Virtual Memory - Strategies

- **Fetch strategies**. When to bring a page into primary memory.
  - → Demand paging
  - → Prepaging
- **Placement strategies**. Where in primary memory should a new page be placed.
  - → Trivial for paging systems
- **Replacement strategies**. Which page should be replaced, if primary memory is full.

Replacement Strategies

- Optimal
- FIFO
- LRU - Least Recently used
- LFU - Least Frequently used
- NUR - Not Used Recently
- Second Chance
- Clock

NUR

NUR is a simple approximation of LRU

Needs two extra hardware bits in the page table entries

reference bit	= 0 if page not reference
	= 1 if page reference
modified bit (dirty bit)	= 0 if page not modified
	= 1 if page modified

- From the beginning all reference and modified bits are 0
- When a page is referenced, the reference bit is changed to 1 by hardware.
- When a page is modified, the modified bit is changed to 1 by hardware.
- After a while, all bits will be 1 and do not give any information anymore. The reference bits must therefor be periodically reset to 0 by software.

NUR

The NUR method gives four groups of pages:

group 1	unreferenced	unmodified
group 2	unreferenced	modified
group 3	referenced	unmodified
group 4	referenced	modified

The preferred page to replace comes from the group with lowest number

Second Chance and Clock

A disadvantage with FIFO is that also frequently used pages are replaced.

A solution to this is to only replace pages whose reference bit is 0.

Second chance is a modified FIFO algorithm there the reference bit is inspected:

- If the reference bit is 0, the page is replaced.
- If the reference bit is 1, it is set to 0 and the page is moved to the back of the FIFO list.

Clock uses the same principle as second chance, but a circular list is used instead of a linear list. The effect is that no pages need to be moved as first and last is adjacent in a circular list.

Locality

Locality means that processes tend to reference storage in nonuniform highly localized patterns.

This behavior is a precondition for all types of virtual memory and caches.

Two types of locality:

- **Temporal locality** - means that storage locations referenced recently are likely to be referenced in a near future. Supporting program constructs are:
  1. loops.
  2. subroutines.
  3. stacks.
  4. variables used for counting.
- **Spatial locality** - means that once a location is referenced, it is highly likely that nearby locations will be referenced. Supporting program constructs are:
  1. arrays.
  2. sequential code.
  3. the tendency by programmers to to place related variable definitions near one another.

Working sets

Informal definition:

A **working set** is a collection of pages a process is actively referencing.

A process's *working set* must be kept in primary storage, otherwise a condition with excessive paging traffic, called *thrashing*, will result.

Mathematical definition:

A process's *working set*, W(t,w), at time t, is the set of pages referenced during the time interval (t-w,t).

The variable *w* is called the window size and defines the time interval used to measure the working set of the process.

- The working set size varies with time and it is not possible to know how big the working set will be at a certain time in the future.
- A goal for all page based virtual memory systems is to keep the working sets for all active processes in primary storage.

Global versus Local Page Replacement

There are two types of replacement strategies.

- **Global page replacement** - A process is allowed to select a replacement frame from the set of all frames in the system.
- **Local page replacement** - A process may only select from it's own set of allocated frames.

With a *global strategy*, the processes may take frames from each other. Thus the execution time for a process will depend on the other processes in the system.

With a *local strategy*, the operating system must keep a set of pages reserved for every process. It is difficult to know how many pages are needed by each process.

Even with a *local strategy*, the processes disturbs each other because they are sharing both processor and paging disk.

Page Fault Frequency

Page Fault Frequency is a replacement algorithm that can be used in systems with a *local replacement strategy*.

A too high page fault frequency is an indication that the process may be thrashing and an unusually low page fault frequency indicates that too many frames are allocated to the process.

The algorithm estimates the page fault frequency by measuring the time between page fault interrupts.

- If the page fault frequency is above an upper limit, the process is assigned an extra frame.
- If the page fault frequency is below a lower limit, a frame is removed from the process. A suitable frame to remove can be localized with help from the *reference bits* in the page table.

Page Size

What page size should be used?

One important factor is the time it takes to transfer a page between primary storage and disk storage.

Transfer Time for Page

How long time does it take to transfer a page between primary storage and disk memory?

Notations:

T_{page} Total transfer time for page
$T_{transport}$ Transport time between primary storage and disk storage
T_{wait} Average rotational latency + seek time
V Transport speed for page transport
L Page size

Transfer time:

$$T_{page} = T_{transport} + T_{wait} = L/V + T_{wait}$$

Typical values:

$V = 10$ Mbit/s, $L = 10000$ bits, $T_{wait} = 10$ ms

This gives: T_{page} =1 ms + 10 ms

Thus for page sizes of 1 Kbyte or less, the wait time is totally dominating making the transfer time almost independent of page size.

Page Size

Arguments for big pages:

- A small page size requires more pages and thus bigger page tables.
- Small pages makes very inefficient use of the disk bandwidth.

Arguments for small pages:

- Small pages gives less internal fragmentation.
- Big pages will not take full advantage of the programs locality properties. Thus more data than needed will be loaded into primary storage.