Working Sets

- Proposed by Peter Denning in 1960's when Multics exhibited thrashing.
- Informal definition: the collection of pages that a process is working with and that must therefore be resident if the process is to avoid thrashing.
- The idea is to use the recent memory needs of a process to predict its future needs.
- More formally:
  - Let τ (tau) be the working set parameter.
  - Let the working set of τ be all pages referenced by a process in its last τ seconds of execution.
  - A process will never be executed unless its working set is resident in main memory.
  - Pages outside the working set may be discarded at any time.
- Working sets are not quite enough to solve the problem…

Balance Sets

- If the sum of the working sets of all runnable processes is greater than the size of memory, refuse to run some processes (for awhile).
- Divide the runnable processes into two groups: active, inactive.
- When a process is made active, its working set is loaded.
- When it is made inactive, its working set is allowed to migrate to disk.
- The collection of active processes is called a balance set.
- Now, all you need is an algorithm for moving processes into and out of the balance set.
- What happens if the balance set changes too frequently?

- As working sets change, balance set changes too.
  - This has a problem that you need to constantly update the working set

Balance Sets

- If the sum of the working sets of all runnable processes is greater than the size of memory, refuse to run some processes (for awhile).
- Divide the runnable processes into two groups: active, inactive.
- When a process is made active, its working set is loaded.
- When it is made inactive, its working set is allowed to migrate to disk.
- The collection of active processes is called a *balance set*.
- Now, all you need is an algorithm for moving processes into and out of the balance set.
- What happens if the balance set changes too frequently?
  - You still get thrashing!
- As working sets change, balance set changes too.
  - This has a problem that you need to constantly update the working set

Balance Set Theory

- The idea was that you stored some sort of capacitor with each memory page.
- When the page was referenced, the capacitor was charged.
- Then it would discharge slowly.
- τ would be determined by the size of the capacitor.
- In practice, you want separate working sets for each process, so capacitor should only discharge while process is running.
- Not clear what you do if a page is shared!

Working Sets in Reality

- Use use bits.
- OS maintains an idle time value for each page.
- This is the amount of CPU time received by the process since the last access to the page.
- Periodically, scan all the pages of a process.
- For each use bit that is set, set page's idle time to 0.
- If the use bit is clear, add the process' CPU time (since the last scan) to the idle time.
- Turn all bits off during scan.
- Scans happen on order of every few seconds (in UNIX, τ is on the order of a minute or more).

Parting Thoughts

- What should τ be?
- What happens if τ is too large?
- What happens if τ is too small?
- What algorithms should be used to determine which processes are in the balance set?
- How do we compute working sets if pages are shared?
- How much memory is needed in order to keep the CPU busy?
- In the working set model, the CPU may occasionally be idle even though there are runnable processes.
- Technology changes problems!
  - In a PC or even a VM, thrashing may be a less critical issue than in timesharing systems.
  - If one user has too many processes, she can just kill some!
  - With multiple users, the OS must somehow arbitrate fairly.