SYSTEMS GOSSIP MEETUP

Learning large systems using peer-to-peer gossip

Policy Against Harassment at ACM Activities

https://www.acm.org/about-acm/policy-against-harassment

OS Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM's policy against harassment. Unacceptable behavior will not be tolerated.

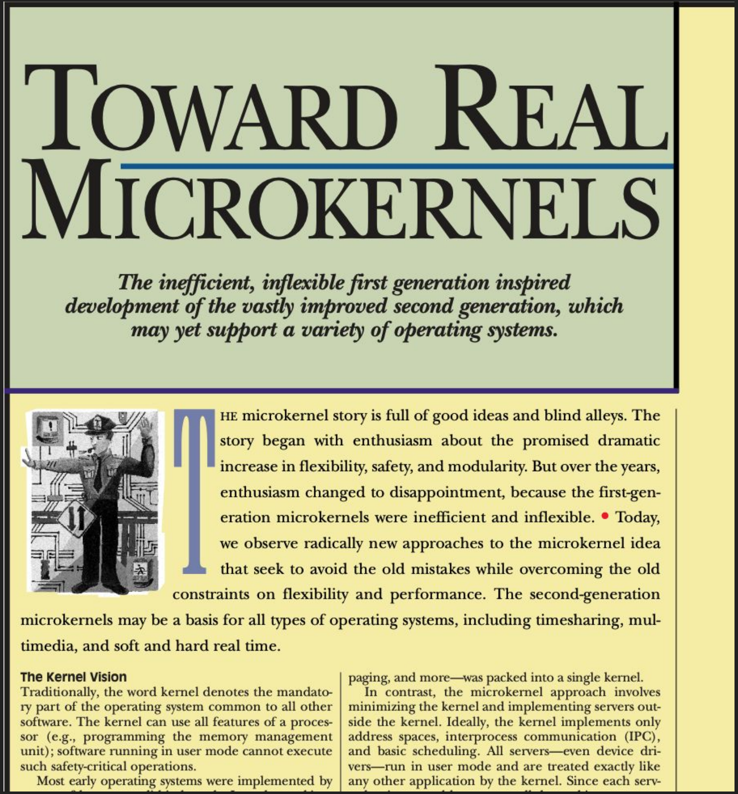
L4 Microkernels

Today

- What is microkernel?
- Why is it important?
- Microkernel Design
- Microkernel Implementation: L4
- Conclusion

What is Microkernel

- Move most operating system kernel functionality to user-space processes
- Minimal kernel
  - Address space
  - Threads
  - IPC (inter-process communication)
- 1980s hot research topic
- Examples: CMU's Mach, L4



TOWARD REAL MICROKERNELS

The inefficient, inflexible first generation inspired development of the vastly improved second generation, which may yet support a variety of operating systems.

THE microkernel story is full of good ideas and blind alleys. The story began with enthusiasm about the promised dramatic increase in flexibility, safety, and modularity. But over the years, enthusiasm changed to disappointment, because the first-generation microkernels were inefficient and inflexible. • Today, we observe radically new approaches to the microkernel idea that seek to avoid the old mistakes while overcoming the old constraints on flexibility and performance. The second-generation microkernels may be a basis for all types of operating systems, including timesharing, multimedia, and soft and hard real time.

The Kernel Vision

Traditionally, the word kernel denotes the mandatory part of the operating system common to all other software. The kernel can use all features of a processor (e.g., programming the memory management unit); software running in user mode cannot execute such safety-critical operations.

Most early operating systems were implemented by

paging, and more—was packed into a single kernel.

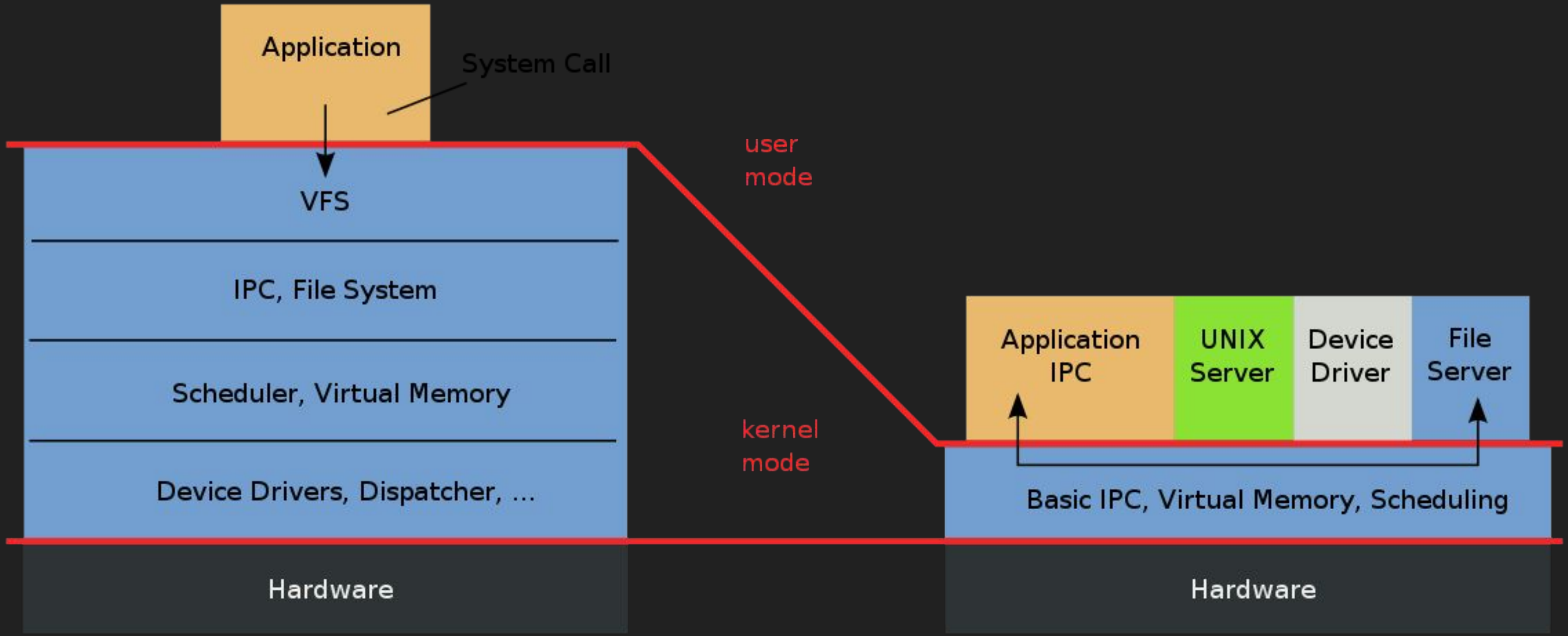
In contrast, the microkernel approach involves minimizing the kernel and implementing servers outside the kernel. Ideally, the kernel implements only address spaces, interprocess communication (IPC), and basic scheduling. All servers—even device drivers—run in user mode and are treated exactly like any other application by the kernel. Since each serv-

Why is it important?

- Elegant and clean
    - Minimal trusted computing base
    - Smaller attack surface
- Easy to prove
- More secure
- Easy to optimize
- More robust and extensible
- Inspires VM designs

Monolithic Kernel based Operating System

Microkernel based Operating System



https://en.wikipedia.org/wiki/Microkernel

Microkernel Evolution

First gen (Mach ‘87)

- Bad performance
- 180 syscalls
- 100 kLOC
- 100 us IPC

Improving IPC by Kernel Design

Jochen Liedtke
Proceeding of the 14th ACM Symposium on Operating Systems Principles
Asheville, North Carolina
1993

Microkernel Evolution

First gen (Mach ‘87)

- Bad performance
- 180 syscalls
- 100 kLOC
- 100 us IPC

Sec gen (L4 ‘95)

- Great perf
- 7 syscalls
- 10 kLOC
- 1 us IPC
- Used in iPhone, iPad, Mac etc.

Third gen (seL4 ‘09)

- Verifiable
- 3 syscalls
- 9 kLOC
- 0.1 us IPC

https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web
https://trustworthy.systems/publications/nicta_full_text/8988.pdf

L4 Basic Abstraction

Task

Threads: each has a global ID

Own address space

Message

From thread ID
To thread ID
Data

Microkernel L4

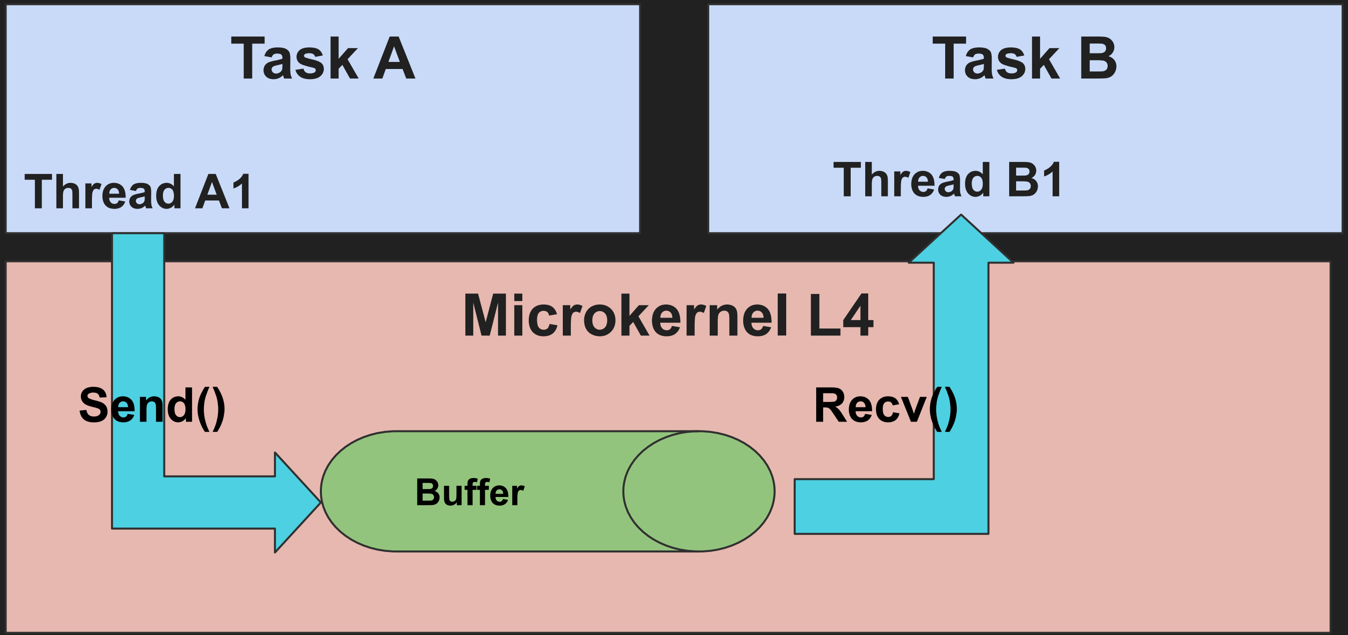
Manage tasks, context switch, IPC (sending messages between tasks)

L4 System Calls

1. ipc()
    a. Message passing, combing sending and receiving operations
2. fpage_unmap()
    a. Revoke mappings
3. task_new()
    a. create/delete task/address space
4. lthread_ex_regs()
    a. create/manipulate thread

Problem: IPC Performance

Observation: microkernel programs do lots of IPC!

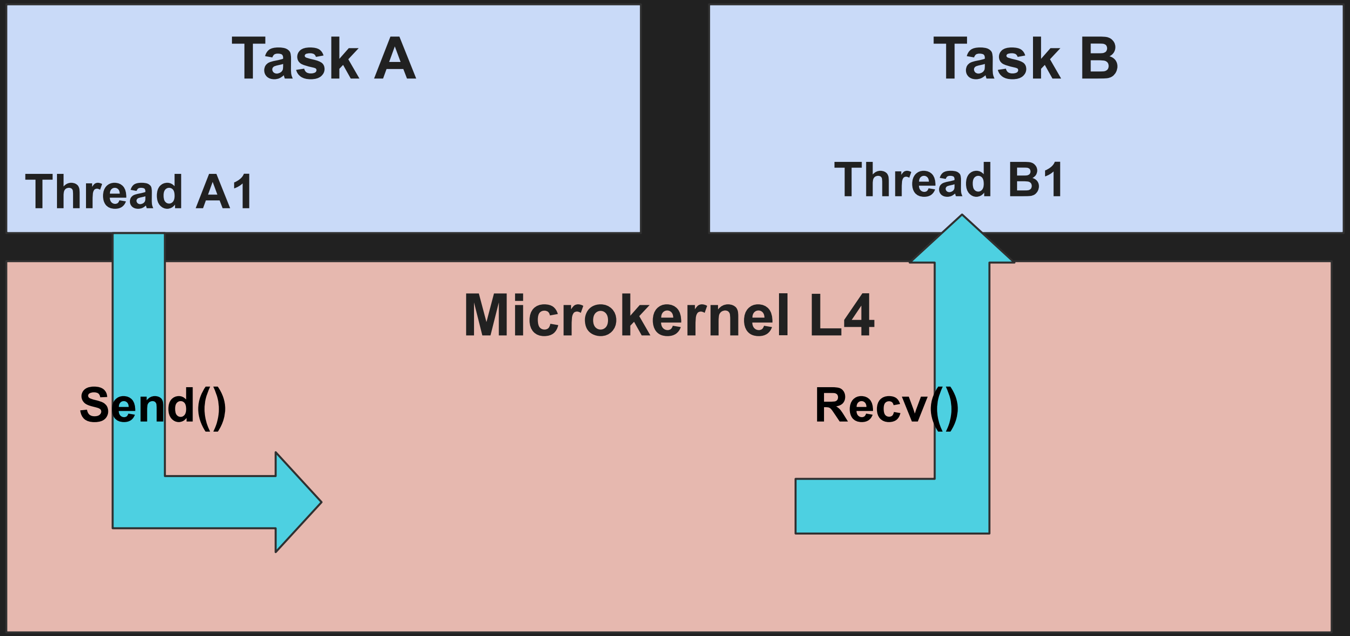


Problem: IPC Performance

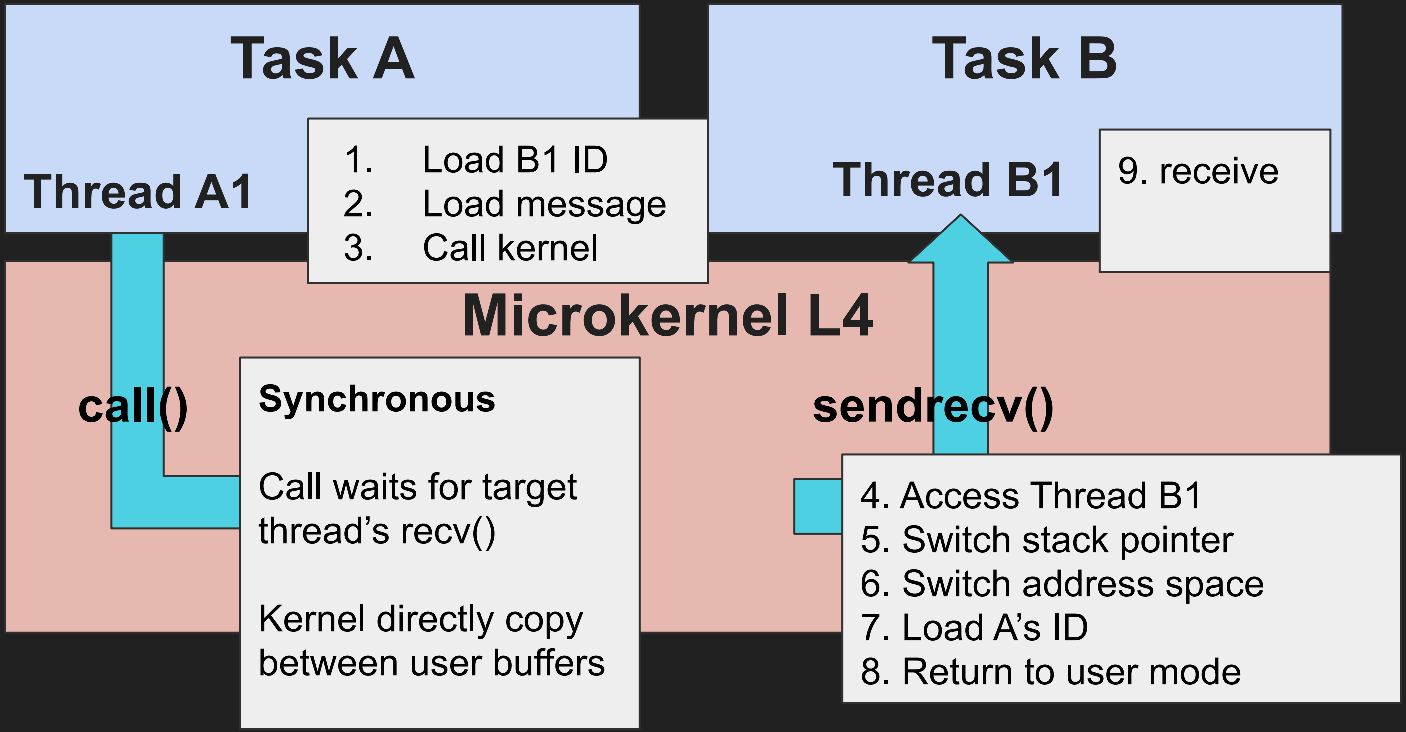
With the usual request-response pattern (RPC)

1. 4 system calls
   a. send() -> recv()
   b. recv() -> send()
2. Each system call may disturb CPU's caches
3. Four message copies (two for request, two for reply)
4. Two context switches, two schedulings

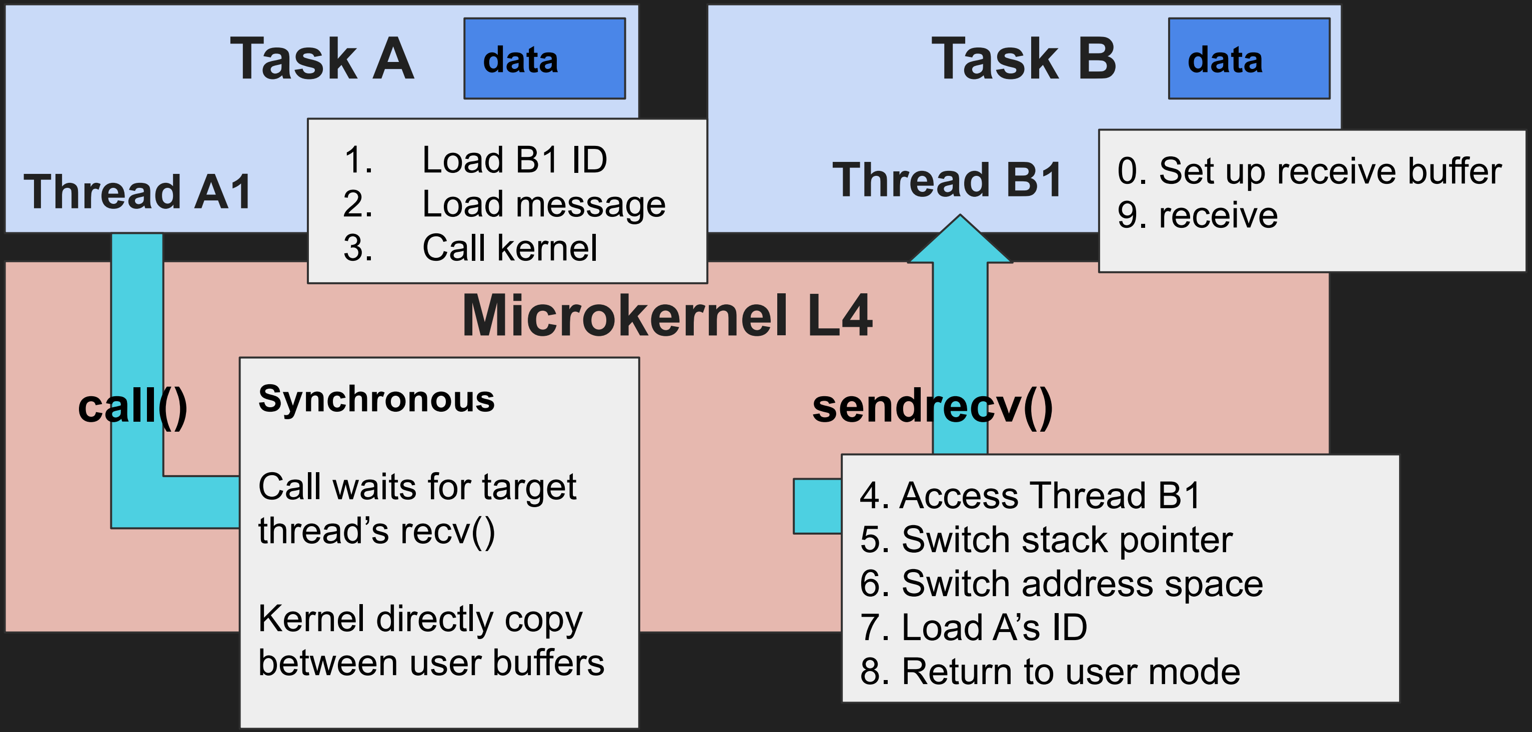
Improving IPC by Kernel Design



Improving IPC by Kernel Design



Improving IPC by Kernel Design



Improving IPC by Kernel Design

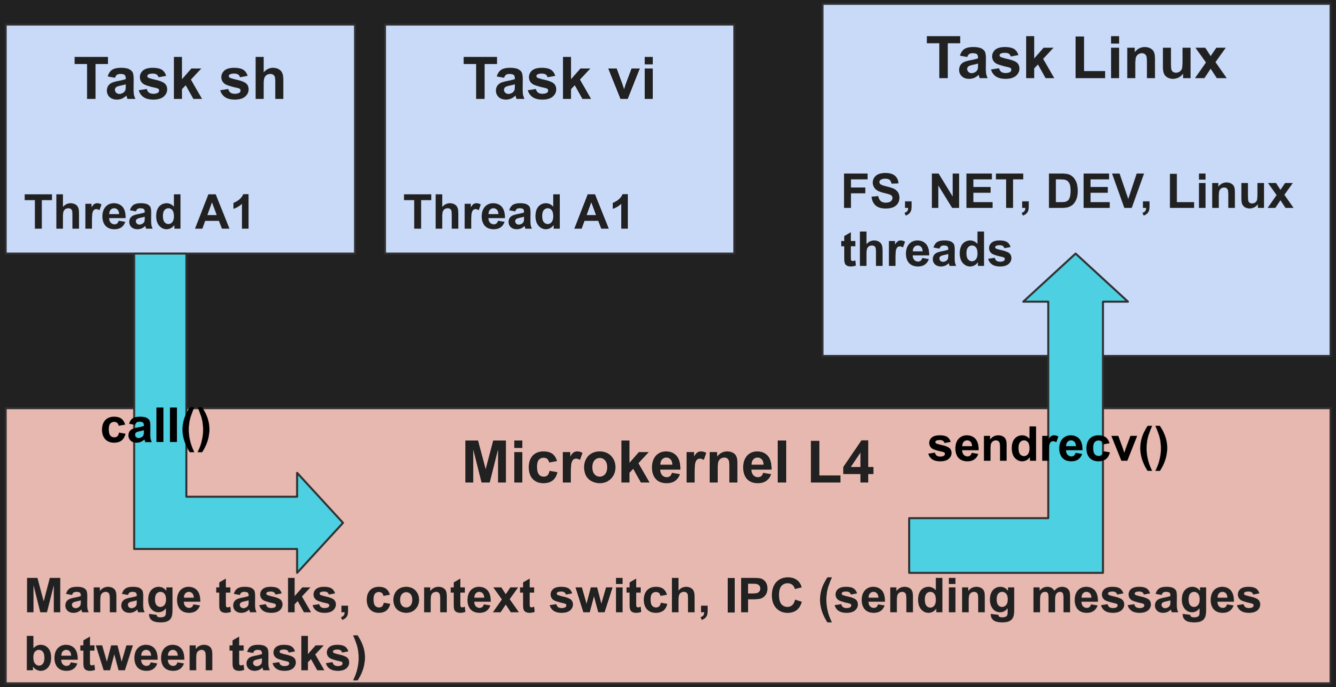
Result: 20x reduction in IPC cost

2x reduction in user/kernel crossings

Makes microkernel performance comparable to monolithic kernel

Today's paper

The performance of microkernel-based systems



Today's paper

System	Time	Cycles
Linux	1.68 μs	223
L^4Linux	3.95 μs	526
L^4Linux (trampoline)	5.66 μs	753
MkLinux in-kernel	15.41 μs	2050
MkLinux user	110.60 μs	14710

Table 2: *`getpid` system-call costs on the different implementations.* (133 MHz Pentium)

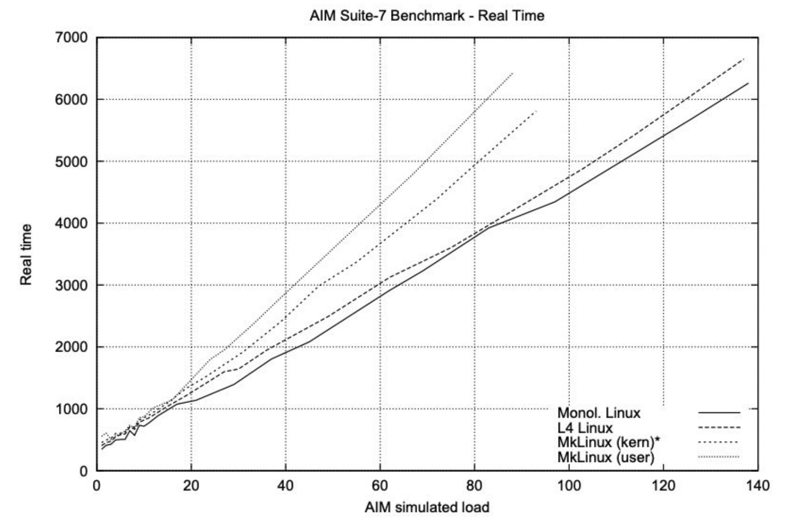


Figure 8: *AIM Multiuser Benchmark Suite VII.* Real time per benchmark run depending on AIM load units. (133 MHz Pentium)

Conclusion

What’s current situation for microkernel?

1. Used often in embedded computing. Apple “enclave” processor.
2. Never caught on for general computing
3. The OS server idea inspires Virtual Machines (VM).

Reference

1. https://www.youtube.com/watch?v=dM9PLdaTpnA
2. https://www.cs.ubc.ca/~norm/508/2007W1/summaries/paper15/index.html
3. https://www.youtube.com/watch?v=mRr1lCJse_I
4. https://www.youtube.com/watch?v=oPjwyykO1bI
5. https://pdos.csail.mit.edu/6.S081/2021/lec/l-organization.txt
6. https://trustworthy.systems/publications/nicta_full_text/8988.pdf
7. https://dl.acm.org/doi/10.1145/168619.168633