1 Resident sets

1.1 Size

How many pages should be allocated to individual processes:

- Small *resident sets* enable to store more processes in memory means **improved** CPU utilisation
- Small resident sets may result in **more** page faults
- Large resident sets may **no longer reduce** the page fault rate (diminishing returns)

A trade-off exists between the sizes of the resident sets and system utilisation.Resident set sizes may be **fixed** or **variable** (i.e. adjusted at runtime). For **variable sized** resident sets, replacement policies can be:

- **Local**: a page of the same process is replaced
- **Global**: a page can be taken away from a different process

Variable sized sets require careful evaluation of their size when a local scope is used (often based on the **working set** or the **page fault frequency**).

2 Working set

The *resident set* comprises the set of **pages of the process** that are in memory. The *working set* $W(t, k)$ comprises the set of **referenced** pages in the last **k** (= working set window) *virtual time units*. For the process k can be defined as **memory references** or as **actual process time**

- The set of **most recently** used pages
- The set of pages used within a pre-specified **time interval**

The working set size can be used as a **guide** for the **number frames** that should be allocated to a process

2.1 Defining and monitoring working sets

The *working set* is a **function** of time **t**:

- Processes **move between localities**, hence, the pages that are **included** in the working set change over time
- Stable intervals **alternate** with intervals of **rapid change**

$|W(t, k)|$ is then variable in time. Specifically:

$$1 \leq |W(t, k)| \leq min(k, N)$$

Where N is the total number of pages of the process.

Choosing the right value for k is **paramount**:

- Too small: **inaccurate**, pages are missing
- Too large: too many **unused** pages present
- Infinity: **all pages** of the process are in the working set

Working sets can be used to **guide** the size of the resident sets

- Monitor the working set
- Remove pages from the resident set that are not in the working set

The working set is costly to maintain `->` page fault frequency (**PFF**) can be used as an approximation.

- If the PFF is increased `->` we need to **increase** k.
- If PFF is very reduced `->` we may try to **decrease** k

3 Replacement Sets

Global replacement policies can select frames from the **entire set**, i.e., they can be taken from other processes.

- Frames are allocated **dynamically** to processes
- Processes **cannot control** their own page fault frequency, i.e., the PFF of one process is influenced by other processes.

Local replacement policies can only select frames that are allocated to the **current** process

- Every process has a **fixed** fraction of memory
- The locally oldest page **is not** necessarily the globally oldest page

Windows uses a **variable approach** with local replacement. Page replacements algorithms explained before can use **both policies**.

4 Paging daemon

It is more efficient to **proactively keep** a number of **free pages** for future page faults. If not, we may have to find a page to **evict** and we write it to the drive (if modified) first when a page fault occurs. Many systems have a background process called a *paging daemon*.

- This process runs at **periodic intervals**
- It inspect the **state of the frames** and, if too few pages are free, it selects pages to **evict** (using page replacement algorithms)

Paging daemons can be combined with **buffering** (free and modified lists) `->` write the modified pages but keep them in *main memory* when possible.

5 Trashing

Assume **all** available pages are in **active use** and a new page needs to be loaded: The page that will be evicted will have to be reloaded soon afterwards, i.e., it is still active. *Thrashing* occurs when pieces are swapped out and loaded again immediately.
If CPU utilisation is too low, then scheduler increases degree of multi-programming

- Frames are allocated to new processes and **taken away** from existing processes
- I/O requests are **queued up** as a consequence of page faults
- CPU utilisation **drops** further `->` scheduler increases degree of multi-programming

5.1 Causes/Solutions

Causes of thrashing include:

- The degree of multi-programming is too high, i.e., the total **demand** (i.e., the sum of all working set sizes) **exceeds** supply (i.e. the **available frames**)
- An individual process is allocated too **few pages**

This can be **prevented** by, e.g., using good page replacement policies, reducing the **degree of multi-programming** (medium term scheduler), or adding more memory. The page fault frequency can be used to **detect** that a system is thrashing.

Reference section

resident set
: In computing, resident set size (RSS) is the portion of memory occupied by a process that is held in main memory (RAM).

working set
: Working set is a concept in computer science which defines the amount of memory that a process requires in a given time interval.