Chapter Three
Memory Management

INTRODUCTION

Memory: RAM is often an area that offers significant performance improvements for most applications. RAM provides the working memory of your system; by having a great deal RAM, you avoid having to access the much slower disk arrays. More RAM is always better, but systems that run parallel queries and populate full-text indexes (and most are primarily DSS systems) require even more RAM.

If the more RAM, the better, how much can you get? Windows 2000 Server is a 32-bit operating system; as a consequence, it can address only 4GB of memory. Windows 2000 Advanced Server and Data Center are also 32-bit operating systems, but through the Address Windowing Extensions (AWE) API, it can address 8GB and 64GB, respectively. You need at least 64 MB of RAM for SQL Server, plus 32 MB of RAM for the operating system. The more RAM you have the more data that can be stored in the data cache and the more procedures that can be stored in the procedure cache. If the data and procedures that you are querying are in RAM, the server will not have to visit its disk system, providing a substantial improvement in performance.

Physical memory

Physical memory (also known as random-access memory (RAM)) is **a form of very fast, but volatile data storage**. When a system is low on physical memory, it often leads to system-wide delays or, in extreme cases, a complete hang of the system. RAM modules are typically measured in nanoseconds (1000^{-3}), and physical disks are typically measured in milliseconds (1000^{-1}). This makes physical memory roughly 100,000 times faster than a common physical disk. Therefore, when possible, Windows and Windows Server keep the most frequently accessed pages of memory in physical memory and rely on a disk only if needed.

When a system is low on physical memory, it often leads to system-wide delays or, in extreme cases, a complete hang of the system. This chapter covers how physical memory is managed by the operating system, how to identify low-memory conditions, and how to alleviate those conditions.

The CPU can directly access the main memory, Registers and cache of the system. The program always executes in main memory. The size of main memory affects degree of Multi programming to most of the extant. If the size of the main memory is larger than CPU can load more processes in the main memory at the same time and therefore will increase degree of Multi programming as well as CPU utilization.

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Overlays, swapping, and partitions

Fixed Partitioning

The earliest and one of the simplest technique which can be used to load more than one processes into the main memory is Fixed partitioning or Contiguous memory allocation.

In this technique, the main memory is divided into partitions of equal or different sizes. The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.

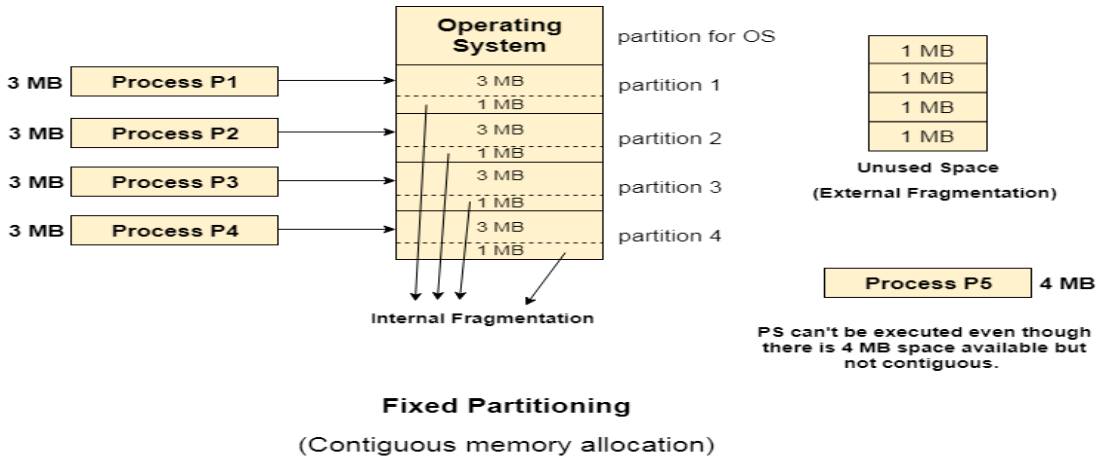
In fixed partitioning,

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

There are various cons of using this technique.

1. **Internal Fragmentation**

If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation. As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.



2. Limitation on the size of the process

If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

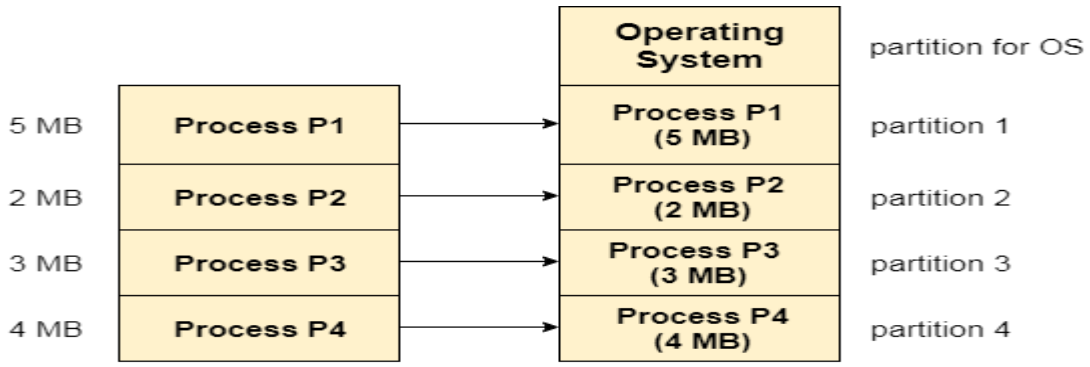
3. Degree of multiprogramming is less

By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.

Dynamic Partitioning

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.

The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.



Dynamic Partitioning

(Process Size = Partition Size)

Pros of Dynamic Partitioning over fixed partitioning

1. No Internal Fragmentation

Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

2. No Limitation on the size of the process

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

3. Degree of multiprogramming is dynamic

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

Cons of dynamic partitioning

External Fragmentation

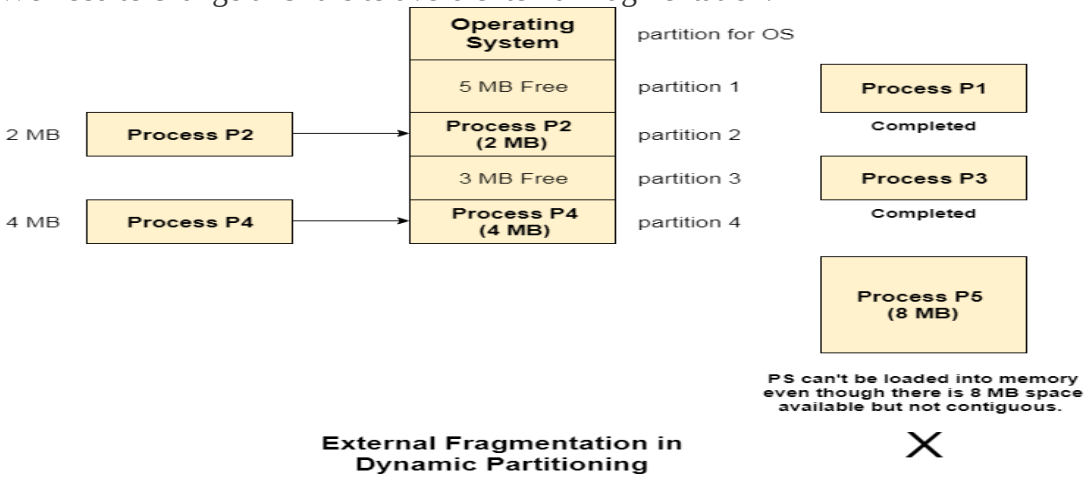
The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

Absence of internal fragmentation doesn't mean that there will not be external fragmentation.

Let's consider three processes P1 (1 MB) and P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory.

After some time P1 and P3 got completed and their assigned space is freed. Now there are two unused partitions (1 MB and 1 MB) available in the main memory but they cannot be used to load a 2 MB process in the memory since they are not contiguously located.

The rule says that the process must be contiguously present in the main memory to get executed. We need to change this rule to avoid external fragmentation.



External Fragmentation in Dynamic Partitioning

Complex Memory Allocation

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and de-allocation is very complex since the partition size will be varied every time when it is assigned to a new process. OS has to keep track of all the partitions.

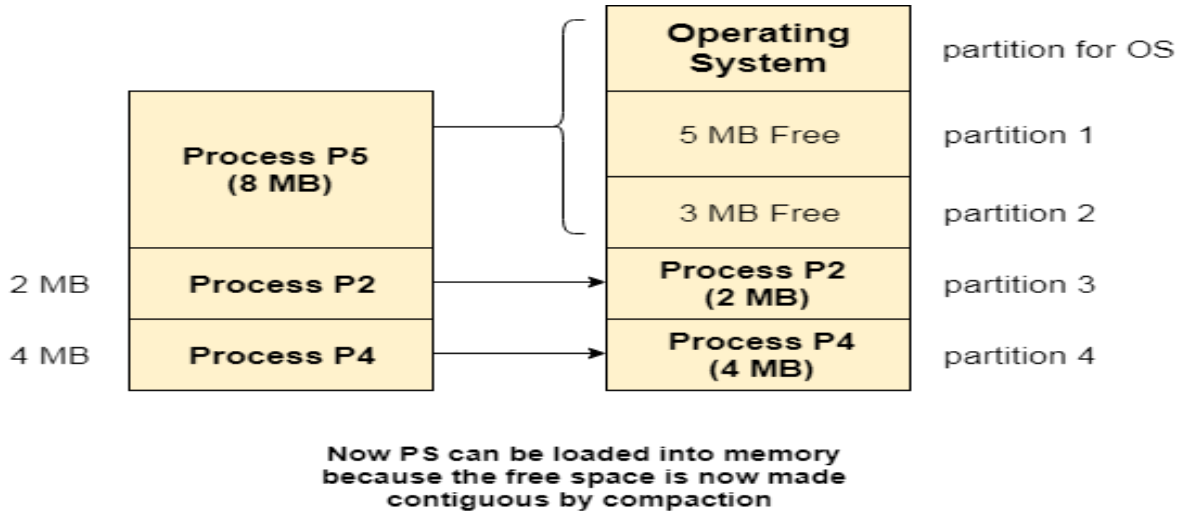
Due to the fact that the allocation and de-allocation are done very frequently in dynamic memory allocation and the partition size will be changed at each time, it is going to be very difficult for OS to manage everything.

Compaction

We got to know that the dynamic partitioning suffers from external fragmentation. However, this can cause some serious problems.

We can also use compaction to minimize the probability of external fragmentation. In compaction, all the free partitions are made contiguous and all the loaded partitions are brought together.

By applying this technique, we can store the bigger processes in the memory. The free partitions are merged which can now be allocated according to the needs of new processes. This technique is also called defragmentation.



Compaction

Problem with Compaction

The efficiency of the system is decreased in the case of compaction due to the fact that all the free spaces will be transferred from several places to a single place.

Huge amount of time is invested for this procedure and the CPU will remain idle for all this time.

Despite of the fact that the compaction avoids external fragmentation, it makes system inefficient.

Overlays

The main problem in fixed partitioning is the size of a process has to be limited by the maximum size of the partition, which means a process can never be span over another. In order to solve this problem, earlier people have used some solution which is called as Overlays. The concept of overlays is that whenever a process is running it will not use the complete program at the same time, it will use only some part of it. Then overlays concept says that whatever part you required, you load it a once the part is done, then you just unload it, means just pull it back and get the new part you required and run it.

Formally, “The process of transferring a block of program code or other data into internal memory, replacing what is already stored”. Sometimes it happens that compare to the size of the biggest partition, the size of the program will be even more, then, in that case, you should go with overlays. So, overlay is a technique to run a program that is bigger than the size of the physical

memory by keeping only those instructions and data that are needed at any given time. Divide the program into modules in such a way that not all modules need to be in the memory at the same time.

Advantage

- Reduce memory requirement
- Reduce time requirement

Disadvantage

- Overlap map must be specified by programmer
- Programmer must know memory requirement
- Overlapped module must be completely disjoint
- Programming design of overlays structure is complex and not possible in all cases

Example – The best example of overlays is assembler. Consider the assembler has 2 passes, 2 pass means at any time it will be doing only one thing, either the 1st pass or the 2nd pass. This means it will finish 1st pass first and then 2nd pass. Let assume that available main memory size is 150KB and total code size is 200KB

Pass 1……..70KB
Pass 2……..80KB
Symbol table……..30KB
Common routine……20KB

As the total code size is 200KB and main memory size is 150KB, it is not possible to use 2 passes together. So, in this case, we should go with the overlay's technique. According to the overlays concept at any time only one pass will be used and both the passes always need symbol table and common routine. Now the question is if overlays-driver is 10KB, then what is the minimum partition size required? For pass 1 total memory needed is = (70KB + 30KB + 20KB + 10KB) = 130KB and for pass 2 total memory needed is = (80KB + 30KB + 20KB + 10KB) = 140KB.So if we have minimum 140KB size partition then we can run this code very easily.

Overlays driver: -It is the user responsibility to take care of overlaying, the operating system will not provide anything. Which means the user should write even what part is required in the 1st pass and once the 1st pass is over, the user should write the code to pull out the pass 1 and load the pass 2. That is what is the responsibility of the user, that is known as the Overlays driver. Overlays driver will just help us to move out and move in the various part of the code.

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store, and then brought back into memory for continued execution.

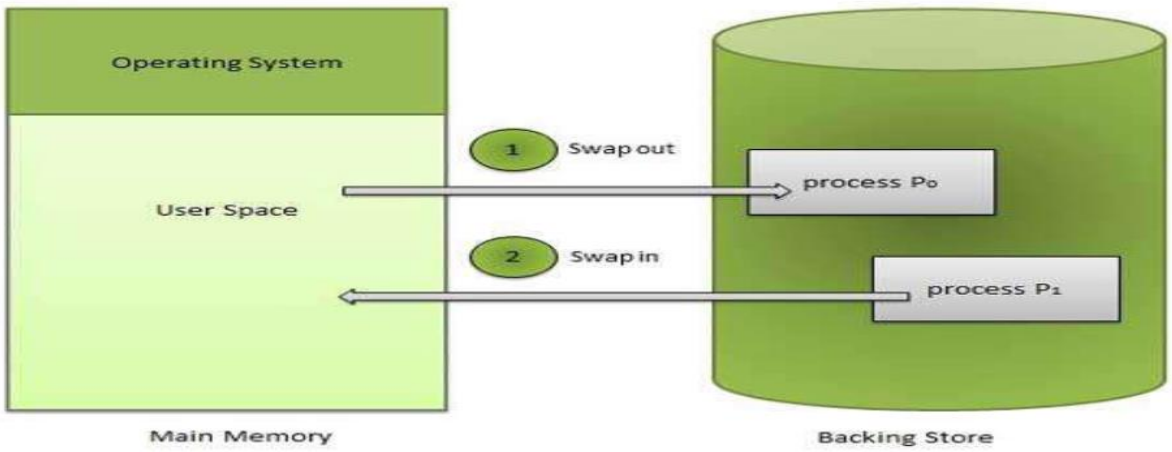
Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Major time-consuming part of swapping is transfer time. Total transfer time is directly proportional to the amount of memory swapped. Let us assume that the user process is of size 100KB and the backing store is a standard hard disk with transfer rate of 1 MB per second. The actual transfer of the 100K process to or from memory will take.

100KB / 1000KB per second

= 1/10 second

= 100 milliseconds



Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective. The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.

Paging

Paging is a method or a technique which is used for non-contiguous memory allocation. It is a fixed size partitioning theme (scheme). In paging, both main memory and secondary memory are divided into equal fixed size partitions. The partitions of secondary storage and main memory area unit known as pages and frames respectively.

Page Table in OS

Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.

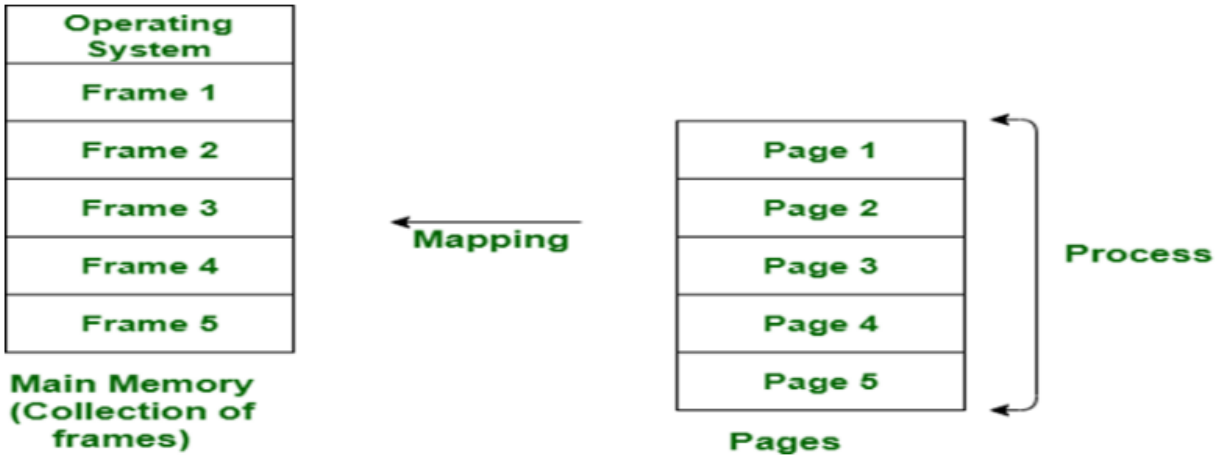
Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystems.

The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical address only.

In this situation, a unit named as Memory Management Unit comes into the picture. It converts the page number of the logical address to the frame number of the physical address. The offset remains same in both the addresses.

To perform this task, Memory Management unit needs a special kind of mapping which is done by page table. The page table stores all the Frame numbers corresponding to the page numbers of the

page table. In other words, the page table maps the page number to its actual location (frame number) in the memory.



Virtual Memory

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory. By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

How it works?

In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced.

The Demand Paging is a condition which is occurred in the Virtual Memory. We know that the pages of the process are stored in secondary memory. The page is brought to the main memory

when required. We do not know when this requirement is going to occur. So, the pages are brought to the main memory when required by the Page Replacement Algorithms.

Page Replacement Algorithms

1. **First In First Out (FIFO)**: The page to be replaced is the "oldest" page in the memory, the one which was loaded before all the others
2. **Least Recently Used (LRU):** The page to be replaced is the one which has not been referenced since all the others have been referenced
3. **Last In First Out (LIFO)**: The page to be replaced is the one most recently loaded into the memory.
4. **Least Frequently Used (LFU)**: The page to be replaced is the one used least often of the pages currently in the memory
5. **Optimal**: Replace the Page which is not used in the Longest Dimension of time in future. This principle means that after all the frames are filled then, see the future pages which are to occupy the frames. Go on checking for the pages which are already available in the frames. Choose the page which is at last.

Segmentation

In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.
The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.
Segment table contains mainly two information about segment:

1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to the Operating system rather than the User. It divides all the processes into the form of pages regardless of the fact that a process can have some relative parts of functions which need to be loaded in the same page.
Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.
It is better to have segmentation which divides the process into the segments. Each segment contains the same type of functions such as the main function can be included in one segment and the library functions can be included in the other segment.
CPU generates a logical address which contains two parts:

1. Segment Number
2. Offset

With the help of segment map tables and hardware assistance, the operating system can easily translate a logical address into physical address on execution of a program.

The **Segment number** is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.
In the case of valid addresses, the base address of the segment is added to the offset to get the physical address of the actual word in the main memory.

Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compared to the page table in paging.

Disadvantages

1. It can have external fragmentation.
2. It is difficult to allocate contiguous memory to variable sized partition.
3. Costly memory management algorithms.

Difference between Paging and Segmentation

No.	Paging	Segmentation
1	Non-Contiguous memory allocation	Non-contiguous memory allocation
2	Paging divides program into fixed size pages.	Segmentation divides program into variable size segments.
3	OS is responsible	Compiler is responsible.
4	Paging is faster than segmentation	Segmentation is slower than paging
5	Paging is closer to Operating System	Segmentation is closer to User
6	It suffers from internal fragmentation	It suffers from external fragmentation
7	There is no external fragmentation	There is no external fragmentation
8	Logical address is divided into page number and page offset	Logical address is divided into segment number and segment offset
9	Page table is used to maintain the page information.	Segment Table maintains the segment information
10	Page table entry has the frame number and some flag bits to represent details about pages.	Segment table entry has the base address of the segment and some protection bits for the segments.