Final class
Today

• Back to reliability briefly
• Circle-back to two missed topics
  – Software protection Chap 8.4
  – Fault tolerance 10.3
• Course wrap up
• Evaluations

• How are the VMs working?
Software Protection

- Talked about hardware protection
  - Address translation, mode bits

- What about doing this in software, reasons:
  - Simplify hardware
  - Application-level protection (e.g. browser needs to protect itself)
  - Protection inside the kernel itself (e.g. 3rd party device drivers)
Methods

- Trap for each instruction?
  - Way too expensive

- 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 (e.g. Java everywhere)

• Language-independent solution
  • Sandbox
  • Compile software memory checks into executable
  • E.g. native-client (does this for C code)
  • \texttt{mov addr reg =>}
    \hspace{1em} if addr between low\&high => \texttt{mov addr reg}
  • Downsides?
Fault Tolerance

• Long-running program meets power-glitch
• Checkpoint/restart
  – User-level
  – System-level
  – Issues?
    • Kernel state
    • Size of checkpoint
  – Block until done or not (copy-on-write)
Solutions

• Checkpoints can be large
  – Memory-intensive, virtual machines, kernel state, ...
  – Performance issue

• #1 Periodic checkpoints
  – Take periodic snapshots and log of subsequent operations:
    replay log against most recent checkpoint at restart
    • Restart cost: re-executions

• #2 Take checkpoint as a delta over previous one
  • Restart requires reading many checkpoints
Solutions

- Incremental checkpoints
The END
Major Topics: 100K feet

• Protection
  – Kernel/user mode, system calls

• Concurrency
  – Threads, synchronization, deadlock, scheduling

• Memory management
  – Address translation, demand paging, virtual memory

• File systems
  – Disk, flash, file layout, reliability/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

<table>
<thead>
<tr>
<th>Physical Reality</th>
<th>Abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited # of CPUs</td>
<td>Can assume near infinite # of processes/threads</td>
</tr>
<tr>
<td>CPU interrupts and time slicing</td>
<td>Each thread appears to run sequentially (at variable speed)</td>
</tr>
<tr>
<td>Limited physical memory</td>
<td>Near-infinite virtual memory</td>
</tr>
<tr>
<td>Apps share physical machine</td>
<td>Isolation between apps via processes or VMs</td>
</tr>
<tr>
<td>Computers can crash</td>
<td>Changes to file system are atomic and durable</td>
</tr>
</tbody>
</table>
OS as Abstraction Provider

• Locks and condition variables
  – Not test&set instructions

• Named files and directories
  – Not raw disk block storage

• Process
  – Not x CPU cycles, y memory, z open files, ...

• 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 core => many
• One computer => server clusters
• Disk => persistent solid state memory/PCM
• Modest memory => huge memory
• Operating systems everywhere (at user level)
  – browsers, databases, servers, parallel runtimes, sandboxes
• Operating system for Internet of Things
Some Cross-Cutting Themes

• Indirection
  – Virtual addressing, File storage, ...

• Batching to overcome latency
  – Disk access, Disk scheduling, LFS

• Isolation
  – processes, transactions, ...
The Final

• Incremental, sort of ...
• Chapters 8.4, 9, 10.3, 11, 12, 13, 14
  – Virtual memory > Basics
  – File systems
  – Storage systems
  – Reliable storage
  – Software protection
• Closed book
• 75 minute exam, have over 2 hours
• Mix of short and long questions 1/3 : 2/3
Questions?
Evaluations