Micro-kernels

Presented by Arun Krishnamurthy
COP 5611
University of Central Florida

Outline of Presentation

- Definitions of Kernel and Microkernel
- Microkernel Features
- Chorus - A First Generation Microkernel
- Potential Microkernel Advantages
- First Generation Microkernel Problems
- L4 - A Second Generation Microkernel
- Conclusion
**Definition of Kernel**

- The fundamental part of an Operating System.

- Responsible for providing secure access to the machine's hardware for various programs.

- Responsible for deciding when and how long a program can use a certain hardware (multiplexing).

- Source: Wikipedia.org
Definition of Microkernel

- A kernel technique that provides only the minimum OS services.
  - Address Spacing
  - Inter-process Communication (IPC)
  - Thread Management
  - Unique Identifiers

- All other services are done independently.

Figure 2: Diagram of Microkernel
Address Spaces

- Definition: A mapping which associates each virtual page to a physical page. (Liedtke)

- The microkernel provides 3 operations:
  - Map
  - Grant
  - Flush

AddressSpaces
(Map)

- Adds a page from one address space to another.

Figure 3: Map Example
Address Spaces (Map)

- Adds a page from one address space to another.

Figure 3: Map Example
Address Spaces (Grant)

- Transfers a page from the granter’s address space to the grantee’s.

Figure 4: Grant Example

Address 1

Oops I did it Again
Gangsta Rap
Jailhouse Rock

Address 2

Ghetto SuperStar

Address Spaces (Grant)

- Transfers a page from the granter’s address space to the grantee’s.

Figure 4: Grant Example

Address 1

Oops I did it Again
Gangsta Rap
Jailhouse Rock

Address 2

Ghetto SuperStar

Grant
Gangsta
Rap
Address Spaces (Grant)

- Transfers a page from the granter’s address space to the grantees.

![Grant Example Diagram]

Address Spaces (Flush)

- Deletes the flushed page from all addresses except the flusher’s.

![Flush Example Diagram]
Address Spaces (Flush)

- Deletes the flushed page from all addresses except the flusher’s.

Figure 5: Flush Example
Address Spaces
(Flush)

- Deletes the flushed page from all addresses except the flusher’s.

Inter-process Communication (IPC)

- Definition: Exchange of data between 2 process.
  - IPC is one way communication
  - RPC (remote procedure call) is round trip communication

- The microkernel handles message transfers between threads.

- Grant and Map operations rely on IPC.
IPC Agreement

- The sender decides whether to send information, and what contents are in it.

![Figure 6: IPC Agreement](image-url)


**IPC Agreement**

- The sender decides whether to send information, and what contents are in it.

- The receiver decides whether to receive the contents, and how to interpret it.

![Figure 6: IPC Agreement](image)

**IPC Interrupt Handling**

- Hardware interrupts are done by IPC Messaging.

- The microkernel transfers the interrupts into messages, but does not handle them.

- Instead, the driver software handles them.

```
  driver thread:
  do
    wait for (msg, sender);
    if sender = my hardware interrupt
      then read/write io ports;
      reset hardware interrupt
      else    ...
    fi
  od .
```

![Figure 7: IPC Interrupt Handling](image)
Unique Identifiers (UID)

- The microkernel must supply UIDs for secure and reliable communication.
  - Sender wants to know whether the correct recipient received the message.
  - Receiver wants to know whether the message came from the correct sender.

- Less expensive than cryptography!

First Generation Microkernels

- MACH Kernel
  - 1985 - Carnegie Mellon University
  - Read Mach Lecture Slides for more information

- Chorus Kernel
  - 1987 – Chorus Systems
Chorus System

Chorus Architecture

Figure 8: Chorus Architecture
Chorus Nucleus

- Supervisor
  - Dispatches traps, interrupts, and exceptions delivered by hardware.
- Real Time Executive
  - Controls allocation of processes and provides pre-emptive based scheduling
- Virtual Memory Manager
  - Manipulates VM hardware and memory resources.
- IPC
  - Provides message Exchanging and Remote Procedure Calls (RPC).
Chorus Nucleus Abstractions

- Unique Identifiers – Global Name
- Actors – Resource Allocation
- Threads – Sequential Execution
- Messages – Communication
- Ports – Addressing
- Regions – Structuring

Figure 10: The Chorus Abstractions
Microkernel Advantages

(if implemented properly)

Microkernel Advantages

- Good Flexibility
  - Many applications can be implemented on top of the microkernel.
Microkernel Advantages
(Flexibility)

- Flexible Applications
  - Memory Managers
  - Pagers
  - Multimedia Resource Allocations
  - Device Drivers
  - Second Level Caches/TLBs

- Non-Flexible Applications
  - Processor Architecture
  - Registers
  - First Level Caches/First Level TLBs

Microkernel Advantages

- Good Flexibility
  - Many applications can be implemented on top of the microkernel.

- Good Security
  - Low level user processes = restricted access to system resources.
Microkernel Advantages

- **Good Flexibility**
  - Many applications can be implemented on top of the microkernel.

- **Good Security**
  - Low level user processes = restricted access to system resources.

- **Robustness/Configurability**
  - A problematic application can be reconfigured without rebooting OS.

First Generation Microkernel Problems
First Generation Microkernel Problems

- Expensive Switching Overhead

First Generation Microkernel Problems: Expensive Switching Overhead

- Kernel-User Switches
  - Cost of Kernel Overhead can be up to 800 cycles.
First Generation Microkernel Problems: Expensive Switching Overhead

- **Kernel-User Switches**
  - Cost of Kernel Overhead can be up to 800 cycles.

- **Address Space Switches**
  - Expensive Page Table and Segment Switch Overhead
  - Untagged TLBs = BAD performance

---

<table>
<thead>
<tr>
<th></th>
<th>TLB entries</th>
<th>TLB miss cycles</th>
<th>Page Table switch cycles</th>
<th>Segment cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>486 Pentium</td>
<td>32</td>
<td>9..13</td>
<td>36..364</td>
<td>39</td>
</tr>
<tr>
<td>PowerPC 601</td>
<td>96</td>
<td>9..13</td>
<td>36..1196</td>
<td>15</td>
</tr>
<tr>
<td>Alpha 21064</td>
<td>256</td>
<td>?</td>
<td>?</td>
<td>29</td>
</tr>
<tr>
<td>Mips R4000</td>
<td>40</td>
<td>20..50&lt;sup&gt;a&lt;/sup&gt;</td>
<td>80..1800</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>20..50&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0&lt;sup&gt;b&lt;/sup&gt;</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<sup>a</sup> Alpha and Mips TLB misses are handled by software.
<sup>b</sup>R4000 has a tagged TLB.

*Figure 11: Address Space Switch Overhead Table*
First Generation Microkernel Problems: Expensive Switching Overhead

- **Kernel-User Switches**
  - Cost of Kernel Overhead can be up to 800 cycles.

- **Address Space Switches**
  - Expensive Page Table and Segment Switch Overhead
  - Untagged TLBS = \textit{BAD} performance

- **IPC Cost**
  - First Generation Microkernels IPC required about 115 microseconds.
  - Unix System Call only required 18 microseconds!

Expensive IPC

- 2.1 \mu s, enter/exit kernel
- 18 \mu s
- 2.1 \mu s, enter/exit kernel
- 1 \mu s, min addr-space switch
- 6 \mu s, max addr-space switch
- 1-Byte IPC
- 115 \mu s

Figure 12: IPC Cost Chart
First Generation Microkernel Problems

- Expensive Switching Overhead
- Expensive Memory Overhead

First Generation Microkernel Problems: Expensive Memory Overhead

- Claim (In a 486 – 50MHZ Computer):
  - MACH had noticeably higher Memory Cycle overhead Per Instruction (MIPS) than Untrix (a monolithic kernel).
First Generation Microkernel Problems: Expensive Memory Overhead

Claim (In a 486 – 50MHZ Computer):
- MACH had noticeably higher Memory Cycle overhead Per Instruction (MIPS) than Untrix (a monolithic kernel).

Reason:
- MACH had higher cache working set than Untrix, which produced more capacity misses.
First Generation Microkernel Problems: Expensive Memory Overhead

Figure 4: MCPI Caused by Cache Misses.

First Generation Microkernel Problems

- Expensive Switching Overhead
- Expensive Memory Overhead
- **Lack of Portability**
  - Having portability meant losing performance and flexibility.
  - This also applies to second generation micro-kernels.
WHAT WENT WRONG???

- Don’t blame it on the microkernel logic and ideas…

- …Blame it on POOR construction!!!
  - Many micro-kernels derived from monolithic kernels.

L4 Microkernels

A Second Generation Microkernel
L4 Microkernel

- Developed by Jochen Liedtke in 1995.
  - German National Research Center for IT

- Assumed that micro-kernels were processor dependent.

- Developed from scratch!!!

L4 Abstractions

- Address Spaces
  - Map, Grant, Unmap (Flush)

- Threads

- IPC
  - Short message passing
  - Copying Large Data Messages
  - Lazy Scheduling
L4 Abstractions (IPC)

- Passing Short Messages
  - Transfers short IPC messages in registers.

- Copying Large Data Messages
  - Allow single-copy transfers by sharing the target region with the sender.

- Lazy Scheduling
  - Delay movement between threads until queue is queried.

L4 Abstractions

- Address Spaces
  - Map, Grant, Unmap (Flush)

- Threads

- IPC
  - Short message passing
  - Copying Large Data Messages
  - Lazy Scheduling

- Clans and Chiefs
  - Implementation of Security Policies
L4 Abstractions
(Clan and Chiefs)

- Basic Definitions
  - Chief – Task Creator
  - Clan – All tasks created by their chief.

- Threads can either send IPC to the chief or members of the same clan.

- All messages to different clans are forwarded to the sender clan’s chief.

Figure 15: Clan and Chiefs Diagram
L4 Abstractions

- Address Spaces
  - Map, Grant, Unmap (Flush)
- Threads
- IPC
  - Short message passing
  - Copying Large Data Messages
  - Lazy Scheduling
- Clans and Chiefs
  - Implementation of Security Policies
- UID

L4 Performance Improvements

- L4 Kernel had lower address space IPC time than MACH. (Liedtke – 96)

<table>
<thead>
<tr>
<th></th>
<th>8 Byte IPC</th>
<th>512 Byte IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L4</strong></td>
<td>5 µs</td>
<td>18 µs</td>
</tr>
<tr>
<td><strong>MACH</strong></td>
<td>115 µs</td>
<td>172 µs</td>
</tr>
</tbody>
</table>
L4 Performance Improvements

- L4-Linux RPC had lot lower latency time than MKLinux (based on Mach).

<table>
<thead>
<tr>
<th>System</th>
<th>Latency</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Linux pipe</td>
<td>29 µs</td>
<td>41 MB/s</td>
</tr>
<tr>
<td>(1a) L4Linux pipe</td>
<td>46 µs</td>
<td>40 MB/s</td>
</tr>
<tr>
<td>(1b) L4Linux (trampoline) pipe</td>
<td>36 µs</td>
<td>38 MB/s</td>
</tr>
<tr>
<td>(1c) MKLinux (user) pipe</td>
<td>722 µs</td>
<td>10 MB/s</td>
</tr>
<tr>
<td>(1d) MKLinux (in-kernel) pipe</td>
<td>316 µs</td>
<td>13 MB/s</td>
</tr>
<tr>
<td>(2) L4 pipe</td>
<td>22 µs</td>
<td>48–70 MB/s</td>
</tr>
<tr>
<td>(3) synchronous L4 RPC</td>
<td>5 µs</td>
<td>65–105 MB/s</td>
</tr>
<tr>
<td>(4) synchronous mapping RPC</td>
<td>12 µs</td>
<td>2470–2900 MB/s</td>
</tr>
</tbody>
</table>

Figure 16: RPC Latency Chart

- L4-Linux had lower compile time than MKLinux.

<table>
<thead>
<tr>
<th>System</th>
<th>Compile Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>476 s</td>
</tr>
<tr>
<td>L4Linux</td>
<td>506 s (+6.3%)</td>
</tr>
<tr>
<td>L4Linux (trampo)</td>
<td>509 s (+6.9%)</td>
</tr>
<tr>
<td>MKLinux (kernel)</td>
<td>555 s (+16.6%)</td>
</tr>
<tr>
<td>MKLinux (user)</td>
<td>605 s (+27.1%)</td>
</tr>
</tbody>
</table>

Figure 17: RPC Overhead Chart
Arun’s Final Thoughts

- The microkernel was supposed to provide good flexibility, security and reliability by providing only the minimum services.

- Unfortunately, first generation micro-kernels showed poor performance due to bad construction.

- However, the L4 showed more hope by displaying improved performance.

- More research is necessary to fully understand and judge the microkernel.

Works Cited

(Microkernel Information) - 1


Works Cited
(Microkernel Information) - 2


Works Cited
(Pictures and Diagrams) - 1

- Title Page: Microsoft Clip Arts

- Definition of Kernel: http://www.tldp.org/LDP/sag/html/x123.html

- Definition of Microkernel + Map, Grant, Flush + IPC Agreement: Arun Krishnamurthy


Works Cited
(Pictures and Diagrams) - 2


- Liedtke Photo: www.l4ka.org