Advanced Computer Systems Architecture

Chip-Multiprocessors: Applications and Architectures

CSE 526M
Prof. Patrick Crowley

Plan for Today

• Palmer talk review
• Questions
• Today’s discussion
Project Logistics

- Dates
  - Today’s date: Mar 23
  - End date: (3.5 weeks later) Thursday, April 15
- Weekly Milestones

<table>
<thead>
<tr>
<th>M1</th>
<th>Mar 4</th>
<th>Implementation 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2</td>
<td>Mar 18</td>
<td>Implementation 2</td>
</tr>
<tr>
<td>M3</td>
<td>Mar 25</td>
<td>Implementation 3</td>
</tr>
<tr>
<td>M4</td>
<td>Apr 1</td>
<td>Implementation Wrap-up</td>
</tr>
<tr>
<td>M5</td>
<td>Apr 8</td>
<td>Plan future work, Reports</td>
</tr>
<tr>
<td>M6</td>
<td>Apr 15</td>
<td>Presentations</td>
</tr>
</tbody>
</table>

Hints for Computer System Design

- System vs. algorithm
  - Requirements are imprecise, more complex and subject to change
  - More internal structure, thus more internal interfaces
  - Measure of success is uncertain (metrics?)
- Sea of possibilities
  - Many choices, but which ones matter most?
- What hints does Lampson give?
  - Functionality
  - Speed
  - Fault-tolerance
Who is he?

- PhD from Berkely EECS, AB from Harvard
- Positions
  - Microsoft Corp., Architect (currently)
  - MIT CSEE, adjunct (currently)
  - Formely: Berkeley, Xerox PARC, DEC SRC
- He has designed systems in these areas
  - Computer architecture, laser printers, local area networks, OS, programming languages, fault-tolerant computing, computer security, text editors
- Honors
  - Member of Nat'l Academy of Engineering
  - Fellow, ACM
  - Fellow, American Academy of Arts and Sciences
  - IEEE Computer Pioneer Award, 1996
  - ACM Turing Award, 1992

Functionality

- The interface allows clients to utilize an implementation
  - Clients ↔ interface ↔ implementation
- Defining interfaces is the most important part of system design
- Lamport’s functionality hints concern interfaces (mostly) and implementations (slightly)
Keep Interfaces Simple

- Do one thing, and do it well
  - Use minimal abstractions
  - Interfaces are contracts for services, cost estimates should be known
  - Example: cost of language constructs
- Get it right
  - Choose the right abstraction!
  - Example: Field lookup in word processor

Corollaries to Simplicity

- Make it fast (rather than general or powerful)
  - Most time is spent in a few operations
  - Dedicate resources to making core operations fast rather than general or powerful
  - Example: RISC
- Don’t hide power
  - Only hide undesirable properties beneath abstractions
- Use procedure arguments (to provide flexibility)
  - Example: Spy from the Berkeley 940 systems
- Leave it to the client
  - Example: client 'hooks' implement functionality in parsers, editors, etc.
Continuity
(i.e., Progress vs. Tradition)

- Keep basic interfaces stable
  - Core interfaces are reflected in far-flung parts of a system, so changes there can involve unseen consequences
  - In a large system, how can you know it is safe to change an interface?
- Keep a place to stand (if you must change)
  - Example: compatibility packages

Making Implementations Work

- Plan to throw one away
  - plan to prototype (because you will anyway)
- Keep secrets
  - The implementation is not a contract
- Divide and conquer
  - Don’t tackle all problems, rather only those you can solve
  - Holds for both design and implementation
- Use a good idea again
  - Generalization is tempting, but not always beneficial
Handling all the cases (i.e., Completeness)

- Handle normal and worst cases separately
  - Normal case must be fast
  - Worst case must handled
  - Examples: caches, fast-path processing

Speed 1 (i.e., How to make a system faster)

- Split resources (unless sharing them can be explicitly justified)
  - Plus: dedicated resources are faster to access
  - Minus: more total resources are often needed
  - Example: special purpose hardware
- Use static analysis
  - Understand your system!
- Cache answers (rather than re-computing the expensive ones)
  - Identify re-use in both software and hardware
Speed 2

- Use hints to speed normal execution
  - Hints can be wrong, need to validate and rollback
  - It must be correct most of the time!
  - Examples: ethernet, branch prediction
- When in doubt, use brute force
  - Easily analysis and modification can enable future improvements
- Use batch processing
  - Let work build up before handling it
  - Example: DMA from a NIC

Speed 3

- Safety first
  - First avoid disaster, then optimize
  - Clever optimizations and variable workloads mix poorly
  - Clever == complex (often) ⇒ not worth it
- Shed load
  - Deny service rather than overload
  - Example: dropping packets in the Arpanet
Fault-tolerance

- **End-to-end**
  - For reliability, errors must always be handled at the application-level (variation: wherever they can be handled completely and reliably)
- **Log updates**
  - In order to understand an object’s history
  - Logs are efficient, and they provide a backup
- **Make actions atomic or re-startable**
  - The key is to know when an operation has completed reliably
  - These are *transactions*

Assignment

- **Thursday (3/25)**
  - Milestone 3
- **Tuesday (3/30):**
  - Work on Milestone 4