Final class
Today

- Circle-back to two missed topics
- Software protection Chap 8.4
- Fault tolerance 10.3
- Course wrap up
Software Protection

• Talked about hardware protection
  – Address translation, mode bits

• What about doing this in software
  – Simplify hardware
  – Application-level protection (e.g. browser)
  – Protection inside the kernel itself (e.g. 3rd party device drivers)
Methods

- Trap for each instruction?
  - Solution (browser as OS)
    - Interpreters, e.g. JavaScript
    - JavaScript attacks: cross-scripting
  - Isolate
    - Run browser within a controlled process: protect OS
    - Run tab in its own process: protect browser
More Methods

– Use safe language, trusted compilers
  • Users don’t want to be constrained

• Language-independent solution
  – Sandbox
  – Compile software memory checks into executable
  – E.g. native-client (does this for C code)
  – Downsides?
Fault Tolerance

• Long-running program meets power-glitch
• Checkpoint/restart
  – User-level
  – System-level
  – copy-on-write
  – Problems?
Solutions

- Common approach: checkpoint virtual machine
  - Copy entire contents of memory and registers

- Performance ...

- Periodic checkpoints
  - Take periodic snapshots and log of subsequent operations: replay log against checkpoint at restart
  - Take checkpoint as a delta over previous
Solutions

- Incremental checkpoints

```
Checkpoint 1 (Full)

A
B
C
D
E

Checkpoint 2

P
Q

Checkpoint 3

R
S

Restore (Full)

A
P
R
Q
S
```
The END
Major Topics: 100K feet

• Protection
  – Kernel/user mode, system calls

• Concurrency
  – Threads, monitors, deadlock, scheduling

• Memory management
  – Address translation, demand paging

• File systems
  – Disk, flash, file layout, transactions
OS as Referee

• Protection
  – OS isolates apps from bugs or attacks in other apps

• CPU scheduling
  – OS decides which application thread is next onto the processor

• Memory allocation
  – OS decides how many memory frames given to each app

• File system
  – OS enforces security policy in accessing file data
## OS as Illusionist

### Physical Reality
- Limited # of CPUs
- CPU interrupts and time slicing
- Limited physical memory
- Apps share physical machine
- Computers can crash

### Abstraction
- Can assume near infinite # of processes/threads
- Each thread appears to run sequentially (at variable speed)
- Near-infinite virtual memory
- Isolation between apps via processors or VMs
- Changes to file system are atomic and durable
OS as Abstraction Provider

- Locks and condition variables
  - Not test&set instructions
- Named files and directories
  - Not raw disk block storage
- Pipes: stream interprocess communication
  - Not fixed size read/write calls
- Memory-mapped files
  - Not raw disk reads/writes
OS Trends and Future Directions

• Optimize for the computer’s time
  => optimize for the user’s time
• One processor => many
• One computer => server clusters
• Disk => solid state memory
• Modest memory => Huge memory
• Operating systems at user level
  – Browsers, databases, servers, parallel runtimes, sandboxes
Some Cross-Cutting Themes

• Indirection

• Batching to overcome latency

• Isolation
The Final

• Incremental, sort of ...
• Chapters 6->6.4, 8.4, 10.3, 11, 12, 13, 14
  – Multi-object synchronization (- deadlock)
  – Software protection
  – Fault tolerance
  – File systems
  – Storage systems
  – Reliable storage
• Closed book
• 75 minute exam, have over 2 hours
• Mix of short and long questions 1/3 : 2/3
Evaluations