Course Review

ECE469

Apr 25

Yiying Zhang

ABET

- ABET requirements and remedy homework online
  - Requirements under each objective have an OR relationship
- ABET outcome is orthogonal to your final grades
  - Passing ABET doesn’t necessarily mean you pass the course (but in most cases it does)
  - Failing ABET however will most likely result in failing the course
- I will posted tentative ABET outcomes on blackboard based on your progress so far.
- If your outcome is fail (0 for fail, 1 for pass), come talk to me to see what you need to achieve in lab 5 or final or ABET remediation homework to pass ABET requirements.
- My goal is to pass as many as possible in ABET

Plan

- Final exam: HAMP 1144, May 3, 7-9pm
- Office hours: today until noon, Thur until noon
- Course evaluation close on May1st
- 2nd half semester review
- Course summary (annotated slides from lec1)

Course review – 2nd half semester

Page Replacement Policies

- Optimal
- Random
- FIFO
  - Belady's anomaly
- Approximate LRU, NRU
- FIFO with 2nd chance
- Clock: a simple FIFO with 2nd chance

More Virtual Memory

- Thrashing
- Working set model, size, replacement aglorithm
- Shared memory
- COW

Virtual Memory Questions to Think About

- What is the use of optimal algo?
- If future is unknown, what make us think there is a chance for doing a good job?
- Without addi. hardware support, the best we can do?
- What is the minimal hardware support under which we can do a decent job?
- Why is it difficult to implement exact LRU? Exact anything
- For a fixed replacement algo, more page frames → less page faults?
- How can we move page-out out of critical path?

More Virtual Memory Questions

- Per-process vs. global page replacement
- Thrashing
- What causes thrashing?
- What to do about thrashing?
- What is working set?
- What’s the benefit of Copy-on-Write?

Practice Question (Quiz 2)

- Consider the following virtual page reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.
- How many page faults will there be under the LRU replacement algorithm on Nick’s PC which has 4 physical pages? How many page faults will there be on Riley’s machine which has 3 physical pages?

- Nick’s: 8
- Riley’s: 10

Storage and File System

Storage Device

- Disk Internals
  - Seek/rotation, random/sequential accesses
- SSD Internals
  - Flash read/write/erase, the granularity of them
  - Erase-before-write, flash wear
  - FTLs
  - Garbage collection and wear leveling
- Both Disks and SSDs can fail!

RAID

- Two motivations
  - Performance, reliability
- Two main ideas
  - Striping, mirroring (parity)

no RAID level 2 3 in exam

- RAID levels: 1-6

Practice question

- Assume that
    - you have a mixed configuration comprising disks organized as RAID Level 1 and as RAID Level 5 disks;
    - the system has flexibility in deciding which disk organization to use for storing a particular file.
    - you have a mixed workload of *frequently-read* and *frequently-written* files
- Which files should be stored in the RAID Level 1 disks and which in the RAID Level 5 disks in order to optimize performance?

File System Overview

- File system abstraction
  - File, directory, FS APIs
- Directories
  - Different directory organizations
  - Directory internals
  - Path walk
  - Hard link, soft link
- Metadata
  - Inode

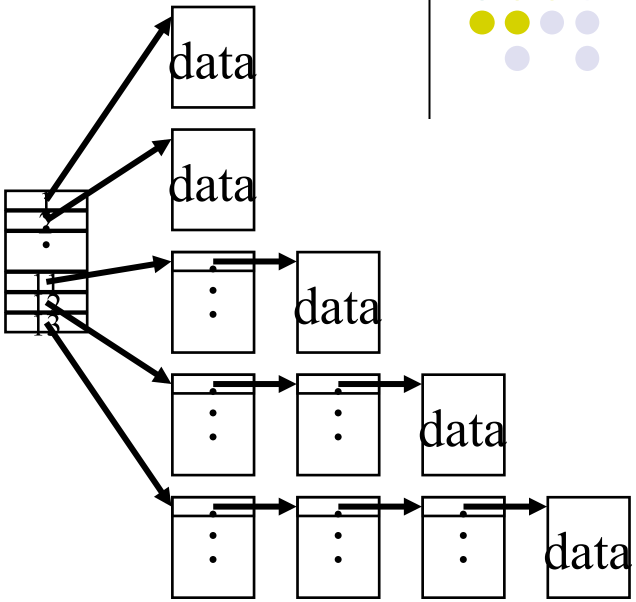
File Allocation

- Two tasks:
  - How to allocate blocks for a file?
  - How to design inode to keep track of blocks?
- Allocation methods:
  - Contiguous
  - Extent-based
  - Linked
  - File-allocation Tables
  - Indexed
  - Multi-level Indexed
- Free space management
  - linked list
  - bitmap

UFS

- UFS
  - multi-level indexed files
  - Inodes all stored on outermost track
  - Free-block linked list
  - Two problems of UFS
    - data blocks are allocated randomly in aging file systems
    - inodes are allocated far from blocks

Indirect blocks addressing ranges



- Assume blocksize = 1K
  - a block contains 1024 / 4 = 256 block addresses
- direct block address: 10K
  - indirect block addresses: 256K
  - double indirect block addresses: 256 * 256K = 64M
  - tripe indirect block addresses: 256 * 64M = 16G

What happens in accessing block 23, 5, 340?

FFS and LFS

- FFS
  - Cylinder group
    - Inodes and data within same dir
  - Free block managed with bitmap

- File system buffer cache

- LFS
  - All writes buffered into chunks and go to an append-only log
  - Locating data is more difficult
  - Needs background cleaner

Journaling File System

- Data reliability
  - Three threats: three directions to prevent data loss
- The problem with write back cache under system crash
  - One file system operation consists of multiple sub-operations, they should be atomic
- Journaling
  - Write to a redo log first, in a transactional way
  - Journal checkpointing, crash recovery
  - Ext3: three journaling modes

Distributed File System

- DFS and client/server model

- NFS
  - Transparency through indirection of VFS
  - Two key ideas for fast crash recovery
    - Stateless server
    - Idempotent operations
  - Client cache and cache consistency

Course Summary
(annotated slides from Lecture 1)

[lec1] What is an OS?

“Code” that *sits between*:

- programs & hardware
- different programs
- different users

But what does it do/achieve?

[lec1] What is an OS?

- Resource manager
- Extended (abstract) machine
- A giant interrupt handler!

Makes computers efficient and simple to use

[lec1] Separating Policy from Mechanism

Policy – decisions on how to use tool

Examples:

- CPU scheduling policies
- Page replacement policies
- Buffer cache replacement policies
- Disk allocation policies

Mechanism – tool to achieve some effect

Examples:

- Priority scheduling vs. lottery scheduling
- FIFO w/ 2nd chance vs. Clock: a simple FIFO w/ 2nd chance

Separation leads to flexibility

[Ice1] Is there a perfect OS?

Portability

Security

Fairness

Robustness

Efficiency

Interfaces

- Conflicting goals
  - Fairness vs efficiency
    - SJF vs. RR
    - FIFO vs. SCAN
  - Efficiency vs robustness
    - Buffer caching
- Don’t know future
  - CPU scheduling
  - Page replacement
  - Disk scheduling

[lec1] There is no magic in OS design

This is Engineering

- Imperfection
  - Don't know future
- Tradeoffs
  - Segmentation vs. paging
  - Read/write API vs. mmap
- Constraints
  - hardware, cost, time
    - FIFO w/ 2nd chance
    - Enhanced version
    - Approx. LRU
- Optimizations
  - After functionality
  - 1-level paging -> 2-level
  - Buffer caching

Nothing's Permanent

- High rate of change
  - Killer-app: Databases/web servers
  - Arch: Multi-core
- Cost / benefit analyses
  - motivation for mmap
  - Semaphore impl on multiprocessor
- One good news:
  - Lots of inertia
    - Principle of locality
      - TLB
      - Demand paging
      - Buffer caching
    - Extra level of indirection
      - Dynamic memory relocation
      - 1-level paging -> 2-level paging
      - UNIX multi-level indexed files

[lec1] About this course…

Principles of OS design

- Some theory
  - SJF optimal
  - Working set modeling
- Some rational
  - Optimize the common case
    - Sequential file access
  - Locality -> caching
  - Why mmap()?
- Lots of practice
  - How much locality?
  - Dist. of file size (UFS inode)
  - I/O access pattern (dir/files)
  - Buffer cache size vs. VM size?

Goals

- Understand OS design decisions
- Basis for future learning

To achieve the goals:

- Learn concepts in class
- Get hands dirty in labs

Great ideas in Computer System Design (1)

- “*All computer science problems can be solved with an extra level of indirection*”

  -- David Wheeler

1. Dynamic memory relocation
   - Base&bound, segmentation, paging
2. One-level paging → Two-level paging
3. UFS multi-level indexed files
4. NFS: transparency via VFS

Great ideas in Computer System Design (2)

- Principle of locality → Caching

1. TLB
2. Demand paging (VM)
3. Buffer cache in FS
4. (On-disk cache)
5. (Client caching in NFS)

5. (Hardware cache, L1, L2, etc.)