Course Summary and Final Review

CS 202: Advanced Operating Systems

UNIVERSITY OF CALIFORNIA, RIVERSIDE

What was this course about?

- How has the role of the OS evolved over time?
- What are the underlying principles of OS?
- What are current and future trends in OS?
- Make it real: lab assignments to get some experience with OS development

- *Ready to do Systems research?*

Topics we have covered

Week	Date	Day	Class activity
0	09/23	Thu	Course Overview
1	09/28	Tue	OS Organization XV6 Overview
	09/30	Thu	Processes and Threads
2	10/05	Tue	Virtual Memory
	10/07	Thu	Extensible OS Design I
3	10/12	Tue	Extensible OS Design II
	10/14	Thu	Scheduling
4	10/19	Tue	Scheduler Design
	10/21	Thu	Synchronization
5	10/26	Tue	Concurrency and Memory Consistency
	10/28	Thu	**Midterm**
6	11/02	Tue	Multicore OS
	11/04	Thu	Scalability to Many Cores
7	11/09	Tue	I/O and File Systems
	11/11	Thu	**Campus holiday; No class**
8	11/16	Tue	File Systems II
	11/18	Thu	Distributed File Systems
9	11/23	Tue	Cloud Storage
	11/25	Thu	**Campus holiday; No class**
10	11/30	Tue	Virtualization
	12/02	Thu	OS for Embedded and IoT

Papers we have reviewed

Readings and Assignment
Ch02
Ch04, Ch05, Ch06, Ch26
Ch13, Ch18, Ch19, Lab 1
Reading: Spin (critique)
Reading: Exokernel (critique), Optional: L4
Ch07, Ch08, Ch10
Reading: Lottery Scheduling (critique), Stride Scheduling, Scheduler Activations, Lab 2
Ch28, Ch29, Ch31, Ch32
Reading: RCU
Reading: Barrelfish (critique), Optional: LegoOS
Reading: Linux scalability (critique), Optional: Decade of wasted cores
Ch37, Ch39, Ch40, Ch41, Lab 3
Reading: LFS, FFS
Reading: NFS, Optional: AFS
Reading: GFS (critique), Optional: HDFS
Reading: Xen (critique), Appendix B
Optional: SW/HW techniques for x86
Reading: Tock (critique), TinyOS, Optional: Contiki, InK

Final Exam

- 12/04/2021, Saturday, 9 am to 11 am, at Sproul 1102
- Closed book with cheat sheets
  - Up to ~~3~~ **5** sheets of letter or A4 paper
- Question types:
  - True and false, single choice, ~~multiple choices~~, short answers, ~~long answers~~
  - Background on OS concepts & papers discussed
  - No “None of the above” type of choices

Scope

- Lecture slides from 11 to 20 (ignore “12_Review for Midterm.pdf”)

10/21	Thu	Synchronization	Ch28, Ch29, Ch31, Ch32
10/26	Tue	Concurrency and Memory Consistency	Reading: RCU
10/28	Thu	**Midterm**	
11/02	Tue	Multicore OS	Reading: Barrelfish (critique), Optional: LegoOS
11/04	Thu	Scalability to Many Cores	Reading: Linux scalability (critique), Optional: Decade of wasted cores
11/09	Tue	I/O and File Systems	Ch37, Ch39, Ch40, Ch41, Lab 3
11/11	Thu	**Campus holiday; No class**	
11/16	Tue	File Systems II	Reading: LFS, FFS
11/18	Thu	Distributed File Systems	Reading: NFS, Optional: AFS
11/23	Tue	Cloud Storage	Reading: GFS (critique), Optional: HDFS
11/25	Thu	**Campus holiday; No class**	
11/30	Tue	Virtualization	Reading: Xen (critique), Appendix B Optional: SW/HW techniques for x86
12/02	Thu	OS for Embedded and IoT	Reading: Tock (critique), TinyOS, Optional: Contiki, InK

Synchronization

- Atomicity
- Mutual exclusion
- Spinlock
- Disabling interrupts
- Test-And-Set (TAS) lock
- Semaphore
- Deadlock conditions
- Lock ranking
- RCU

True and False

- Read-Copy Update (RCU) requires garbage collection for old versions of data

  TRUE

- Hardware atomic instructions are expensive since they may stall other CPUs in a multi-core system

  TRUE

- Deadlocks do not occur with spinlocks

  FALSE

Multicore OS

- Scalability
- Amdahl’s law
- Cache coherence
- Problem of Test-And-Set (TAS) Lock
- Multikernel
  - Message passing
  - Replicas
  - CPU drivers and monitors
  - Inter-core communication
  - Memory management
  - System knowledge base

Scalability

- Eliminate bottlenecks in the Linux kernel
  - Multicore packet processing
    - Per-core RX/TX queues
  - Sloppy counters
    - Local reserve for reference counters
  - Lock-free comparison
    - Generation counter
  - Per-core data structures
  - Eliminating false sharing
    - Use separate cache lines

Short Answer

- If 80% of a program is parallelizable, what is the maximum speedup achievable with an infinite number of processors?

$$SPEEDUP = \frac{1}{(1-p)+\frac{p}{s}}$$

If s → ∞, speedup approaches 1/(1-p)

1/(1-0.8) = 5

Single Choice

- Typically, it is hard to achieve “X” times of speedup by adding “X” processors. Which of the following is **NOT** the correct reasons for imperfect scalability?
  A. Multiple tasks updates the shared data protected by RCU simultaneously
  B. Tasks compete for memory bus
  C. Too few tasks are running
  D. Shared data updated often are located on the same cache line as those read often
  E. The kernel uses per-core data structure
  F. Shared kernel states are protected by mutex locks

E: good for scalability

Short Answer

- In multikernel, the CPU drivers are completely event-driven, [   ], and [   ]. Choose the correct words to fill in the blanks.

  Word pool: time-triggered, message-based, preemptible, nonpremptible, consistent, global, single-threaded, multi-threaded, hardware-neutral,

single-threaded, nonpreemptible

I/O and Storage Devices

- I/O layers
  - Polling vs. Interrupts
- Disks and SSD
  - Disk I/O time = Seek Time + Rotation Time + Transfer Time
  - SSD Advantages
  - Write Amplification
  - Wear Leveling
- Virtual File System
- Inode
- UNIX File System (UFS)

FFS and LFS

- FFS
  - Cylinder groups
  - Larger block sizes
  - Sub-blocks
- LFS
  - Sequential writes
  - inode map
  - Checkpoint Region
- Pros and cons

True and False

- Virtual file system (VFS) provides a uniform interface to user programs regardless of the actual file system being used

  TRUE

- Larger block size of Fast file system (FFS) help improve latency

  FALSE

- In Log-structure File System (LFS), there may exist multiple versions of the same file

  TRUE

Single Choice

- Which one is correct about polling vs. interrupts?
  A. Interrupts consume more CPU time than polling for long I/O operations
  B. Polling improves CPU utilization if the device needs service from the CPU occasionally
  C. Interrupts allow asynchronous I/O in programs
  D. The number of instructions for handling an event after polling is higher than that for handling the same event after receiving an interrupt

C

Single Choice

- Which of the following is **NOT** the feature of the Fast File System (FFS)?
  A. Cylinder groups to improve average access time
  B. Larger block size to improve bandwidth
  C. Larger block size to support larger files
  D. Replicated superblocks in cylinder groups
  E. Pre-allocate blocks to improve write performance

E

Network File System (NFS)

- Stateful vs. stateless protocols
- NFS
  - Stateless
  - Single centralized server
  - File lookup
  - Caching and consistency
  - Idempotent requests
  - File locking
  - Time synchronization

Google File System (GFS)

- Single master server
- Chunk servers
- Replicas
- User-space API
- Namespace
- Lease & Mutation order
- Record append

True and False

- In the design of network file systems, stateful protocols make crash/disconnect recovery easier

FALSE

- Google File System (GFS) is designed for expensive high-endurance disk drives that rarely fails

FALSE

Single Choice

- To open a file /foo/bar/cs202/test.c stored in the server, how many lookup messages will be made by the client?
  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

E: 4 lookup messages

Virtualization

- Trap-and-emulate
- x86 virtualization challenges
- Dynamic binary translation
- Shadow paging
- Para-virtualization
  - Xen
  - Dom0
  - CPU/memory/IO virtualization
- Hardware extension

Single Choice

- The trap-and-emulate approach faces multiple challenges in virtualizing classical x86 architectures (without hardware extensions). Which one is correct?
  A. The guest OS does not know that it’s not running in a privileged mode
  B. A privileged instruction in the guest OS does not trigger a trap
  C. x86 does not provide a mechanism to set write-protected pages for shadow paging
  D. x86’s hierarchical page table structure prevents the use of shadow page tables

B

Single Choice

- The trap-and-emulate approach faces multiple challenges in virtualizing classical x86 architectures (without hardware extensions). Which one is correct?
  A. The guest OS does not know that it's not running in a privileged mode
  B. A privileged instruction in the guest OS does not trigger a trap
  C. x86 does not provide a mechanism to set write-protected pages for shadow paging
  D. x86's hierarchical page table structure prevents the use of shadow page tables

B

Short Answer

- Can shadow paging be faster in address translation than nested/extended paging? Give your answer with a short explanation

Embedded OS

- TinyOS
  - Concurrency: Event-driven architecture; no preemption
  - Modularity: scheduler + graph of components
  - Compiled into one executable
  - Static memory allocation
- Tock
  - Rust - Isolate drivers
  - MPU - Isolate applications
  - Grants - enable dynamic memory without crashing the kernel

True and False

- TinyOS implements priority-based scheduling for tasks

  FALSE

- Tock supports dynamic memory allocation for processes

  TRUE

- Tock supports virtual memory

  FALSE