OPERATING SYSTEMS:

Memory Management Concepts and Techniques

August 12, 2025

SINDHULI COMMUNITY TECHNICAL INSTITUTE

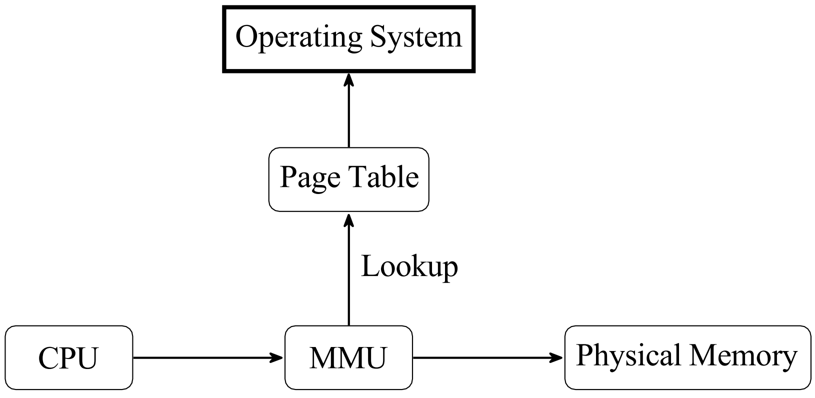
Submitted by: Susmita Karki

1. Addresses

Memory management in operating systems ensures processes run efficiently by managing how memory is allocated and accessed. A key aspect is the distinction between logical and physical **addresses**.

- **Logical Addresses**: These are virtual addresses generated by a process during execution. They exist in a process's virtual address space, which is an abstraction that allows each process to operate as if it has exclusive access to memory. Logical addresses are independent of the physical memory layout, providing portability and protection.
- **Physical Addresses**: These correspond to actual locations in the computer's RAM where data is stored. The operating system maps logical addresses to physical addresses to ensure correct data access.

The **Memory Management Unit (MMU)** facilitates this translation. It uses a page table to map logical addresses (consisting of a page number and offset) to physical addresses. For example, in a paging system, a logical address's page number is looked up in the page table to find the corresponding physical frame, and the offset is added to locate the exact memory location.



Address Translation

Figure 1: Logical to Physical Address Translation by MMU

This mechanism ensures process isolation and efficient memory use, critical for multitasking systems.

2. Swapping Mechanism

Swapping is a memory management technique that temporarily moves a process's memory pages from RAM to secondary storage (e.g., HDD or SSD) when RAM is full, and restores them when needed. This allows the system to support more processes than physical memory can accommodate.

2.1 Swapping Process

The swapping process involves the following steps:

1. **Detect Memory Pressure**: The OS identifies when RAM is insufficient for active processes.
2. **Select Process**: A process (often inactive or low-priority) is chosen for swapping out.
3. **Save State**: The process's state (registers, program counter) is saved.
4. **Write to Disk**: The process's memory pages are written to a swap space on secondary storage.
5. **Free RAM**: The RAM occupied by the process is released.
6. **Swap In**: When the process is needed again, its pages are read from disk and loaded into RAM.
7. **Restore State**: The process's state is restored, and execution resumes.

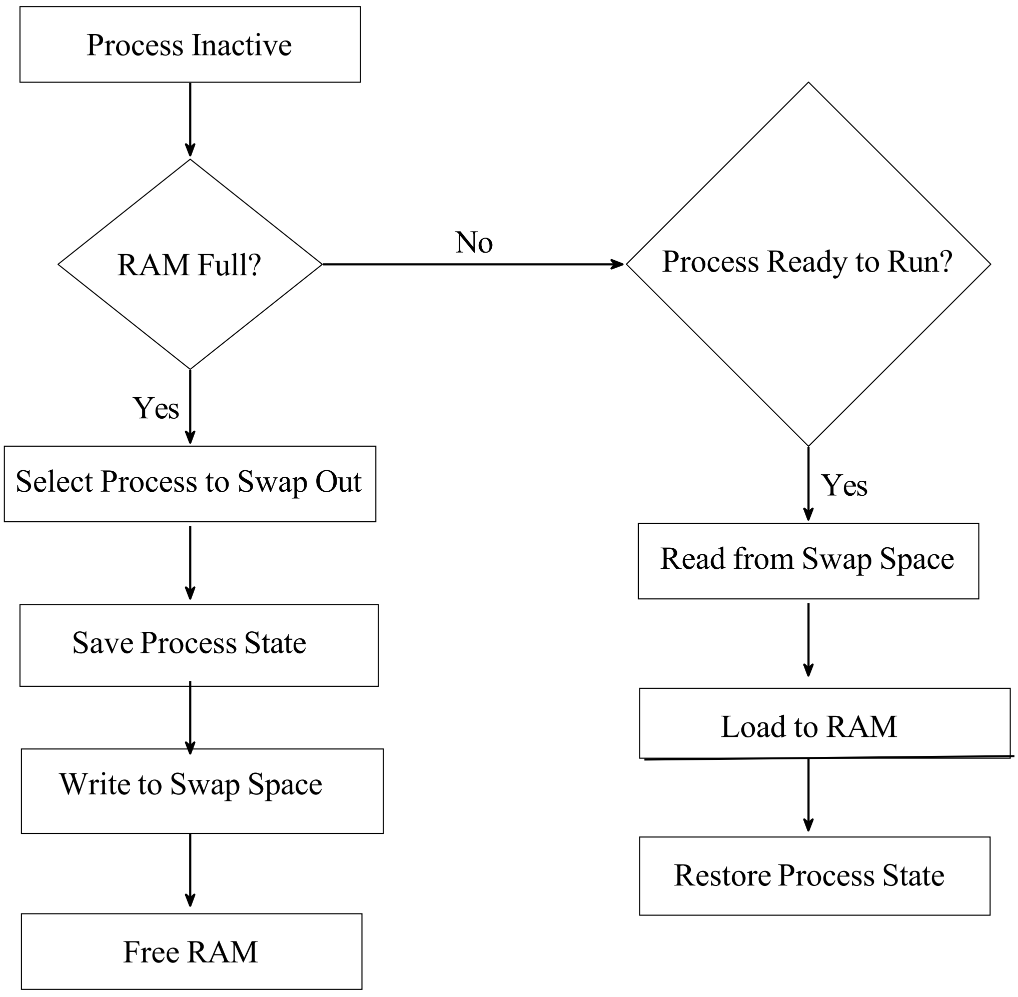


Figure 2: Flowchart of Swapping Mechanism

2.2 Performance Trade-offs

Swapping introduces overheads:

- **Context Switch Overhead**: Saving and restoring process states requires CPU cycles,
- impacting performance.
- **I/O Latency**: Disk access is slower than RAM. HDDs have access times of 5–10ms, while SSDs are faster (0.1–0.5ms).

In Nepal, hardware costs influence swapping performance. A 1TB HDD costs NPR 5,000–10,000, offering larger swap space but slower access. A 512GB SSD costs NPR 12,000–20,000, providing faster swaps but less capacity. For budget-constrained institutions, HDDs are common, but SSDs are increasingly adopted for performance-critical systems like virtual labs.

Swapping introduces several performance trade-offs. First, there is context switch overhead: swapping requires saving and restoring the process state, which involves CPU time for updating data structures like the Process Control Block (PCB). This can add milliseconds to the switch time, impacting system responsiveness.

Second, I/O latency is a major factor. Swapping to secondary storage involves disk operations, which are significantly slower than RAM access. For Hard Disk Drives (HDDs), seek times can be 5-10 ms, with transfer rates around 100-200 MB/s. Solid State Drives (SSDs) reduce this to sub-millisecond seek times and 500+ MB/s transfers, but at a higher cost.

In the Nepali context, cost considerations are crucial, especially for educational institutions or small businesses. As of 2025, HDD prices in Nepal range from Rs. 3,000 for 500GB to Rs. 32,000 for larger capacities (e.g., 1TB internal HDD around Rs. 5,000). SSDs are more expensive, with 256GB models starting at Rs. 3,900 and 1TB at Rs. 12,000-21,000. For resource-constrained setups like those in rural Nepal or public universities, HDDs offer better cost-per-GB (around Rs. 5/GB vs. Rs. 12-21/GB for SSDs), but they increase swapping latency, leading to slower system performance. SSDs, while reducing I/O bottlenecks, may not be feasible for large-scale swapping due to budget limits, pushing systems towards hybrid approaches (SSD for OS, HDD for swap space).

3. Contiguous Allocation Techniques

Contiguous memory allocation requires that each process occupies a single, continuous block of memory in RAM. This simplifies address translation, as the operating system only needs to manage a base address and a limit for each process. However, it introduces challenges like fragmentation. Below, we explore the two primary contiguous allocation methods—fixed-partitioning and dynamic-partitioning (including first-fit, best-fit, and worst-fit algorithms)—with explanations and memory layout diagrams for each.

3.1 Fixed Partitioning

Memory is divided into fixed-size partitions, each accommodating one process. This is simple but inflexible, as processes must fit within partition sizes, leading to **internal fragmentation** (unused space within a partition).

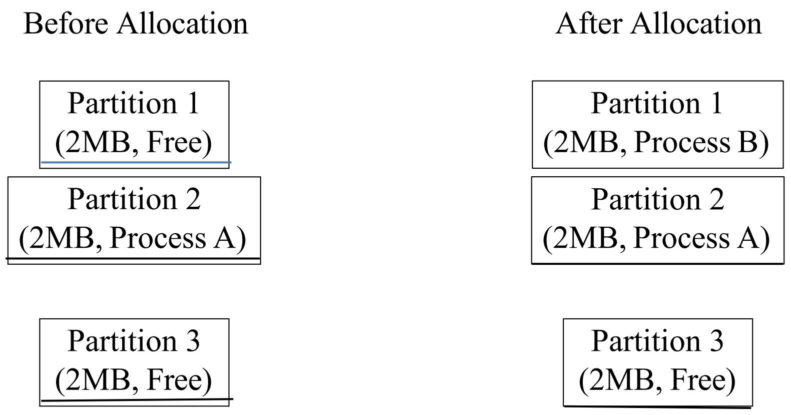


Figure 3: Fixed Partitioning: Before and After Allocation

3.2 Dynamic Partitioning

Memory is allocated dynamically based on process size, reducing internal fragmentation but introducing **external fragmentation** (scattered free memory). Allocation strategies include:

- **First-fit**: Allocates the first block large enough for the process.
- **Best-fit**: Allocates the smallest block that fits, minimizing waste.
- **Worst-fit**: Allocates the largest block, aiming to leave larger free blocks.

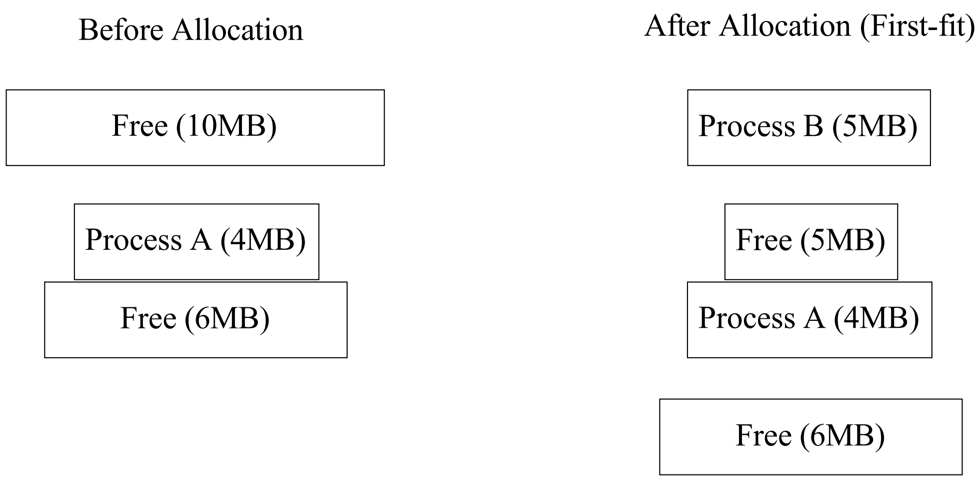


Figure 4: Dynamic Partitioning: First-fit Allocation

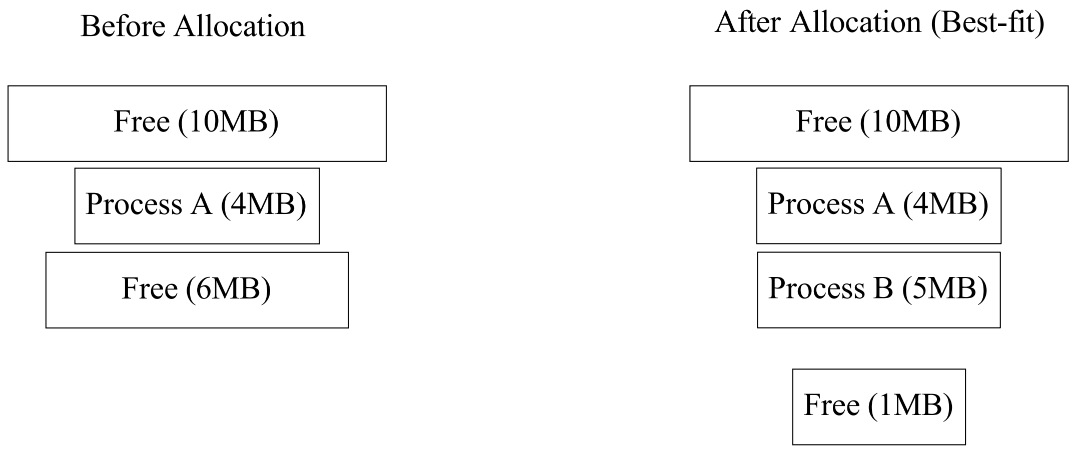


Figure 5: Dynamic Partitioning: Best-fit Allocation

Each strategy has trade-offs. First-fit is fast but may fragment memory unevenly. Best-fit minimizes waste but requires searching the entire memory list. Worst-fit leaves larger free blocks but may not optimize space usage.

4. Analysis & Example

Scenario: Virtual Labs at Tribhuvan University

Tribhuvan University (TU) runs virtual labs for computer science students at its Institute of Engineering, using a 16GB RAM server to load Virtual Machines (VMs): VM1 (OS, 4GB), VM2 (DB, 3GB), VM3 (AI, 6GB), VM4 (Web, 2GB). Total demand (15GB) nearly fills memory, requiring efficient allocation and swapping to maximize utilization (>80%) in a cost-conscious Nepali context.

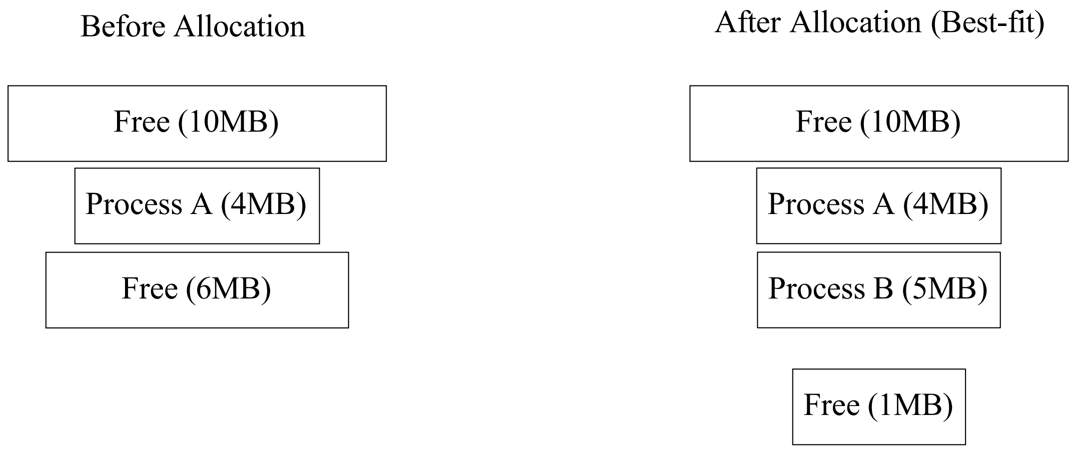


Figure 5: Dynamic Partitioning: Best-fit Allocation

Each strategy has trade-offs. First-fit is fast but may fragment memory unevenly. Best-fit minimizes waste but requires searching the entire memory list. Worst-fit leaves larger free blocks but may not optimize space usage.

5. Analysis & Example

Scenario: Virtual Labs at Tribhuvan University

Tribhuvan University (TU) runs virtual labs for computer science students at its Institute of Engineering, using a 16GB RAM server to load Virtual Machines (VMs): VM1 (OS, 4GB), VM2 (DB, 3GB), VM3 (AI, 6GB), VM4 (Web, 2GB). Total demand (15GB) nearly fills memory, requiring efficient allocation and swapping to maximize utilization (>80%) in a cost-conscious Nepali context.

Allocation and Swapping Process (Dynamic Partitioning, Best-Fit)

1. **Initial Allocation**:
   - Allocate VM1 (4GB), VM2 (3GB), VM3 (6GB) in 16GB memory.
   - Layout: [VM1:4GB] [VM2:3GB] [VM3:6GB] [Free:3GB] (Utilization: 81.25%).
2. **Swapping Out**:
   - To load VM4 (2GB), swap out VM2 (3GB, least recently used) to SSD (Rs. 3,000 for 256GB, fast ~0.1ms seek time vs. HDD's 5-10ms at Rs. 3,000 for 500GB).
   - Layout: [VM1:4GB] [Free:3GB] [VM3:6GBz] [Free:3GB] (Utilization: 62.5%).
3. **Allocate VM4 and Compact**:
   - Place VM4 in a 3GB hole; compact if fragmented.
   - Layout: [VM1:4GB] [VM3:6GB] [VM4:2GB] [Free:4GB] (Utilization: 75%).
4. **Swapping Back**:
   - Swap VM2 back into 4GB hole when needed.
   - Final Layout: [VM1:4GB] [VM2:3GB] [VM3:6GB] [VM4:2GB] [Free:1GB] (Utilization: 93.75%).

Analysis

- **Benefits**: Best-fit minimizes fragmentation; swapping to SSD ensures high utilization (93.75%) for TU's 20–30 student labs, saving costs vs. dedicated servers.
- **Trade-offs**: Swapping incurs I/O latency (SSD faster but pricier; HDD at Rs. 5/GB suits budget). Context switches add ~10–50ms delay.
- **Outcome**: Balances cost (HDD/SSD hybrid) and performance, ideal for TU's virtual

labs with intermittent VM usage.

6. Discussion

Contiguous allocation offers simplicity and low overhead, as processes are loaded into single memory blocks, reducing management complexity. However, it has significant limitations:

- **Internal Fragmentation** (Fixed Partitioning): Memory within partitions may go unused if processes are smaller than the partition size.
- **External Fragmentation** (Dynamic Partitioning): Free memory becomes scattered in small, non-contiguous blocks, preventing allocation of larger processes.

A remedy for external fragmentation is **compaction**, which relocates processes to consolidate free memory into a single block. This process involves:

1. Identify fragmented free blocks.
2. Pause running processes.
3. Relocate processes to adjacent memory locations.
4. Update address mappings in the page table.
5. Resume execution.

Compaction, however, introduces overhead due to process relocation and address table updates, which may disrupt real-time systems. In Nepal, where cost-effective solutions are critical, compaction can extend the usability of older systems with limited RAM.

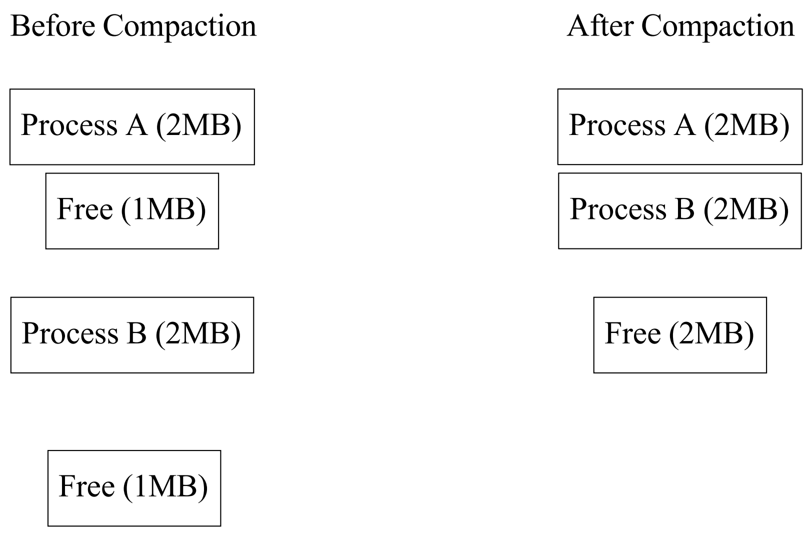


Figure 6: Compaction to Mitigate External Fragmentation