CS 3204
Operating Systems

Lecture 27

Godmar Back

Announcements

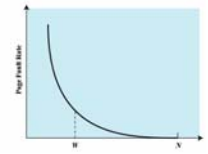
- Project 3 page table design document
  - due tonight
  - only data structures & comments, no code
- Project 3 due April 13
- If you have bugs left in Project 2, seek help quickly
  - To pass course, must have 95% passing P2 and show reasonable effort on P3 & P4 – can't do that until P2 is done
  - Vijay or I will help you go over your code and point out problems

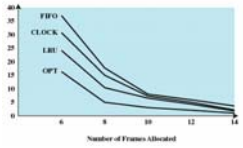
VM Design Issues

N-bit Clock Algorithm

- 1-bit says was recently used or wasn't
  - But how recently?
- Idea: associate n-bit counter with page
  - "age" or "act_count"
  - have R-bit as before
- When hand passes page
  - act_count >>= 2 aging
  - act_count |= (R << (n-1)) recent access
- Replace page with lowest act_count

of Page Faults vs Frame Allocation





- Desired behavior of paging algorithm: reduce page fault rate below "acceptable level" as number of available frames increases
- Q.: does increasing number of physical frames always reduce page fault rate?
  - A.: usually yes, but for some algorithms not guaranteed ("Belady's anomaly")

Page Buffering

- Select victim (as dictated by page replacement algorithm – works as an add-on to any algorithm we discussed)
- But don't evict victim – put victim on tail of victim queue. Evict head of that queue instead.
- If victim page is touched before it moves to head of victim queue, simply reuse frame
- Further improvement: keep queue of unmodified victims (for quick eviction – aka *free page list*) and separate queue of modified pages (aka *modified list* - allows write-back in batch)
- Related issue: when should you write modified pages to disk?
  - Options: demand cleaning vs pre-cleaning (or pre-flushing)

Local Replacement

- So far, considered global replacement algorithms
  - Most widely used
- But could also divide memory in pools
  - Per-process or per-user
- On frame allocation, requesting process will evict pages from pool to which it belongs
- Advantage: Isolation
  - No between-process interference
- Disadvantage: Isolation
  - Can't temporarily "borrow" frames from other pools
- Q.: How big should pools be?
  - And when should allocations change?

When Virtual Memory works well

- Locality
  - 80% of accesses are to 20% of pages
  - 80% of accesses are made by 20% of code
- Temporal locality:
  - Page that's accessed will be accessed again in near future
- Spatial locality:
  - Prefetching pays off: if a page is accessed, neighboring page will be accessed
- If VM works well, average access to all memory is about as fast as access to physical memory

VM Access Time & Page Fault Rate

access time = p * memory access time
+ (1-p) * (memory access time + page fault service time)

- Consider expected access time in terms of fraction p of page accesses that don't cause page faults.
- Then 1-p is page fault frequency
- Assume p = 0.99, assume memory is 100ns fast, and page fault servicing takes 10ms – how much slower is your VM system compared to physical memory?
- access time = 99ns + 0.01*(10000100) ns ≈ 100,000ns or 0.1ms
  - Compare to 100ns or 0.0001ms speed ≈ about 1000x slowdown
- Conclusion: even low page fault rates lead to huge slowdown

When Virtual Memory Does Not Work Well

- System accesses a page, evicts another page from its frame, and next access goes to just-evicted page which must be brought in
- Worst case a phenomenon called Thrashing
  - leads to constant swap-out/swap-in
  - 100% disk utilization, but no process makes progress
    - CPU most idle, memory mostly idle

When does Thrashing occur?

- Process does exhibit locality, but is simply too large
  - Here: locality hurts us
- Process doesn't exhibit locality
  - Does not reuse pages
- Processes individually fit & exhibit locally, but in total are too large for the system to accommodate all

What to do about Thrashing

- Buy more memory
  - ultimately have to do that
  - increasing memory sizes ultimately reason why thrashing is nowadays less of a problem than in the past – still OS must have strategy to avoid worst case
- Ask user to kill process
- Let OS decide to kill processes that are thrashing
  - Linux has an option to do that
- In many cases, still: reboot only time-efficient option