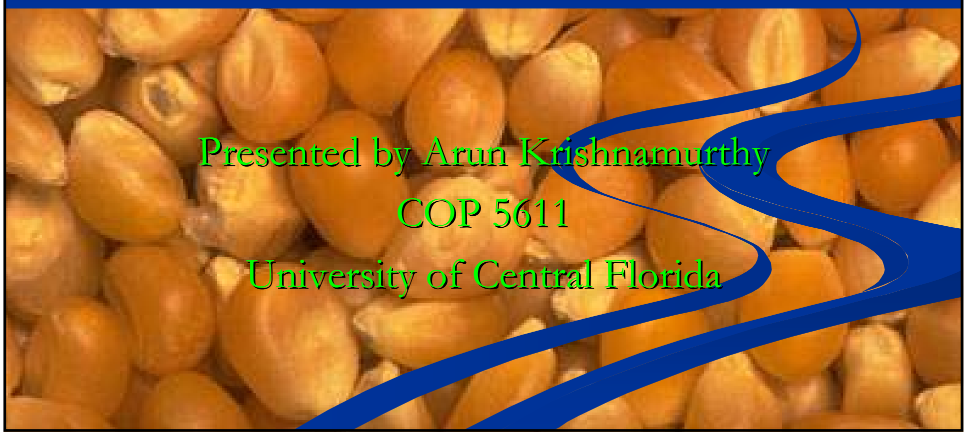
Micro-kernels



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 Kernel

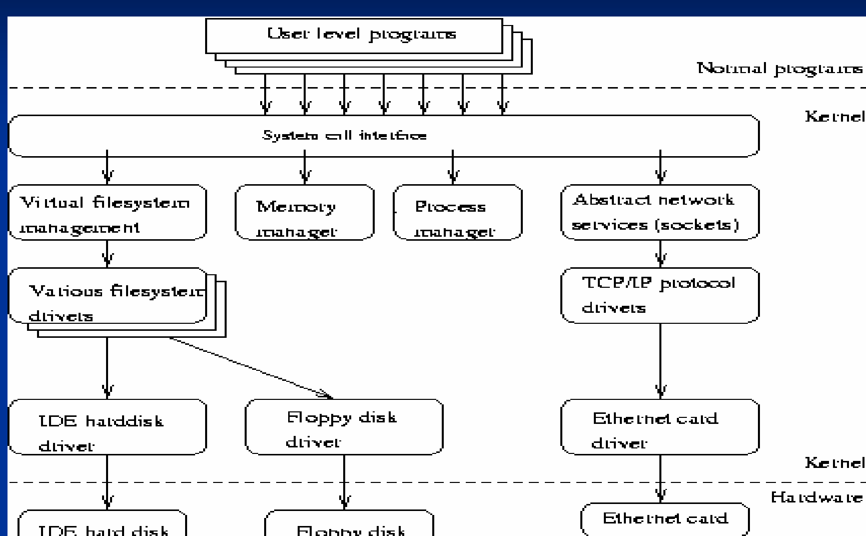


Figure 1: Diagram of Linux Kernel

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.

Definition of Microkernel

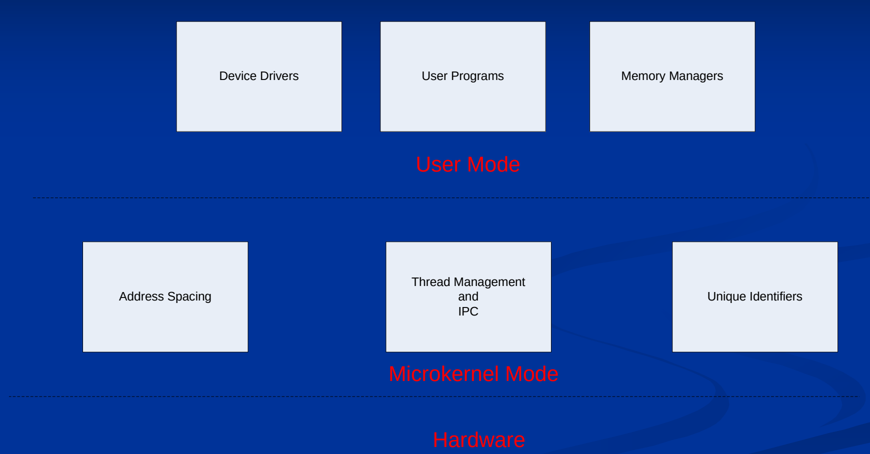


Figure 2: Diagram of Microkernel

Address Spaces

- Definition: A mapping which associates each virtual page to a physical page. (Liedtke)
- The microkernel provides 3 operations:
  - Map
  - Grant
  - Flush

Address Spaces (Map)

- Adds a page from one address space to another.

Figure 3: Map Example

Address Spaces (Map)

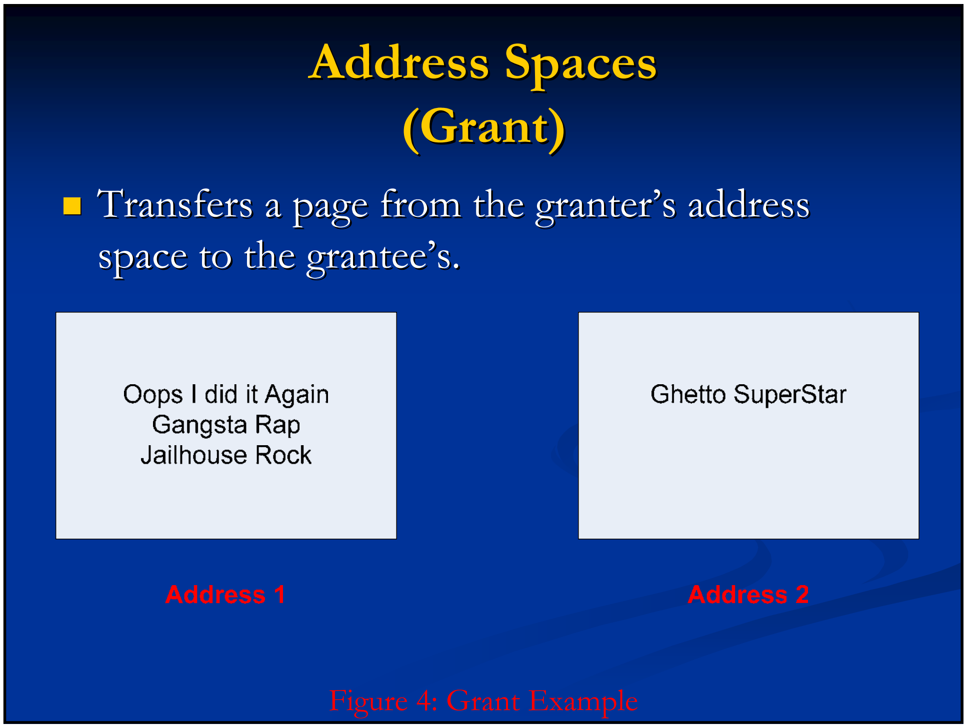
- Adds a page from one address space to another.

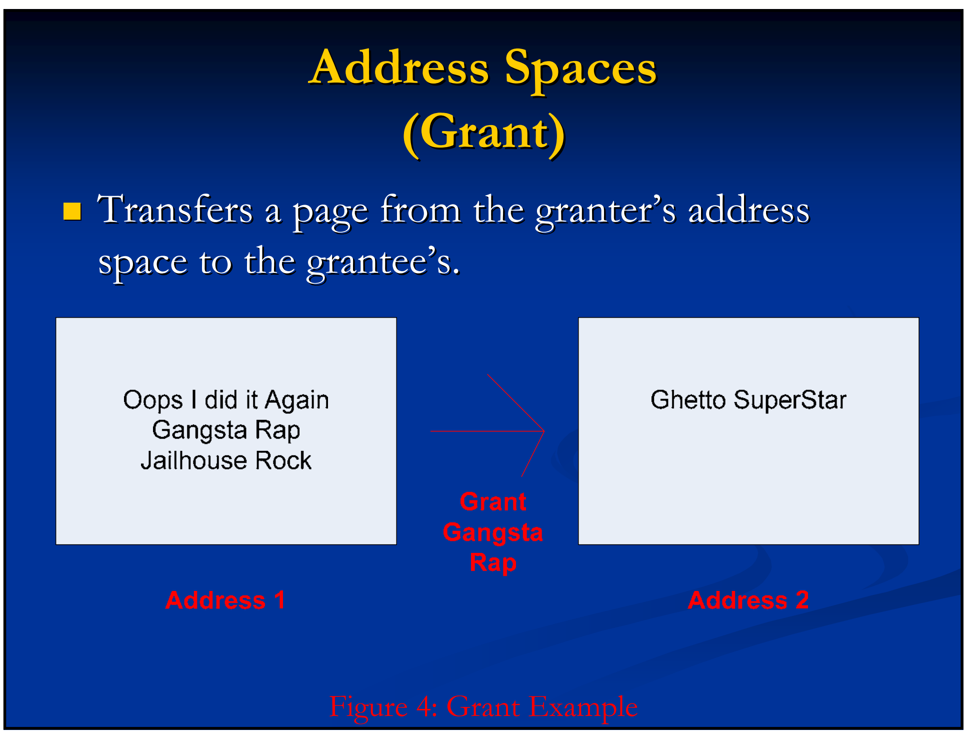
Figure 3: Map Example

Address Spaces (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.
Oops I did it Again
Gangsta Rap
Jailhouse Rock
Ghetto SuperStar
Address 1
Address 2
Figure 4: Grant Example


Address Spaces
(Grant)
Transfers a page from the granter's address space to the grantee's.
Oops I did it Again
Gangsta Rap
Jailhouse Rock
Ghetto SuperStar
Grant
Gangsta
Rap
Address 1
Address 2
Figure 4: Grant Example

Address Spaces (Grant)

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

Figure 4: Grant Example

Address Spaces (Flush)

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

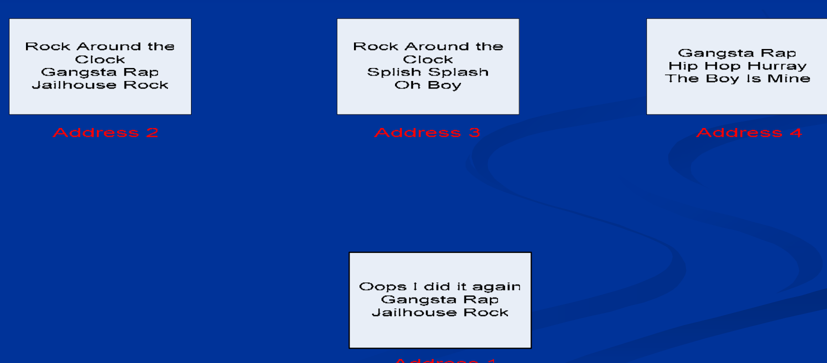


Figure 5: Flush Example

Address Spaces (Flush)

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

Figure 5: Flush Example

Address Spaces (Flush)

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

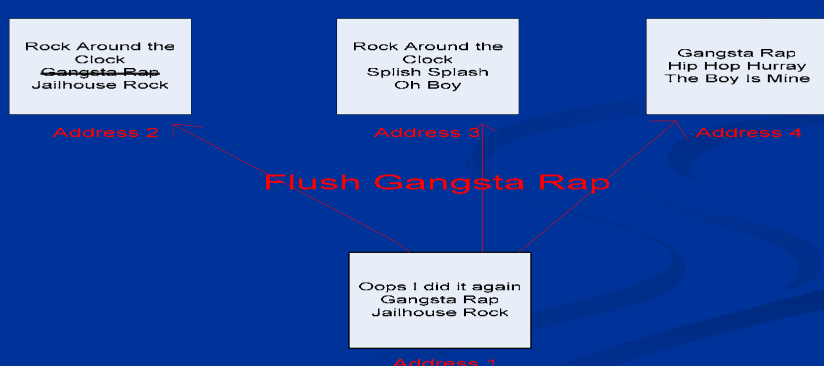


Figure 5: Flush Example

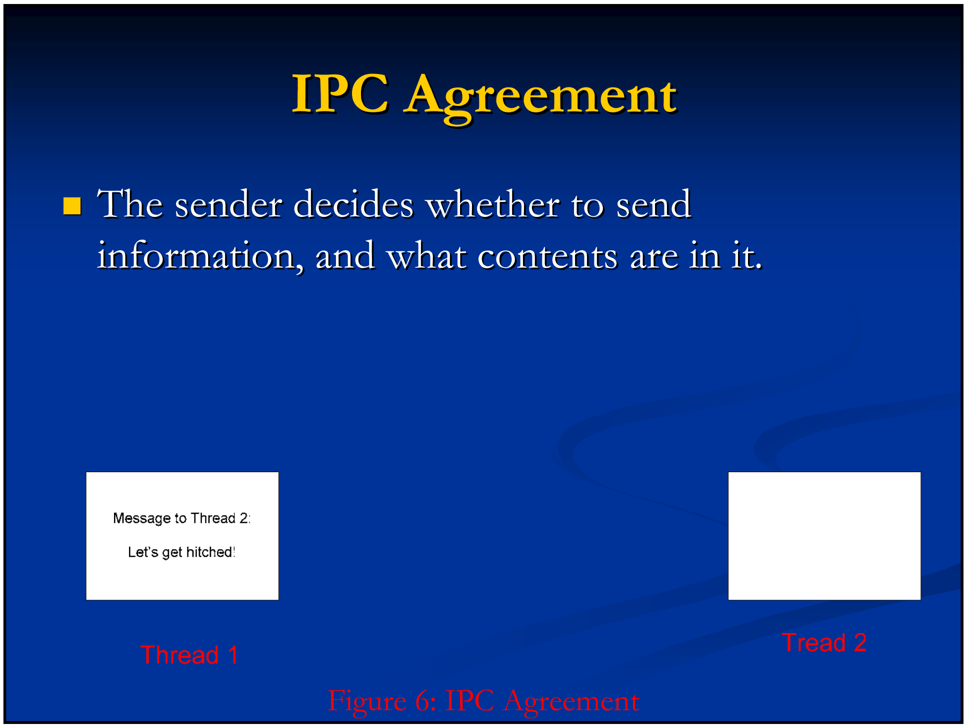
Address Spaces (Flush)

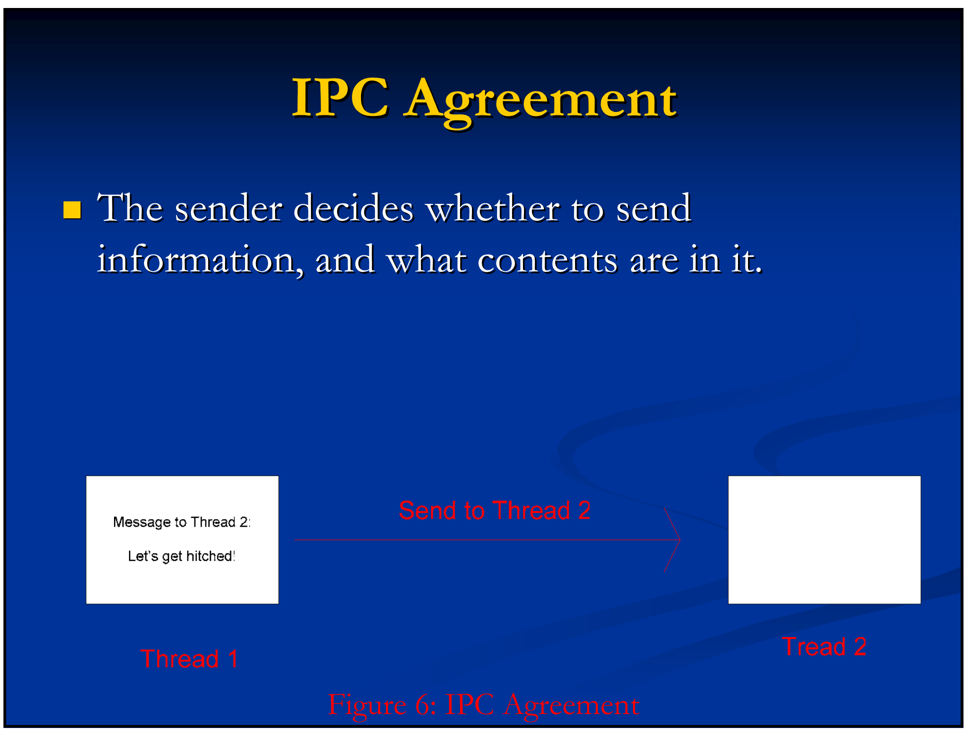
- Deletes the flushed page from all addresses except the flusher’s.

Figure 5: Flush Example

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.
Message to Thread 2:
Let's get hitched!
Thread 1
Tread 2
Figure 6: IPC Agreement


IPC Agreement
The sender decides whether to send information, and what contents are in it.
Message to Thread 2:
Let's get hitched!
Send to Thread 2
Thread 1
Tread 2
Figure 6: IPC Agreement

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.

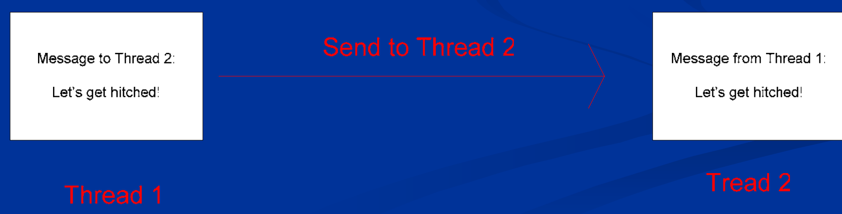


Figure 6: IPC Agreement

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

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

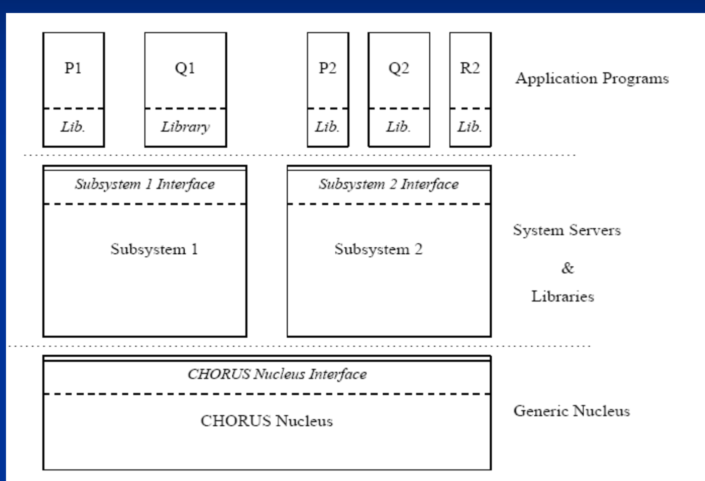


Figure 8: Chorus Architecture

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

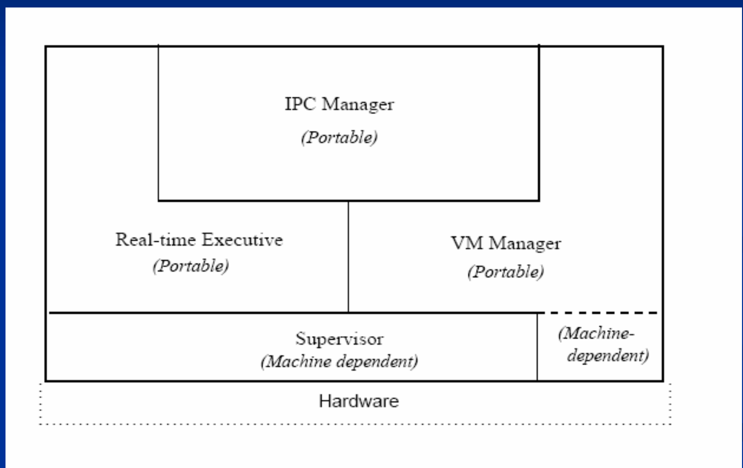


Figure 9: The Chorus Nucleus

Chorus Nucleus Abstractions

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

Chorus Nucleus Abstractions

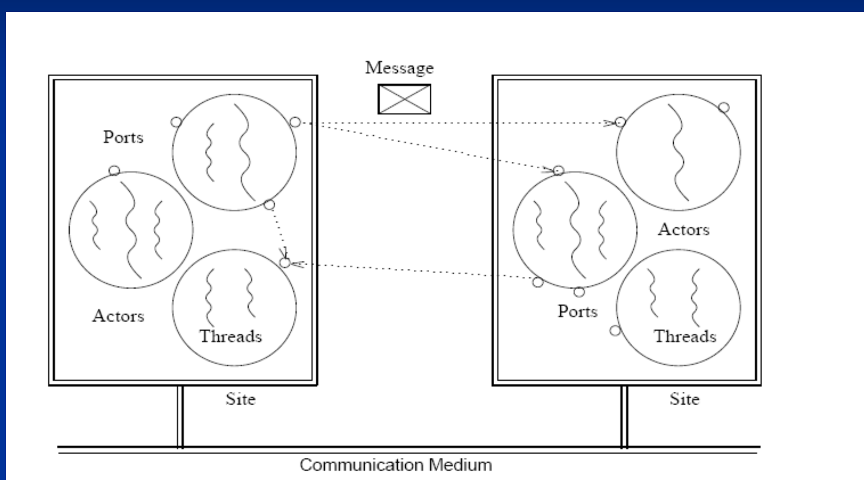


Figure 10: The Chorus Abstractions

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

First Generation Microkernel Problems: Address Space Switches

	TLB entries	TLB miss cycles	Page Table switch cycles	Segment switch cycles
486	32	9...13	36...364	39
Pentium	96	9...13	36...1196	15
PowerPC 601	256	?	?	29
Alpha 21064	40	20...50[a]	80...1800	n/a
Mips R4000	48	20...50[a]	0[b]	n/a

[a] Alpha and Mips TLB misses are handled by software.
[b] 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 = BAD performance
- IPC Cost
  - First Generation Microkernels IPC required about 115 microseconds.
  - Unix System Call only required 18 microseconds!

Expensive IPC

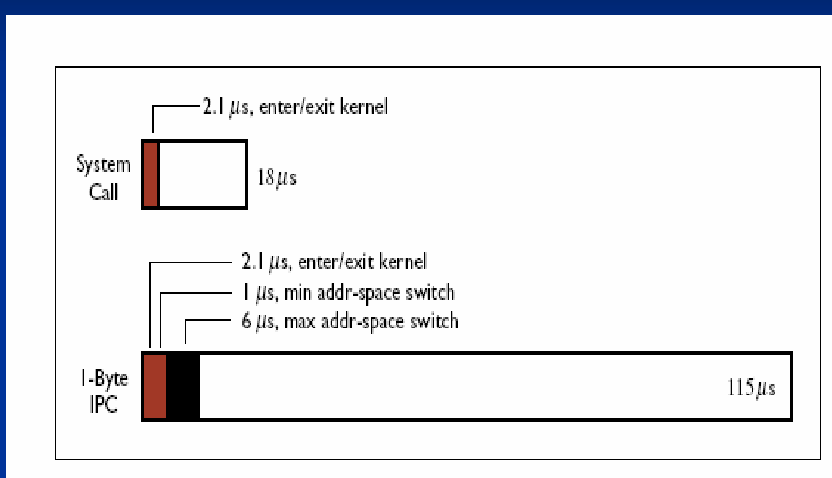


Figure 12: IPC Cost Chart

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

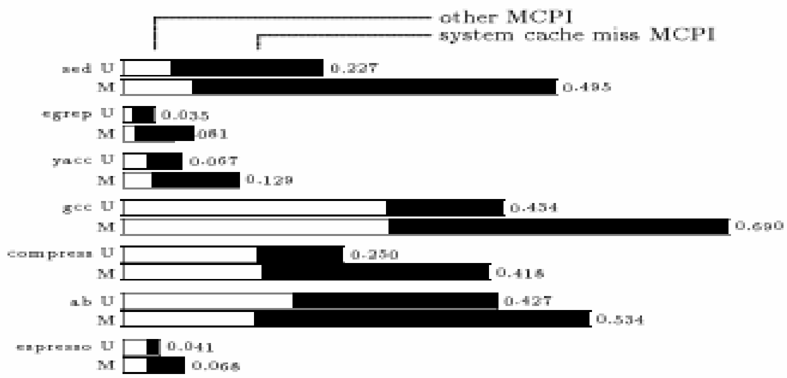


Figure 3: *Baseline MCPI for Ultrix and Mach.*

Figure 13: Baseline MCPI for Ultrix and Mach

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

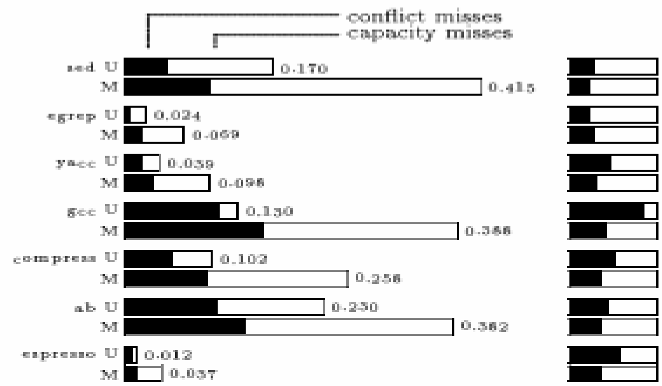


Figure 4: *MCPI Caused by Cache Misses.*

Figure 14: 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.

L4 Abstractions (Clan and Chiefs)

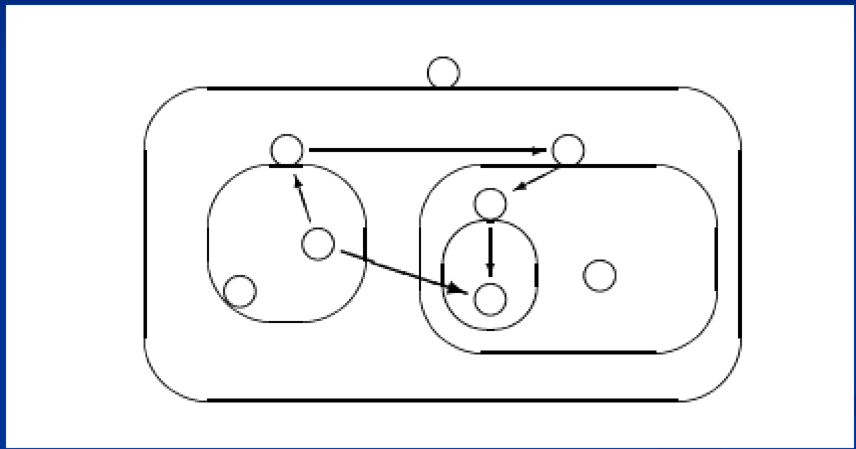


Figure 15: Clan and Chiefs Diagram

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)

	8 Byte IPC	512 Byte IPC
L4	5 μs	18 μs
MACH	115 μs	172 μs

L4 Performance Improvements

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

System	Latency	Bandwidth
(1) Linux pipe	29 μs	41 MB/s
(1a) L^4Linux pipe	46 μs	40 MB/s
(1b) L^4Linux (trampoline) pipe	56 μs	38 MB/s
(1c) MkLinux (user) pipe	722 μs	10 MB/s
(1d) MkLinux (in-kernel) pipe	316 μs	13 MB/s
(2) L4 pipe	22 μs	48–70 MB/s
(3) synchronous L4 RPC	5 μs	65–105 MB/s
(4) synchronous mapping RPC	12 μs	2470–2900 MB/s

Figure 16: RPC Latency Chart

L4 Performance Improvements

- L4-Linux had lower compile time than MKLinux.

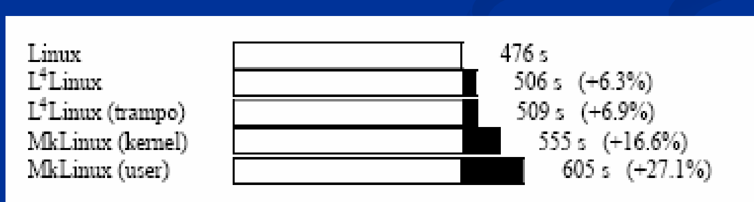


Figure 17: RPC Overhead Chart

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

- Au, Alan and Gernot Heiser. **L4 User Manual**. The University of New South Wales. 1999
- Browne, Christopher. **Microkernel Based OS Efforts.** http://www.cbbrowne.com/info/microkernel.html
- Erlingsson Úlfar and Athanasios Kyparlis. **Microkernels.** http://www.cs.cornell.edu/Info/People/ulfar/ukernel/ukernel.html#current-l4
- Hermann Hartig, Michael Hohmuth, Jochen Liedtke, Sebastian Schonberg, and Jean Wolter. **The Performance of Microkernel Based Systems**. Association of Computing Machinery. 1997

Works Cited (Microkernel Information) - 2

- Liedtke, Jochen. **On Microkernel Construction**. Association of Computing Machinery. 1995
- Liedtke, Jochen. **Towards Real Microkernels**. Association of Computing Machinery. 1996
- M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Hermann, C. Kaiser, S. Langlois, P. Leonard, and W. Neuhauser. **Overview of The Chorus Distributed Operating System**. Chorus Systems. 1991
- **Wikipedia Encyclopedia**. www.wikipedia.org

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
- Chorus Diagrams: M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien,
- M. Guillemont, F. Hermann, C. Kaiser, S. Langlois, P. Leonard, and W. Neuhauser. **Overview of The Chorus Distributed Operating System**. Chorus Systems. 1991
- Performance Issues + Driver Thread: Liedtke, Jochen. **On Microkernel Construction**. Association of Computing Machinery. 1995

Works Cited (Pictures and Diagrams) - 2

- Performance Issues: Liedtke, Jochen. **Towards Real Microkernels**. Association of Computing Machinery. 1996
- L4 Performance Improvements: Hermann Hartig, Michael Hohmuth, Jochen Liedtke, Sebastian Schonberg, and Jean Wolter. **The Performance of Microkernel Based Systems**. Association of Computing Machinery. 1997
- Liedtke Photo: www.l4ka.org