CS 471 Operating Systems

Yue Cheng

George Mason University
Fall 2019

Midterm II

- Thursday, Nov 7, 1:30pm – 2:45pm
  - 75 min, closed book, closed note

- Covering topics from lec-3a to lec-4f
  - CPU scheduling:
    - FIFO/SJF/SRTF/RR/Priority/MLFQ
  - Memory management:
    - Paging/PT/TLB/Swapping/Page Replacement

CPU Scheduling Policies

- FIFO
  - How it works?
  - Its inherent issues (why we need SJF)?
- SJF
  - How it works?
  - Any limitations (why we need SRTF)?
- SRTF (preemptive SJF)
  - How it works? How it solves SJF's limitations?

Various Metrics

- Average waiting time
- Average turnaround time

- How to calculate each metric under a specific schedule (Gantt chart)

CPU Scheduling Policies (cont.)

- RR
  - How it works?
  - Why it is needed (compared to SJF & SRTF)?
    - The turnaround time vs. response time tradeoff
  - Impact of quantum tuning on turnaround time
- Priority
  - How it works?
  - Problems of Priority scheduling and solution?
- MLFQ
  - How it works?
  - Rules that were discussed in lecture. Which rule solves what problem?

Virtual Memory Accesses

- Virtual memory accesses for executing one assembly instruction
  - Fetch instruction at addr XYZ
  - Mem load/store if any
- Segfault vs. page fault
  - Basic concepts of memory segmentation

Paging

- Virtual addresses and physical addresses
  - VPN, PFN, page offset
- Virtual to physical address translation
  - (Basic) linear page table: VPN as index of the array
    - Each PTE contains PFN + other status info
- Page tables' performance and memory issues
  - Performance problem – TLB to the rescue
  - Memory consumption problem – various solutions

PT's are Too Slow – Translation Lookaside Buffer (TLB)

- Address translation steps
  - Each virtual address access requires **two** physical memory accesses: one for PTE access, one for data access (assuming a 1-level PT)
  - Too much performance overhead!
- TLB entry: `VPN | PFN | ASID | Valid | Prot | …`
- Having TLB as a hardware cache to accelerate address translation
  - PT access workloads exhibit **data locality**
  - To save the physical memory access(es) for fetching PTE(s)

PT-related calculation

- Assume a linear page table array

- Virtual address: VPN + offset
  - Assume a 32-bit virtual address space
  - 4KB pages: offset has 12 bits
  - How many pages: 2^{20} pages since VPN has 20 bits
  - How much memory: **4MB** assuming each PTE has 4 bytes
    - 2 ^ (32-log(4KB)) * 4 = 4MB

PT's are Too Big

- Approach 1: Linear Inverted Page Table
  - Whole system maintains only **one** PT
  - Performs a whole-table linear search using `pid+VPN` to get the index
    - The index corresponds to the PFN: An "**Inverted**" process compared against the baseline linear page table array
- Approach 2: Hash Inverted Page Table
  - Leverages hashing to reduce the algorithmic time complexity from `O(N)` to `O(1)`
- Approach 3: Multi-Level Page Table
  - Uses hierarchy to reduce the overall memory usage

Page Replacement Policies

- FIFO
  - Why it might work? Maybe the one brought in the longest ago is one we are not using now
  - Why it might not work? No real info to tell if it's being used or not
  - Suffers "**Belady's Anomaly**"
- Random
  - Why it might work? Sometimes non intelligence is better
  - Why it might not work? No real info to tell if it's being used or not
- OPT
  - Assume we know about the future
  - Not practical in real cases: **offline** policy
  - However, can be used as a **best case baseline** for comparison purpose
- LRU
  - Intuition: we can't look into the future, but let's look at past experience to make a good guess
  - Our "bet" is that pages used recently are ones which will be used again (**principle of locality**)

Average Memory Access Time

- Metric to quantify the effective memory access performance
  - Assume a more generic two-tier memory (or storage) model
  - $\texttt{AMAT = Hit\_rate*T}_{T1} + \texttt{Miss\_rate*T}_{T2}$
  - T_{T1} : Performance cost given existence of Tier 1
  - T_{T2} : Performance cost given absence of Tier 1

- Variants
  - Simple case: If Tier 1 is memory, then Tier 2 can be disk
  - TLB-related AMAT
  - Tier 1 can be CPU cache, tier 2 can be memory, and tier 3 can be disk