Dealing with thrashing

- Approach 1: working set
  - Thrashing viewed from a caching perspective: given locality of reference, how big a cache does the process need?
  - Or: how much memory does process need in order to make reasonable progress (its working set)?
  - Only run processes whose memory requirements can be satisfied

- Approach 2: page fault frequency (PFF)
  - Thrashing viewed as poor ratio of “page fetch” to “useful work”
  - PFF = page faults / instructions executed
  - If PFF rises above threshold, process needs more memory. If not enough memory on the system, swap out.
  - If PFF sinks below threshold, memory can be taken away

Two-level scheduler

- Divide processes into active and inactive
  - Active – means working set resident in memory
  - Inactive – working set intentionally not loaded
- Balance set: union of all active working sets
  - Must keep balance set smaller than physical memory
- Use long-term scheduler
  - Moves processes from active to inactive state until balance set is small enough
  - Periodically allows inactive to become active
  - As working set changes, must update balance set
- Complications
  - How to chose idle time threshold T?
  - How to pick processes for active set?
  - How to count shared memory (e.g., libc.so)?