory allocation are used but nobody's perfect one can use according to requirements.

REFERENCES

[1] Durgesh Raguvanshi, “Memory Management in the Operating System", Volume 2, Issue 5, July Aug 2018. (Raghuvanshi)

[2] Fuqing vu, Max Q-H, “The study and improvement of memory management based on OS ”, 2009.

[3] Amit Kumar Mandal. Dilip Kumar Baruah, Jagamohan Medak, Neelutpol Gogoi, ParthaPratim Gogoi, “Critical Scrutiny of Memory Allocation Algorithm”, 2020.

[4] Jiameng, Ying Rui, Hou Lutan, Zhao Fengkai, Yuan Pengh, ZhaoDan, Meng, “A lightweight memory page management extension to prevent code pointer leakage”, 2022.

[5] Lae Wah Htun, Moh Moh Myint Kay, Aye Aye Cho, “Analysis of Allocation Algorithms in Memory Management ”, 2019.

[6] Muhammad Abdullah Awais, “Memory Management: Challenges and Techniques For Traditional Memory Allocation Algorithms in Relation with Today's Real Time Needs”, 2016

[7] Anwar Al-Yatama, Imtiaz Ahmad & Naelah Al-Dabbous, “Memory allocation algorithm for cloud services ”, 2017

[8] Zorn, B., & Grunwald, D. , “Evaluating models of memory allocation”, ACM Transactions on Modeling and Computer Simulation,107–131, doi:10.1145/174619.174624, 1994

[9] Khan, S. D., & Shin, H. (2009). “Effective memory access optimization by memory delay modelling, memory allocation, and buffer allocation”, 2009 International SoC Design Conference (ISOCC). doi:10.1109/socdc.2009.5423893

[10] Jiang, K., Sanan, D., Zhao, Y., Kan, S., & Liu, Y. (2019). A Formally Verified Buddy Memory Allocation Model. 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS). doi:10.1109/iceccs.2019.00023

[11] L. W. Htun, M. M. M. Kay, A. A. Cho.,“Analysis Of Allocation Algorithms In Memory Management” IJTSRD, Vol. 3, No. 5, August 2019

Biographies

Kuldeep Vayadande is working as Assistant Professor in Dept. Of AI and DS. He has completed PhD in Computers Science and Engineering and having 14 Years of Teaching Experience. Published various papers in International Journals. His area of Specialization includes Operating System, System Programming, Information Security, Automata Theory and Cloud Computing. He is also working as Reviewer for various International Journals.

Aishwarya Pujari is currently pursuing B. Tech. in AI and Data Science from Vishwakarma Institute of Technology, Pune. She completed her schooling and college from D.G. Tatakare High School, Kolad. She completed her diploma from Institute of Petrochemical Engineering, Lonere.

Arvind Shelke is currently pursuing B. Tech. in AI and Data Science from Vishwakarma Institute of Technology, Pune. He completed his schooling and college from Gurukul Second Senior College, Beed.

Sakshi Suryawnshi is currently pursuing B. Tech. in AI and Data Science from Vishwakarma Institute of Technology, Pune. She completed her schooling and college from Kendriya Vidyalaya No.1 AFS Pune.

Siddhant Deshpande is currently pursuing B. Tech. in AI and Data Science from Vishwakarma Institute of Technology, Pune. He completed his schooling from S.P.M.English School, Pune. He completed his college from Parashurambhau College, Pune.

Suyog Savalkar is currently pursuing B. Tech. in AI and Data Science from Vishwakarma Institute of Technology, Pune. He completed his schooling from Kamalnayan Bajaj school, Pune. He completed his college from City pride Junior College, Pune.