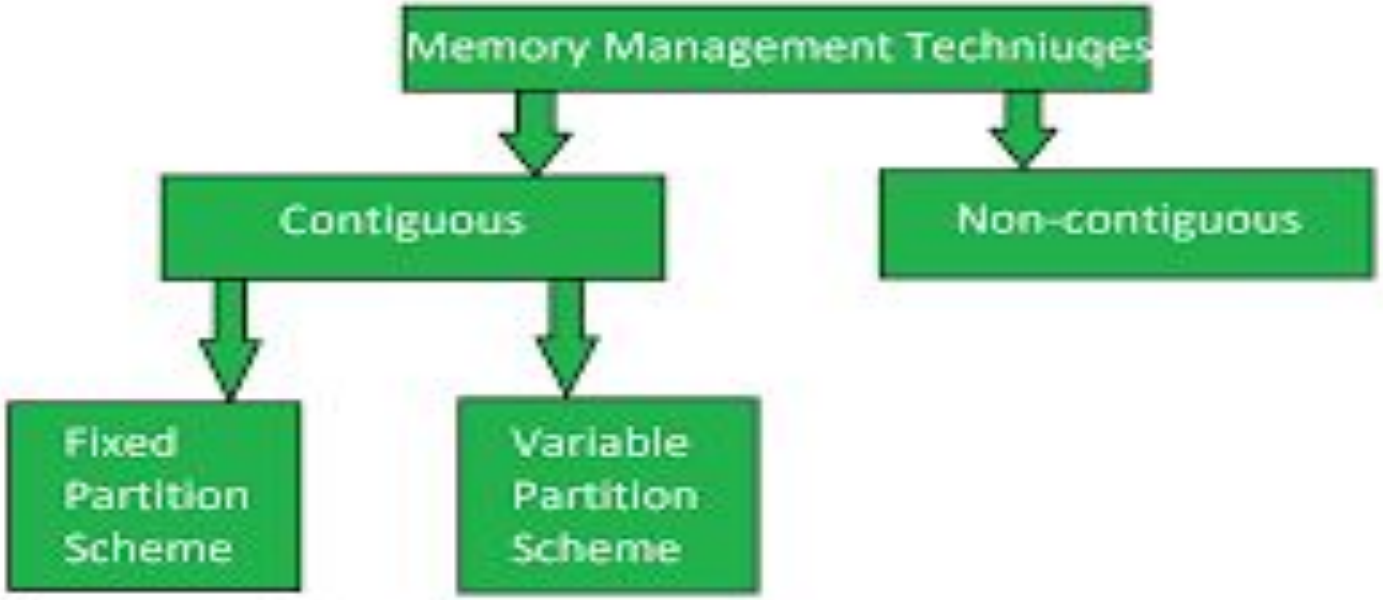
Memory Management
Lecture 2

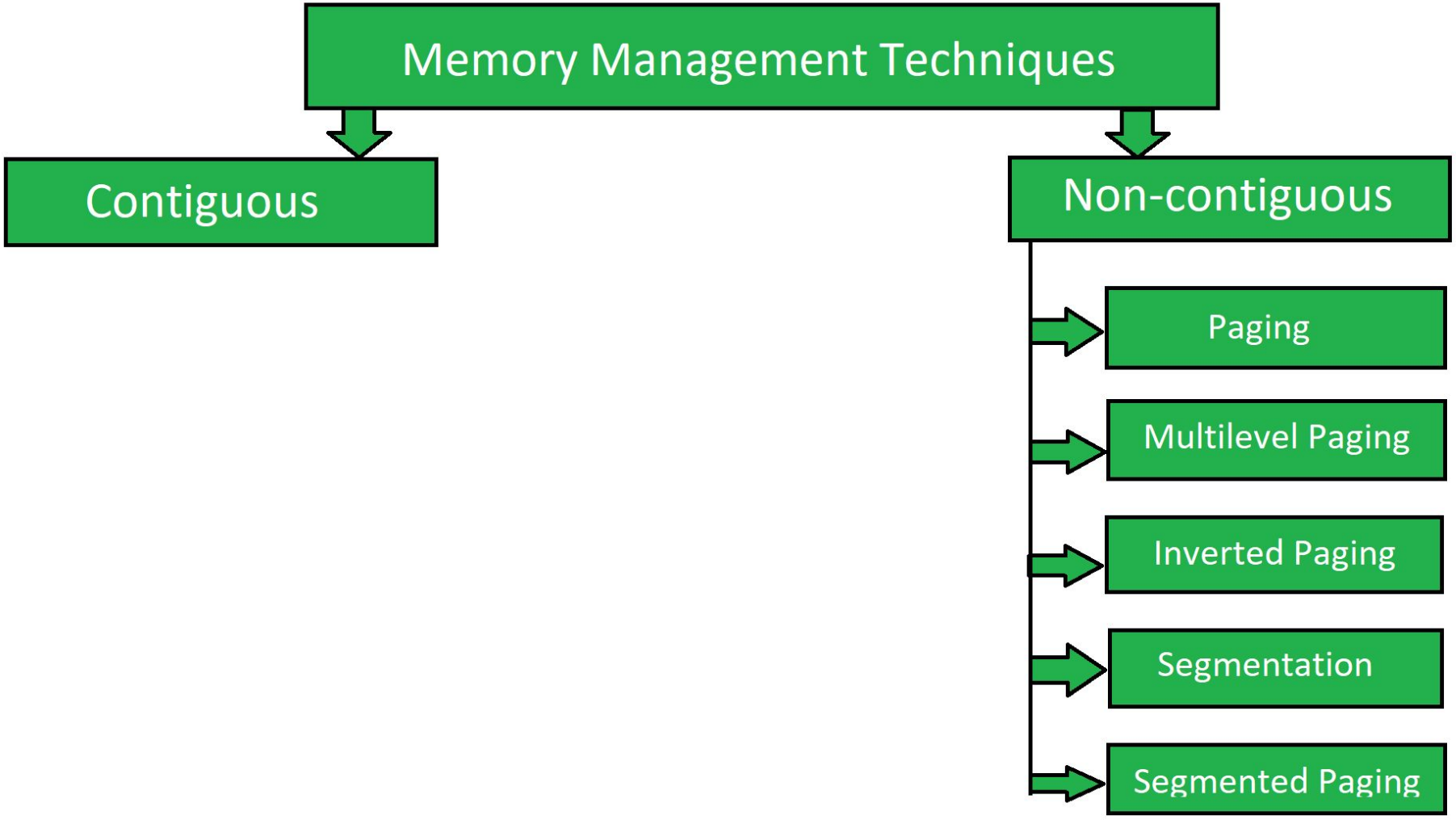
Memory Partitioning

Minakshi Retharekar

OPERATING SYSTEMS: INTERNALS AND DESIGN PRINCIPLES
William Stalling 6th edition

Memory management Techniques




Memory Management Techniques
Contiguous
Non-contiguous
Paging
Multilevel Paging
Inverted Paging
Segmentation
Segmented Paging

Degree of Multiprogramming

- An operating system can support multiple processes in memory.
- While one processes receives service from CPU, another process receives services from I/O device and other processes are waiting in Queue.
- The degree of multiprogramming describes the maximum number of processes that a single-processor system can accommodate efficiently.
- The primary factor affecting the degree of multiprogramming is the amount of memory available to be allocated to executing processes.
- if n processes are in memory, the probability that all processes are waiting for I/O is P^n
- CPU Utilization is = 1-P^n

Contiguous Memory Allocation

- Contiguous memory allocation is a memory management technique used by operating systems to allocate a block of contiguous memory to a process.
- When a process requests memory, a single contiguous section of memory blocks is allotted depending upon its requirements.
- The allocation of contiguous memory to a process involves dividing the available memory into fixed-sized partitions or segments.

Non-Contiguous Memory Allocation

- Non-contiguous allocation involves the use of pointers to link the non-contiguous memory blocks allocated to a process.
- It allows a process to obtain multiple memory blocks in various locations in memory based on its requirements
- These pointers are used by the operating system to keep track of the memory blocks allocated to the process and to locate them during the execution of the process

MEMORY PARTITIONING

- ✔ The principal operation of memory management is to bring processes into main memory for execution by the processor.
- ✔ Partitioning types
  - ✔ Fixed Partitioning
  - ✔ Dynamic Partitioning

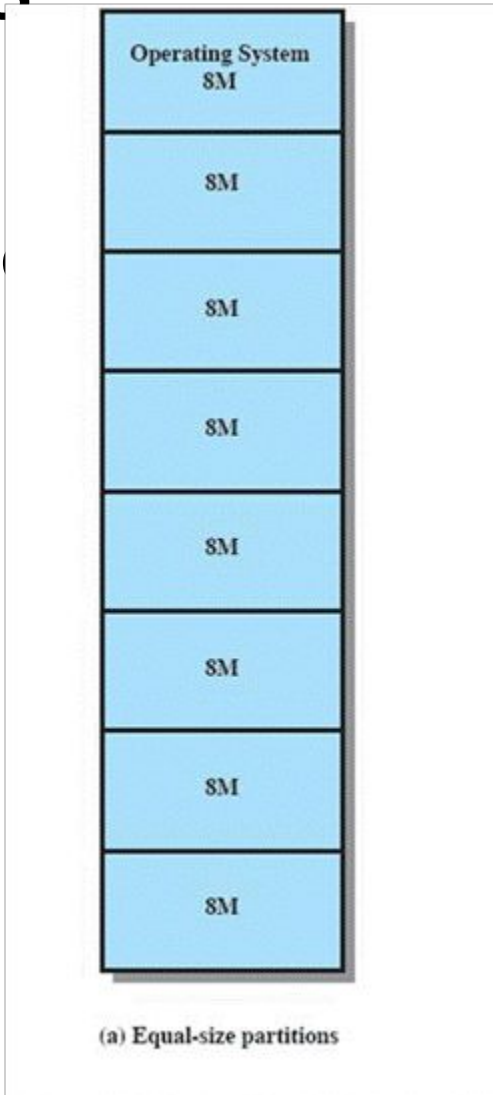
Fixed Partitioning

- ✔ we can assume that the operating system occupies some fixed portion of main memory and that the rest of main memory is available for use by multiple processes.
- ✔ The simplest scheme for managing this available memory is to partition it into regions with **fixed boundaries.**
  - ✔ Divide memory into several fixed size partitions at system boot time.
  - ✔ Each partition may contain exactly one process.

Fixed Partitioning

Equal-size partitions

- ✔ Any process whose size is less than or equal to the partition size can be loaded into an available partition



(a) Equal-size partitions

Advantages of Fixed Size Partition

- It is simple and easy to implement
- It is predictable, means operating system can ensure minimum amount of memory for each process.
- It can prevent processes from interfering with each others memory space, improving the security and stability of system.

Disadvantages of Fixed Size Partition

- Internal Fragmentation, means memory in a partition remains unused.
- Limits in process size
- Limitation on degree of multiprogramming
- External fragmentation- total space that is unused in multiple partitions can't be utilized for loading the process even if there’s some space available because it is not in contiguous form.

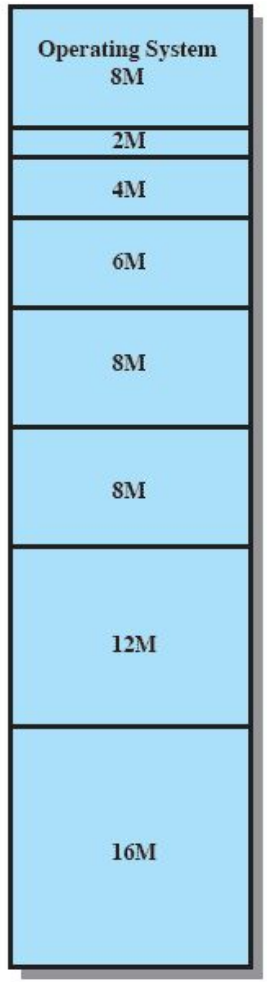
Fixed Partitioning Problems

- ✔ A program may not fit in a partition.
  - The programmer must design the program with overlays
- ✔ Main memory use is inefficient.
  - Any program, no matter how small, occupies an entire partition.
  - This is results in ***internal fragmentation.***

Solution – Unequal Size Partitions

Unequal Size Partitions

- ✔ but doesn't solve completely
  - Programs up to 16M can be accommodate without overlay
  - Smaller programs can be placed in smaller partitions, reducing internal fragmentation

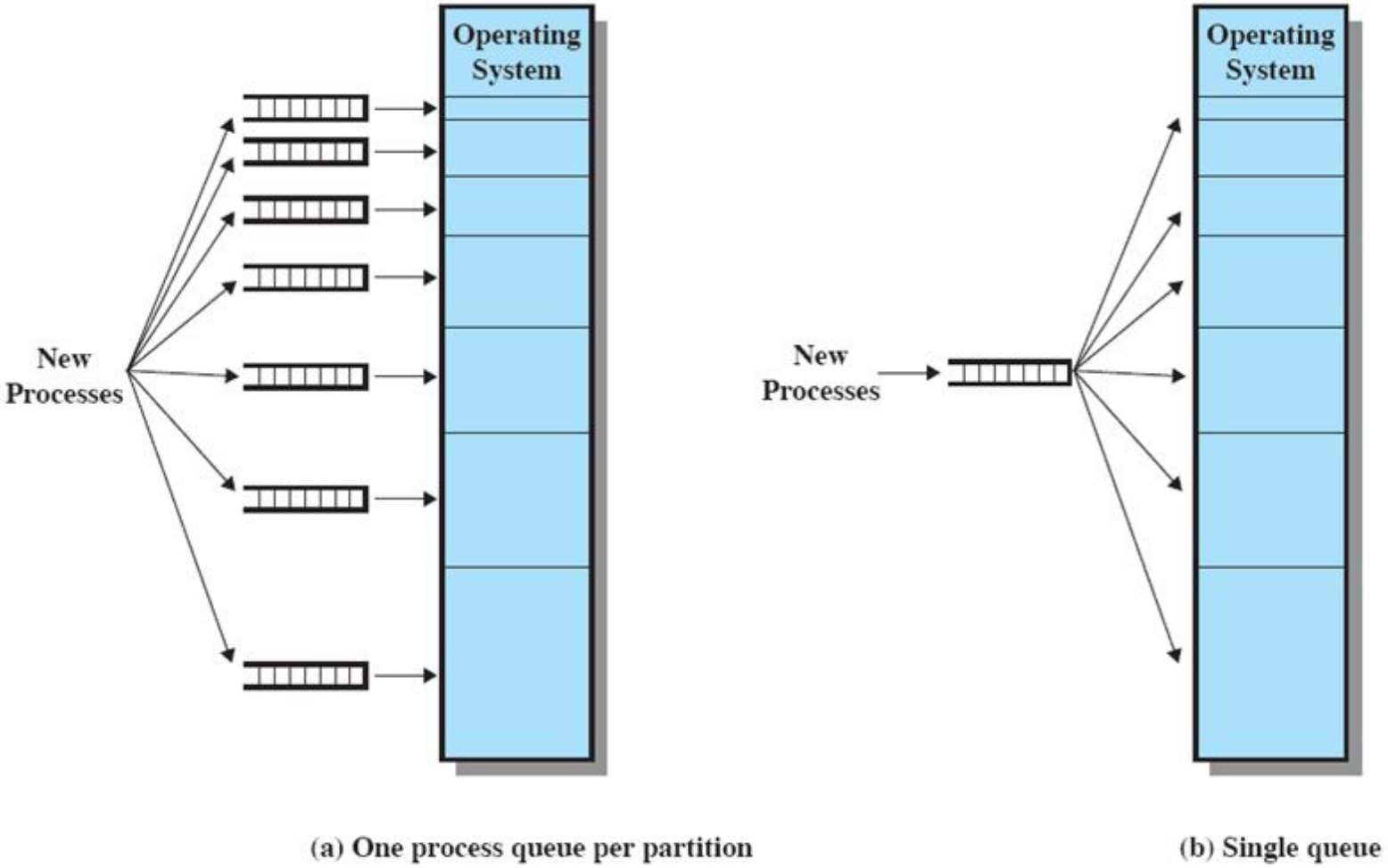


(b) Unequal-size partitions

Placement Algorithm

- ✔ Equal-size
  - Placement is trivial (no options)
- ✔ Unequal-size
  - Can assign each process to the smallest partition within which it will fit
  - Queue for each partition
  - Processes are assigned in such a way as to minimize wasted memory within a partition

Fixed Partitioning



(a) One process queue per partition

(b) Single queue

Memory Assignment for Fixed Partitioning

Remaining Problems with Fixed Partitions

- ✔ The number of active processes is limited by the system
- ✔ A large number of very small process will not use the space efficiently
  - In either fixed or variable length partition methods

Variable (Dynamic) Partitioning

- ✔ Partitions are of variable length and number
- ✔ Process is allocated exactly as much memory as required
- ✔ Partitions are not made before execution or during system configuration
- ✔ Initially RAM is empty and partitions are made during the run time according to process's need.
- ✔ The size of partition will be equal to incoming process

Advantages of Variable Size Partition

- No internal fragmentation
- No restriction on degree of multiprogramming
- no limitation on process size
-

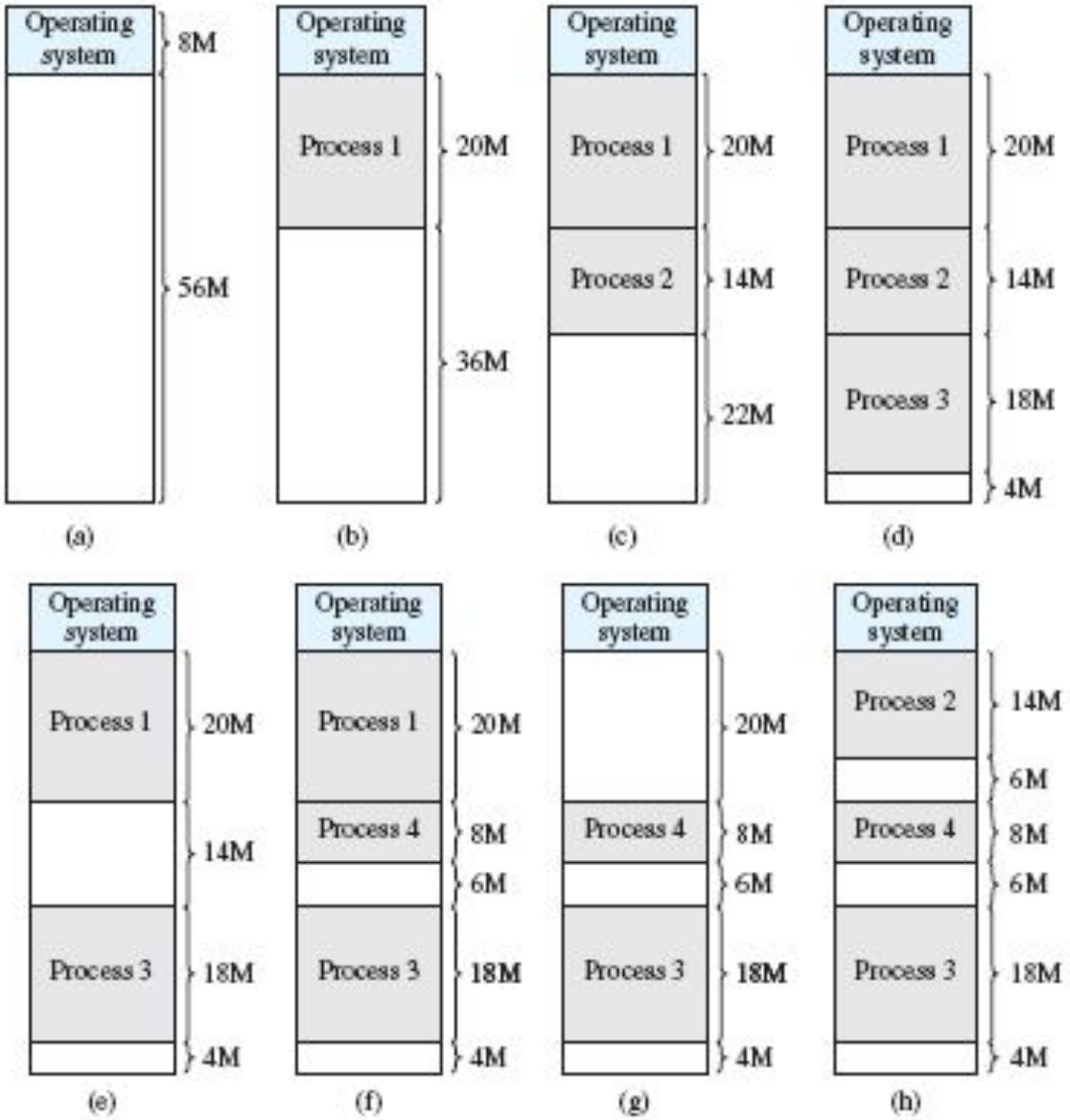
Disadvantages of Variable Size Partition

- Difficult to implement due to run time allocation
- External Fragmentation
- Example
-
-

Dynamic Partitioning Example

- ✔ ***External Fragmentation***
- ✔ Memory external to all processes is fragmented
- ✔ Can resolve using ***compaction***
  - OS moves processes so that they are contiguous
  - Time consuming and wastes CPU time

External Fragmentation



Dynamic Partitioning

- ✔ Operating system must decide which free block to allocate to a process
- ✔ Algorithms:
- ✔ **Best-fit algorithm**
  - Chooses the block that is closest in size to the request
  - Worst performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left
  - Memory compaction must be done more often

Dynamic Partitioning

First-fit algorithm

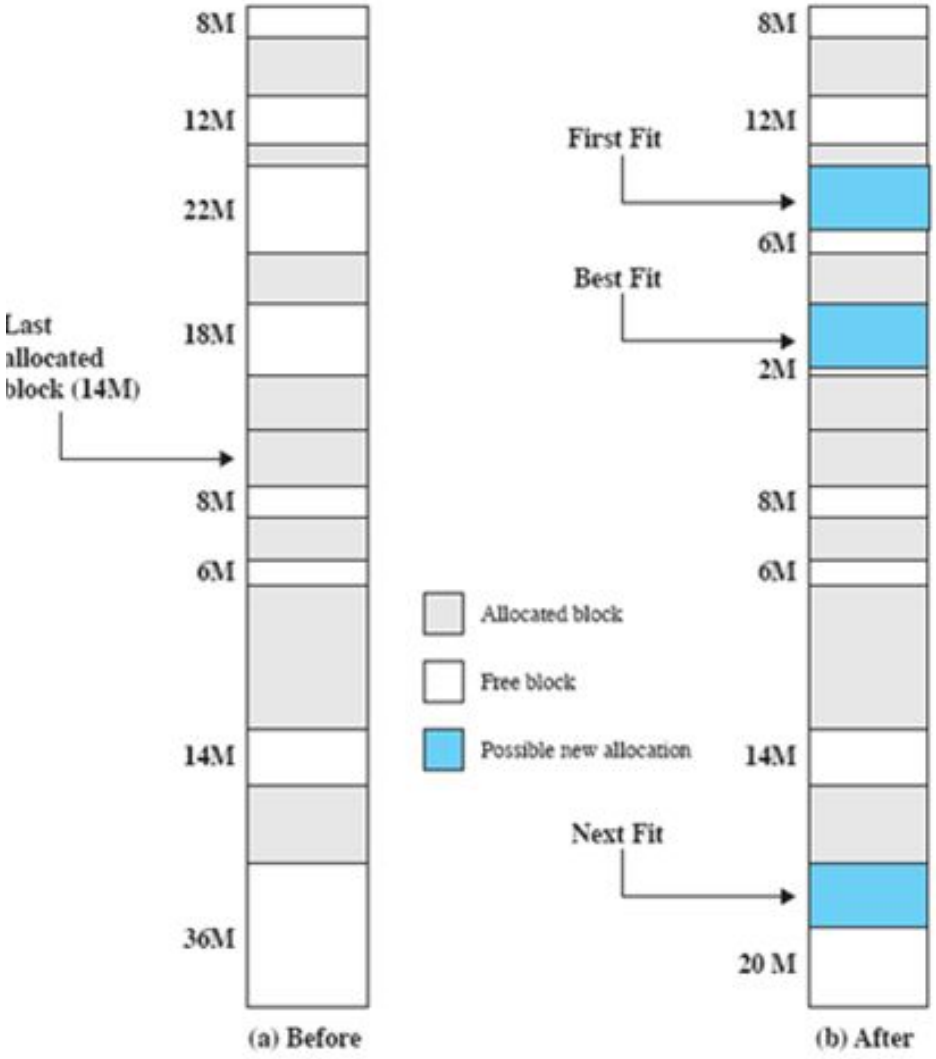
- Scans memory form the beginning and chooses the first available block that is large enough
- Fastest
- May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Dynamic Partitioning

Next-fit

- Scans memory from the location of the last placement
- More often allocate a block of memory at the end of memory where the largest block is found
- The largest block of memory is broken up into smaller blocks
- Compaction is required to obtain a large block at the end of memory

Allocation


8M
12M
22M
18M
8M
6M
14M
36M
Last allocated block (14M)
(a) Before
First Fit
Best Fit
Next Fit
6M
2M
20 M
Allocated block
Free block
Possible new allocation
(b) After

THANK YOU