Operating Systems Z02-54c

Frame allocation algorithms for multiple processes

Mustafa Tayyip BAYRAM 257639

OVERVIEW

An important aspect of operating systems, virtual memory is implemented using demand paging. Demand paging necessitates the development of a page-replacement algorithm and a frame allocation algorithm. Frame allocation algorithms are used if you have multiple processes; it helps decide how many frames to allocate to each process. Algorithms that work well for one process can give terrible results if they are extended to multiple processes in a naive way.

There are various constraints to the strategies for the allocation of frames:

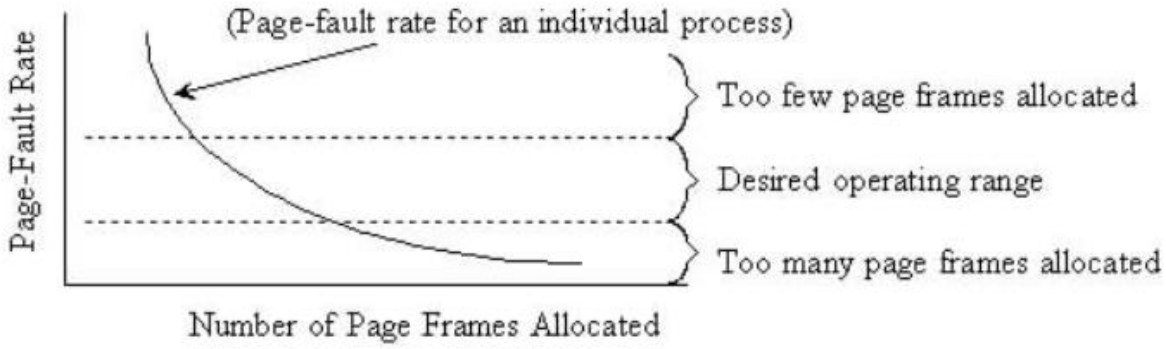
- You cannot allocate more than the total number of available frames.
- At least a minimum number of frames should be allocated to each process. This constraint is supported by two reasons. The first reason is, as less number of frames are allocated, there is an increase in the page fault ratio, decreasing the performance of the execution of the process. Secondly, there should be enough frames to hold all the different pages that any single instruction can reference.

Frame Allocation Algorithms

1. **Random :** Frames are allocated to each process randomly.
   - Disadvantage: In systems with processes of varying sizes, it does not make much sense to give each process randomly frames. Allocation of a large number of frames to a small process will eventually lead to the wastage of a large number of allocated unused frames.
   - If average probabilities are taken, this will probably be the worst working algorithm.

2. **Proportional allocation :** Frames are allocated to each process according to the process size.
   For a process pi of size si, the number of allocated frames is ai = (si/S)*m, where S is the sum of the sizes of all the processes and m is the number of frames in the system. For instance, in a system with 62 frames, if there is a process of 10KB and another process of 127KB, then the first process will be allocated (10/137)*62 = 4 frames and the other process will get (127/137)*62 = 57 frames.
   - Advantage: All the processes share the available frames according to their needs, rather than equally.
   -

3. **Page-Fault Frequency :** OS monitors page-fault rate of each process to decide if it has too many or not enough page frames allocated. Additional, free frames could be allocated from the free-frame list or removed from processes that have too many pages. If the free-frame list is empty and no process has any free frames, then a process might be swapped out.
    - More direct approach than Working Set Model.



4. **Working-Set Model:** Use past references of a process to define a working set of pages that should be kept in main memory. The OS has an integer parameter Δ (say 10,000) that is the size of the working-set window. The working-set window consists of the last Δ references. The working-set for the program consists of the set of pages that have been references in the working-set window. OS allocates enough page frames to each process to hold its current working set. If the sum of all cardinality of all working sets for the loaded processes exceeds the number of physical frames, then the OS could swap out a process. Later when the process is swapped back into memory, its working set of pages can be preloaded back into main memory. Thus, reducing page faults when the process is restarted by prepaging the working set.

➢

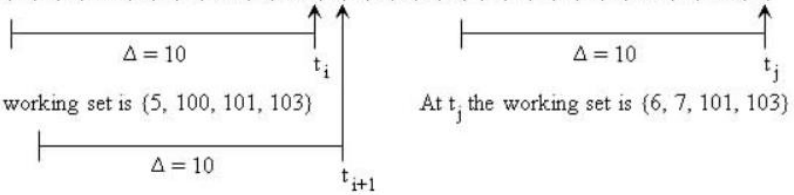
Working-Set Example

Suppose that $\Delta = 10$ (unrealistically small)

Consider the working set at two different times, t_i and t_j

Reference string:

5, 5, 101, 5, 5, 5, 101, 5, 5, 5, 100, 5, 103, 6, 6, 6, 6, 6, 6,6, 7, 7, 7, 7, 7, 7, 103, 7, 101, 7,



At t_i the working set is {5, 100, 101, 103}

At t_j the working set is {6, 7, 101, 103}

At t_{i+1} the working set is {5, 6, 100, 101, 103}