Virtual memory

Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites and program-generated addresses are translated automatically to the corresponding machine addresses.

The size of virtual storage is limited by the addressing scheme of the computer system and the amount of secondary memory available not by the actual number of main storage locations.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.

1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of the main memory such that it occupies different places in the main memory at different times during the course of execution.
2. A process may be broken into a number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time address translation and the use of a page or segment table permits this.

If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded into memory whenever required. Virtual memory is implemented using Demand Paging or Demand Segmentation.

Demand Paging

The process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging. The process includes the following steps are as follows:

1. If the CPU tries to refer to a page that is currently not available in the main memory, it generates an interrupt indicating a memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
3. The OS will search for the required page in the logical address space.
4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision-making of replacing the page in physical address space.
5. The page table will be updated accordingly.
6. The signal will be sent to the CPU to continue the program execution and it will place the process back into the ready state.

Hence whenever a page fault occurs these steps are followed by the operating system and the required page is brought into memory.

Advantages of Virtual Memory

- **More processes may be maintained in the main memory:** Because we are going to load only some of the pages of any particular process, there is room for more processes. This leads to more efficient utilization of the processor because it is more likely that at least one of the more numerous processes will be in the ready state at any particular time.
- **A process may be larger than all of the main memory:** One of the most fundamental restrictions in programming is lifted. A process larger than the main memory can be executed because of demand paging. The OS itself loads pages of a process in the main memory as required.
- It allows greater multiprogramming levels by using less of the available (primary) memory for each process.
- It has twice the capacity for addresses as main memory.
- It makes it possible to run more applications at once.
- Users are spared from having to add memory modules when RAM space runs out, and applications are liberated from shared memory management.
- When only a portion of a program is required for execution, speed has increased.
- Memory isolation has increased security.
- It makes it possible for several larger applications to run at once.
- Memory allocation is comparatively cheap.
- It doesn't require outside fragmentation.
- It is efficient to manage logical partition workloads using the CPU.
- Automatic data movement is possible.

Disadvantages of Virtual Memory

- It can slow down the system performance, as data needs to be constantly transferred between the physical memory and the hard disk.
- It can increase the risk of data loss or corruption, as data can be lost if the hard disk fails or if there is a power outage while data is being transferred to or from the hard disk.
- It can increase the complexity of the memory management system, as the operating system needs to manage both physical and virtual memory.

Page Fault Service Time: The time taken to service the page fault is called page fault service time. The page fault service time includes the time taken to perform all the above six steps.

```
Let Main memory access time is: m
Page fault service time is: s
Page fault rate is : p
Then, Effective memory access time = (p*s) + (1-p)*m
```

Swapping

Swapping is a process out means removing all of its pages from memory, or marking them so that they will be removed by the normal page replacement process. Suspending a process ensures that it is not runnable while it is swapped out. At some later time, the system swaps back the process from the secondary storage to the main memory. When a process is busy swapping pages in and out then this situation is called thrashing.

Thrashing

At any given time, only a few pages of any process are in the main memory, and therefore more processes can be maintained in memory. Furthermore, time is saved because unused pages are not swapped in and out of memory. However, the OS must be clever about how it manages this scheme. In the steady state practically, all of the main memory will be occupied with process pages, so that the processor and OS have direct access to as many processes as possible. Thus when the OS brings one page in, it must throw another out. If it throws out a page just before it is used, then it will just have to get that page again almost immediately. Too much of this leads to a condition called Thrashing. The system spends most of its time swapping pages rather than executing instructions. So a good page replacement algorithm is required.

In the given diagram, the initial degree of multiprogramming up to some extent of point(lambda), the CPU utilization is very high and the system resources are utilized 100%. But if we further increase the degree of multiprogramming the CPU utilization will drastically fall down and the system will spend more time only on the page replacement and the time taken to complete the execution of the process will increase. This situation in the system is called thrashing.

Causes of Thrashing

1. High Degree of Multiprogramming: If the number of processes keeps on increasing in the memory then the number of frames allocated to each process will be decreased. So, fewer frames will be available for each process. Due to this, a page fault will occur more frequently and more CPU time will be wasted in just swapping in and out of pages and the utilization will keep on decreasing.

For example:
Let free frames = 400
Case 1: Number of processes = 100
Then, each process will get 4 frames.
Case 2: Number of processes = 400
Each process will get 1 frame.
Case 2 is a condition of thrashing, as the number of processes is increased, frames per process are decreased. Hence CPU time will be consumed just by swapping pages.

2. Lacks of Frames: If a process has fewer frames then fewer pages of that process will be able to reside in memory and hence more frequent swapping in and out will be required. This may lead to thrashing. Hence a sufficient amount of frames must be allocated to each process in order to prevent thrashing.

Recovery of Thrashing

- Do not allow the system to go into thrashing by instructing the long-term scheduler not to bring the processes into memory after the threshold.
- If the system is already thrashing then instruct the mid-term scheduler to suspend some of the processes so that we can recover the system from thrashing.

Performance in Virtual Memory

- Let p be the page fault rate(0 <= p <= 1).
- if p = 0 no page faults
- if p =1, every reference is a fault.

Effective access time (EAT) = (1-p)* Memory Access Time + p * Page fault time.

Page fault time = page fault overhead + swap out + swap in +restart overhead

The performance of a virtual memory management system depends on the total number of page faults, which depend on "paging policies" and "frame allocation"

Frame Allocation

A number of frames allocated to each process in either static or dynamic.

- **Static Allocation:** The number of frame allocations to a process is fixed.
- **Dynamic Allocation:** The number of frames allocated to a process changes.

Paging Policies

- **Fetch Policy:** It decides when a page should be loaded into memory.
- **Replacement Policy:** It decides which page in memory should be replaced.
- **Placement Policy:** It decides where in memory should a page be loaded.

Frequently Asked Questions

Q.1: How does virtual memory work?

Answer:

Virtual memory works by dividing the virtual address space used by a program into smaller units called pages. The operating system manages a mapping between virtual addresses used by the program and physical addresses in the actual RAM or disk. When a program accesses a virtual address that is not currently in physical memory, a page fault occurs. The operating system then retrieves the required page from disk and brings it into RAM, evicting other pages if necessary.

Q.2: What are the benefits of virtual memory?

Answer:

Virtual memory provides several benefits:

1. ***Increased memory capacity:*** *It allows programs to use more memory than is physically available, enabling the execution of larger programs or multiple programs simultaneously.*
2. ***Memory isolation:*** *Each program operates in its own virtual address space, ensuring that one program cannot access or modify the memory of another program.*

3. ***Simplified memory management:*** *Virtual memory simplifies memory management for both the operating system and application developers by providing a uniform memory model.*
4. ***Improved system stability:*** *Virtual memory helps prevent crashes and system instability by allowing the operating system to handle memory shortages and prioritize memory usage efficiently.*

Q.3: What is a page fault?

Answer:

A page fault occurs when a program references a virtual memory page that is not currently resident in physical memory. This happens when the required page has been paged out to disk or has not been accessed yet. When a page fault occurs, the operating system handles it by fetching the required page from disk and updating the page tables to reflect the new mapping.

Similar Reads