DOI: 10.36108/ujees/2202.40.0270

UNIOSUN Journal of Engineering and Environmental Sciences. Vol. 4 No. 1. Sept. 2022

Comparative Analysis and Performance Evaluation of Contiguous Memory Techniques

Adeleke, I.A.

Abstract: Memory is a resource that must be carefully managed in computing systems due to its importance in job executions and in saving information. This paper is based on the techniques that operating systems use to manage memory allocation through simulation in allocating processes and data to partitioned memory. The physical memory of the computer system was modeled and simulation was performed under four contiguous allocation techniques namely; First-fit, Next-fit, Best-fit and Worst-fit with different percentages of free memory availability. Jobs and processes are assigned to the memory of a computer system that contained 512 kilobytes (KB) as memory capacity. Each of these techniques was tested and the result obtained revealed the capacity of memory wastage by each of them. This showed that the four tested algorithms do not optimize the storage concurrently. Thus, it was discovered that at 10% memory free, only the worst-fit algorithm had the highest memory wastage which is 120 KB while the other three techniques had the same memory wastage of 115 KB. The result repeated itself at 50% memory free in that worst-fit had 296 KB as the highest memory wastage. Therefore, out of the techniques considered, it was noticed that the worst-fit wasted the highest memory with a total of 1047 KB while the best-fit had the lowest memory wastage with a total of 1000 KB.

Keywords: Allocation, best-fit, contiguous, memory, processes, worst-fit

I. Introduction

Computer memory is expected to be kept in proper utilization for the users to adequately enjoy its usage just as human memory should be well managed for proper coordination. The efficiency of several memory units has been identified as fundamental element for improving the performance and scope of the application of computer technologies [1]. This computer memory is under the control of software written by the computer manufacturers to stand as an intermediary between the user and the system which is recognized as an operating system. The operating system manages the memory by allocating processes to it and recovers it when execution is terminated as one of its major responsibilities to avoid memory wastage. Thus, memory management technique is a technique that the operating system uses to allocate processes to the memory by assigning them to a specific memory location for their execution and recovery of the memory when the process's execution terminates or any other condition causes the process to free the memory. [2] Expressed memory management as all methods used in memory to store code and data, track use, and, where possible, retrieve memory space. [3] Equally opined that memory management is the task carried out by the Operating System (OS) and hardware to accommodate multiple processes in main memory. This has proved the significance of managing computer memory for the effective functionality of a computer set if our system will not be battling with frequent series of interruptions, hanging and wastage of resources.

Adeleke, I.A (Department of Computer Science, Emmanuel Alayande College of Education, Oyo State, Nigeria)
Corresponding author: adeleke_israel@yahoo.com
Phone Number: +234-706-222-9188
Submitted: 21-01-2022
Accepted:06-06-2022

In the recent era of computing, applications of an operating system cannot survive without efficient memory management, especially if an application has to be under review load for an undefined long time. Resources must be utilized efficiently to enhance performance [4]. Memory management offers different processes and threads for the allocation of memory and deallocation techniques [5]. The module for memory management performs memory allocation and deallocation for the program [6]. The operating system 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 [7]. At the time of the process memory request, the operating system helps to allocate the memory [8]. If the process no longer requires memory, the memory will be deallocated [9]. The task of memory allocation is done with the help of the operating system in multi-programming [10]. OS offers two common memory allocation methods which are static and dynamic. In static memory management, OS assigns memory to a system that cannot be modified over time [11]. Static allocation cannot forecast the amount of memory needed, particularly in real-time scenarios [12]. Dynamic management technology, however, offers flexibility in memory acquisition in runtime [13].

Meanwhile, **d**uring job performance, the bottleneck of a computer system is seriously affected [14]. Consequently, it is apparent from the work of computer scientists that effective and efficient main memory management and virtual memory management in computer systems will undoubtedly improve the computer system's performance by increasing throughput and processor utilization, decreasing the response time and turnaround time [15]. Since 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 [16]**.** It is part of the computer which stores data and is an important resource of the computer which should be managed carefully by the memory manager. The speed of a computer system depends on the way of managing different types of memory in the computer system. Unlike data on a hard disk, the data that lives in the memory cannot be saved and will be lost when the application quits or the computer is powered off. Also, if only a few processes can be kept in the main memory, then much of the time all processes will be waiting for I/O and the CPU will be idle. Hence, memory needs to be allocated efficiently in order to pack as many processes into memory as possible. The OS manages memory by allocating available memory to different processes and applications so that running applications have enough memory to perform their functions. This memory must be fairly allocated for high processor utilization and systematic flow of information between main and secondary memory.

Therefore, in order to maximally utilize the computer memory, memory partition is highly imperative. Memory partition is the system by which the memory of a computer system is divided into sections for use by the resident programs which may be fixed, variable or dynamic partitioning. Fixed partitioning is the system of dividing memory into non-overlapping (static) sizes in which the partition of memory apportioned to an active process is unalterable during the period of existence of a process. In general, static memory allocation is the allocation of memory at compile time, before the associated program is executed, unlike dynamic memory allocation or automatic memory allocation where memory is allocated as required at run time [17].

This is one of the simplest methods for allocating memory which is to divide memory into several fixed-sized partitions. This static memory allocation categorizes into fixed equal-size partitions in which the main memory is divided into equal number of fixed sized partitions and operating system occupies some fixed portion and remaining portion of main memory is available for user's processes. Any process whose size is less than or equal to the partition size can be loaded into any available partition. It supports multiprogramming. Variable partitioning is the system of dividing memory into non-overlapping (unmovable/static) but variable sizes. This system of partitioning is more flexible than the fixed partitioning configuration, but it is still not the most ideal solution. Small processes fit into small partitions and large processes fit into larger partitions [18]. Fixed variable size partitions overcome the disadvantage present in fixed equal size partitioning in that if a program is too big to fit into a partition of fixed equal size partition is a serious issue.The Fixed Partitioning approach has a severe loss of memory utilization for processes that exhibit a wide variance of locality size [19]. Dynamic memory allocation is a memory management technique in which a program can request and return memory while it is executing [20]. It is discovered as the best suited for workloads that have regular and predictable fluctuations in memory demands which can be achieved using certain functions like malloc(), calloc(), realloc(), free in C and "new", "delete" in C++to get memory dynamically. In dynamic Memory allocation, memory is allocated during run-time in heap and this is used when the size of memory is variable and is known only during run-time.

II. Materials and Methods

In order to achieve the setup goal, the physical memory is simulated and implemented by using bit-map approach in which the memory was divided into a small cell that could hold only one bit of data. This is done as large as several kilobytes, corresponding to each allocation unit is a bit in the bit-map which is zero (0) if the unit is free and one (1) if it is used. This was implemented by using array data structure in which each array entry can hold only one bit of data. Then, the input queue of processes was defined by using an array of processes where each entry determined the size of the process. Also, a memory table was defined which is a record type, containing information about each memory space, that indicates if the memory is free or which process is currently allocated to it and the duration at which the process has to be in the memory. The four memory allocation techniques are simulated and implemented with their respective algorithms. This was therefore assigned process of the queue in the memory space and keeping the track in the memory table.

Finally, both the total memory small spaces not used by each technique are summed and compared for different memory availability by using table and plotting graph in which the differences are clearly highlighted. This was implemented by using an object-oriented language called Delphi 6 programming language when partitioned memory was examined within the range of 10% to 50% free memory availability.

A. Different Types of Contiguous Memory Allocation Techniques

Having partitioned the memory and the processes are already on a queue the next thing for the process is to search for the free block in

the memory using any ***fit*** algorithms but priority is to be given to the process/job in front of the queue. There are four types of memory allocation techniques that are well pronounced namely;

i. Best-fit: It chooses the block, that is closest in size to the given request from the beginning to the ending free blocks. This strategy produces the smallest leftover hole. When a process needs memory, the operating system will allocate the smallest memory space that will fulfill the memory requirement for the process. So, for example, if there are the following free memory blocks: 10 KB, 25 KB, 30 KB, 15 KB, 8KB and 20 KB, and a process needs 12 KB to run, it will be assigned the 15 KB space. This is because there is no 12 KB space, and the 10KB space will be too small, but the 25 KB will be too large. Even though the 15 KB space is a little larger than what is required by the process, this is the best space available for the process.

ii. First-fit: It begins to scan memory from the beginning and chooses the first available block which is large enough. Searching can start either at the beginning of the set of blocks or where the previous first-fit search ended. We can stop searching as soon as we find a free block that is large enough.

iii. Worst-fit: Worst-fit memory allocation allocates free available block to the new job [21]. That is, it allocates the largest block but by searching the entire list, unless it is sorted by size. This strategy produces the largest leftover hole which may be more useful than the smaller leftover hole from a best-fit approach.

iv. Next-fit: It begins to scan memory from the location of the last placement and chooses the next available block. In the figure below the last allocated block is 18k, thus it starts from this position and the next block itself can accommodate this 20K block in place of 36K free block. It leads to the wastage of 16KB space.

B. Algorithms of How Process/Job is Assigned

Here are the algorithms that had been utilized to achieve the total amount of memory wastage during the execution of the programs. Table 1 represents the initial algorithm which other algorithms are depending upon.

i. First-fit Approach

In first -fit algorithm, the memory manager scans along the list of segments until it finds a space that is big enough. The space is then broken up into two pieces, one for the process and one for the unused memory except in the unlikely case of an exact fit. First fit is a fast algorithm because it searches as little as possible. The algorithm for first-fit is as presented in Table 2.

ii. Next-fit Approach

It works the same way as first-fit except that it keeps track of where it is when it finds a suitable space. The next time it is called, it starts searching from where it is left off instead of always at the beginning as first-fit normally does. The algorithm that wa*s* employed for this is written in Table 3.

iii. Best-fit approach

Best-fit searches the entire list of the partitioned memory and takes the smallest space that is adequate enough. Rather than breaking up a big space that might be needed later, best fir tries to find a space that is close to the actual size needed. The algorithm that doe*s* this is written in Table 4

iv. Worst-fit approach

This always takes the largest available space so that the space broken off will be big enough to

hold the process or job. Table 5 was presented to show the algorithm of Worst-fit approach.

Table 1: Initial algorithm

STEP 1	*If there still exist job/process in the queue and there exist free block in the memory*
STEP 2	*Then pick the job from the front of queue that can fit that block and assign it to the block*
STEP 3	*If process has completed its execution*
STEP 4	*Remove it from the memory block*
STEP 5	*Put it on the set of executed job*
STEP 6	*Free that memory partition*
STEP 7	*Go to the incoming queue and start again Until no more job on the queue*

Table 2: First-fit algorithm

STEP 1	*If there still exist job in the queue*
STEP 2	*If there are several free blocks (spaces) in the memory*
STEP 3	*Pick the job from the front of the queue*
STEP 4	*Allocate it to the first space that is big enough to accommodate it*
STEP 5	*Return to the beginning of the activity*
STEP 6	*Repeat the exercise until there is no more job on the queue*
STEP 7	*Take note of the memory wasted for the whole exercise*

Table 3: Next-fit Algorithm

STEP 1	*If there exist several free space on the memory*
STEP 2	*If there are jobs to be allocated into the memory*
STEP 3	*Pick job from the set of jobs*
STEP 4	*Allocate it to any first free space big enough to accommodate it*
STEP 5	*Pick another job*
STEP 6	*Start searching from where you stopped before*
STEP 7	*Allocate the job to the next big enough free space*
STEP 8	*If there is no free space enough to accommodate the job*
STEP 9	*Return to the beginning of the memory partition*
STEP 10	*Continue with the searching until you get free space that can accommodate it*
STEP 11	*Find the total memory wasted during allocation exercise*

.

Table 4: Best-fit Algorithm

STEP 1	*If there still exist job in the queue as well as free memory space in the memory*
STEP 2	*Pick a job from the queue and allocate it to the smallest free hole that is adequate to accommodate it*
STEP 3	*Repeat the exercise until there is no more job on a queue*
STEP 4	*Terminate the process*
STEP 5	*Determine the memory wasted all through*

Table 5: Worst-fit Algorithm

STEP 1	*If there exist job in the queue and there exist free memory space*
STEP 2	*Pick job from the front of the queue*
STEP 3	*Assign it to the largest free space large enough to accommodate it*
STEP 4	*Return to the beginning of the operation if there are still jobs and free spaces*
STEP 5	*Terminate the work if there is no more job*
STEP 6	*Record the memory wasted to do the job allocation*

III. Result and Discussions

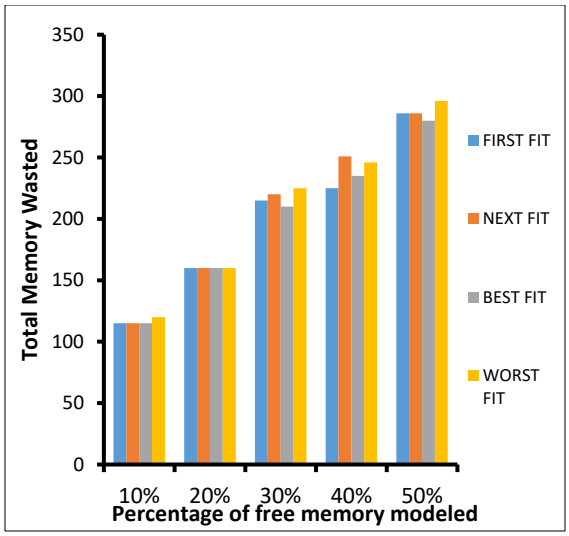
Having modelled the physical memory of a computer system using fixed equal-size partition method system and implemented a simulation of four contiguous memory allocation techniques (best-fit, next-fit, worst-fit and first-fit) under the set up range of memory availability, it was discovered that a lot of the memory is wasted by all those algorithms although some seem to be better than others. Therefore, discovering the

most efficient algorithm among the four memory allocation techniques has to do with the factor that has been tried to optimize (memory usage) which includes; the size of the memory and the percentage at which the memory is used. Table 6 was presented to show the various percentages employed for the simulation of the four memory techniques under consideration with their respective memory wastage in KB. Therefore, discovering the most efficient algorithm among the four memory allocation techniques has to do with the factor that has been tried to optimize (memory usage) which includes; the size of the memory and the percentage at which the memory is used.

Table 6. Result of the Simulation of the Physical Memory that was Modeled

FREEMEM	10%	20%	30%	40%	50%
FIRST FIT	115	160	215	225	286
NEXT FIT	115	160	220	251	286
BEST FIT	115	160	210	235	280
WORSTFIT	120	160	225	246	296

Table 6had revealed that at 10% free memory all the techniques wasted 115 KB memory space while worst-fit wasted 120 KB memory space. At 20%, the four of them wasted 160 KB memory space, at 30%, best-fit wasted the smallest memory (210KB memory), next-fit wasted 220 KB memory, the worst-fit wasted 225 KB memory which is the highest and first-fit wasted 215 KB memory, at 40%, best-fit wasted 235KB memory, next-fit wasted 251KB memory, worst-fit wasted 246 KB memory and first-fit wasted 225 KB memory and at 50% best-fit wasted 280 KB memory, next-fit and first-fit wasted 286 KB memory each while worst-fit wasted the highest memory which is 296KB. The summation of the memory wasted within the range of 10% and 50% memory free among the four contiguous memory allocation techniques



Percentage of free memory modeled

Figure 1. Graph of the comparative result of four contiguous memory allocations

had revealed that worst-fit was the highest which is 1047KB while best-fit had the smallest memory wasted which is 1000KB. This implied that the worst-fit algorithm could not efficiently manage memory space in static variable memory partition. The obtained result from Table 6 is represented using both bar chart in Figure 1 and line graph in Figure 2 for clarity of the memory size that is been wasted by each of the algorithms. Figure 1 was used to represent memory wasted by first-fit, next-fit, best-fit and worst-fit based on the percentage range under consideration in which *X*-axis was used for percentage range and *Y*-axis was used for the amount of memory wasted. From 10% to 50%, it revealed that best-fit has the smallest memory wastage. Figure 2 was used to present the overall amount of memory wasted at a glance. The four contiguous memory allocation techniques (first-fit, next-fit, best-fit and worst-fit) under consideration were used as *X*-axis while the total amount of memory wasted by each technique was shown in *Y*-axis. This line graph in Figure 2 revealed equally that the best-fit wasted the lowest amount of memory. Thus, it should be a

technique that should be adopted for efficient memory management.

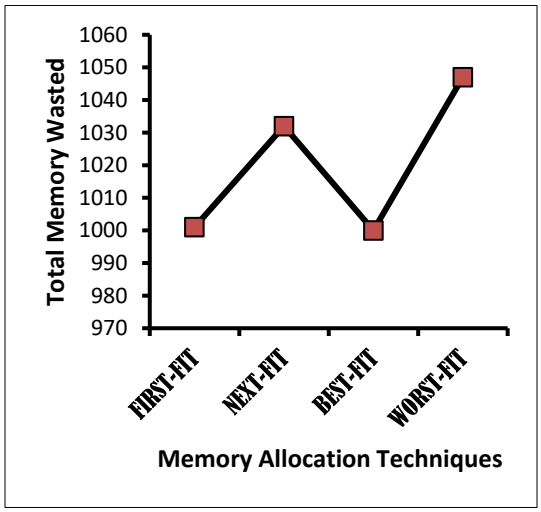


Figure 2. Graph of the summation of four contiguous memory allocation techniques

IV. Conclusion

The study had surveyed the memory allocation system under different free memory availability where each process/job was allocated with fixed memory space through modeling system. The total storage wasted by each memory allocation technique was observed and recorded. The system produced a summary of the memory table by checking which memory block or memory address each process was assigned and give an output of the location of the memory address where the process was assigned. It also performed the summation of all memory space that was not used when processes were still on the queue waiting for free space in the memory large enough to accommodate them and the wasted memory size.

References

[1] Dinesh, K., Mandeep, S. and Harpreet, K. "Memory Management in Operating System", *Journal of Emerging Technologies and Innovative Research (JETIR),* vol. 6, Issue 4, www.jetir.org (ISSN-2349-5125), 2019, pp. 465-471

[2] Nihad, R.O., Rezgar, H.S., Jihan, A.A., Shilan, B.M., Zainab, S.A. and Zryan, N.R. "Enhancing OS Memory Management Performance: A Review", *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, vol. 3, Issue 12, 2021, pp. 46-51.

[3] McGuire, T. "Memory Management Chapter 9: CS 431 -- Operating Systems by Dr. Tim McGuire", Sam Houston State University, 2016.

[4] Durgesh, R. "Memory Management in Operating System", *International Journal of Trend in Scientific Research and Development (ijtsrd)*, ISSN: 2456-6470, vol. 2, Issue 5, 2018, pp. 2346-2347, URL:https://www.ijtsrd.com/papers/ijtsrd18342.pdf

[5] Mohammad, O.F., Rahim, M.S.M., Zeebaree, S.R.M. and Ahmed, F.Y. "A Survey and Analysis of the Image Encryption Methods", *International Journal of Applied Engineering Research,* vol. 12, 2017, pp. 13265-13280.

[6] Pupykina, A. and Agosta, G. "Survey of Memory Management Techniques for HPC and Cloud Computing", *IEEE Access,* vol. 7, 2019, pp. 167351-167373.

[7] Ageed, Z., Mahmood, M.R., Sadeeq, M., Abdulrazzaq, M.B. and Dino, H. "Cloud Computing Resources Impacts on Heavy-load Parallel Processing Approaches", *IOSR Journal of Computer Engineering (IOSRJCE),* vol. 22, 2020, pp. 30-41.

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

[9] Sallow, A., Zeebaree, S., Zebari, R., Mahmood, M., Abdulrazzaq, M. and Sadeeq, M. "Vaccine Tracker SMS Reminder system: Design and Implementation", 2020.

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

[11] Zeebaree, S.R., Jacksi, K. and Zebari, R.R. "Impact Analysis of SYN Flood DDoS Attack on HAProxy and NLB Cluster-based Web Servers", *Indones. Journal. Of Electronic Engineering and Computer Science,* vol. 19, 2020, pp. 510-517.

[12] Zeebaree, S.R. and Yasin, H.M. "Arduino Based Remote Controlling for Home: Power Saving, Security and Protection", *International Journal of Scientific & Engineering Research,* vol. 5, 2014, pp. 266-272.

[13] Yasin, H.M., Zeebaree, S.R., Sadeeq, M.A., Ameen, S.Y., Ibrahim, I.M. and Zebari, R. "IoT and ICT Based Smart Water Management, Monitoring and Controlling System: A Review", *Asian Journal of Research in Computer Science,* 2021, pp. 42-56.

[14] Abdulqadir, H.R., Zeebaree, S.R., Shukur, H.M., Sadeeq, M.M., Salim, B.W. and Salih, A.A. "A Study of Moving from Cloud Computing to Fog Computing", *Qubahan Academic Journal,* vol. 1, 2021, pp. 60-70.

[15] Ahmed, F. "A Review of Memory Allocation and Management in Computer systems", *Computer Science & Engineering: An International Journal (CSEIJ)*, vol.6, no. 4. 2016.

[16] Maulud, D.H., Zeebaree, S.R., Jacksi, K., Sadeeq, M.A.M. and Sharif, K.H. "State of Art for Semantic Analysis of Natural Language Processing", *Qubahan Academic Journal,* vol. 1, 2021, pp. 21-28.

[17] Jack, R. "What is Static Memory Allocation and Dynamic Memory Allocation?" http://www.merithub.com/: [An Institute of Career Development], 2008.

[18] Lyna, G. "What is Memory Partitioning"? Definition & Concept, 2011. https://study.com/academy/lesson/what-is-memory-partitioning-definition-concept.html.

[19] Denning, P.J. and Graham, G.S. "Multi-Programmed Memory Management", *Proc. IEEE*, vol 63, 1975, pp. 924-939.

[20] MeritHub, B. "What is Static Memory Allocation and Dynamic Memory Allocation? 2011", http://www.merithub.com/: [An Institute of Career Development]. Retrieved. 2018-06-16. Best-Fit, First-Fit and Worst-Fit Memory Allocation Method for Fixed Partition. http://thumbsup2life.blogspot.com.ng/2011/02/best-fit-first-fit-and- worst- fit-memory.html

[21] Bays, C. "A Comparison of Next-fit, First-fit, and Best-fit". *Journal of Communication, ACM*, vol. 20, 1977, pp. 191-192. Do:10.1145/359436.359453, Available at: https://www.researchgate.net/publication/220427661_A_Comparison_of_Next-fit_First-fit_and_Best-fit [accessed Apr. 22, 2018].

http://thumbsup2life.blogspot.com.ng/2011/02/best-fit-first-fit-and-worst-fit-memory.html
https://study.com/academy/lesson/memory-allocation-schemes-definition-uses.html
https://www.tutorialspoint.com/operating_system/os_memory_management.htm
http://www.sciencehq.com/computing-technology/memory-management.html