Exokernels: Extensible Kernels

Salman Abid

September 19th, 2024

(Based on slides from Edgar Velázquez-Armendáriz and Dawson Engler)

Agenda

- Exokernel: An Operating System architecture for Application-Level Resource Management
- Extensibility, Safety and Performance in the SPIN Operating System

Agenda

- Exokernel: An Operating System architecture for Application-Level Resource Management
- Extensibility, Safety and Performance in the SPIN Operating System

Timeline

- 1981: **Operating system support for database management** by Michael Stonebraker
- 1980s: **Mach** by Rashid et al.
- 1990s:
  - **SPIN**
  - **Exokernel**
  - **Disco** (VM)
- 2000s:
  - **Xen**

Why an exokernel?

- **Exterminate All Operating System Abstractions** (HotOS ‘95) by Dawson Engler and Frans Kaashoek
- Traditional OS multiplexes and *abstracts* physical resources
- Root of all (OS) problems
- Abstractions are good, just not in the kernel

Why an exokernel?

- Traditional OS software is generic… but opinionated
  - OS abstractions preempt application design decisions
- Result: Applications are slow, or can’t be written
- Embody the “end-to-end” argument as much as possible

Context

- Written by Dawson R. Engler, M. Frans Kaashoek and James O'Toole Jr.
- Engler's thesis for his MS.
- Followed by 'Application Performance and Flexibility on Exokernel Systems' in SOSP '97

Overview

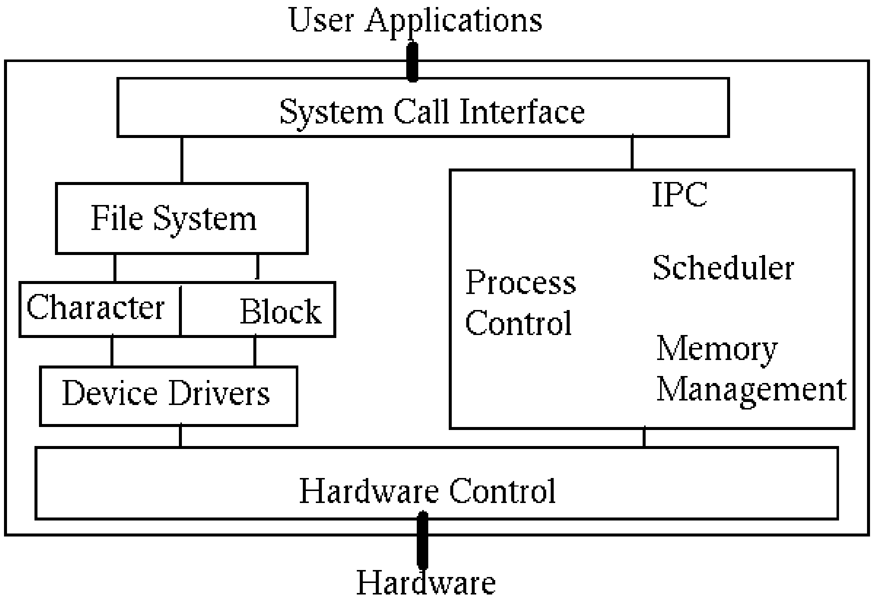
Kernel:

- Separate hardware access from resource management
    - Share resources, not policies

Library Operating System:

- Implement high-level abstractions (e.g. IPC, Scheduling, VM)
- These abstractions are *extensible*

Traditional OS Architecture



Exokernel Architecture

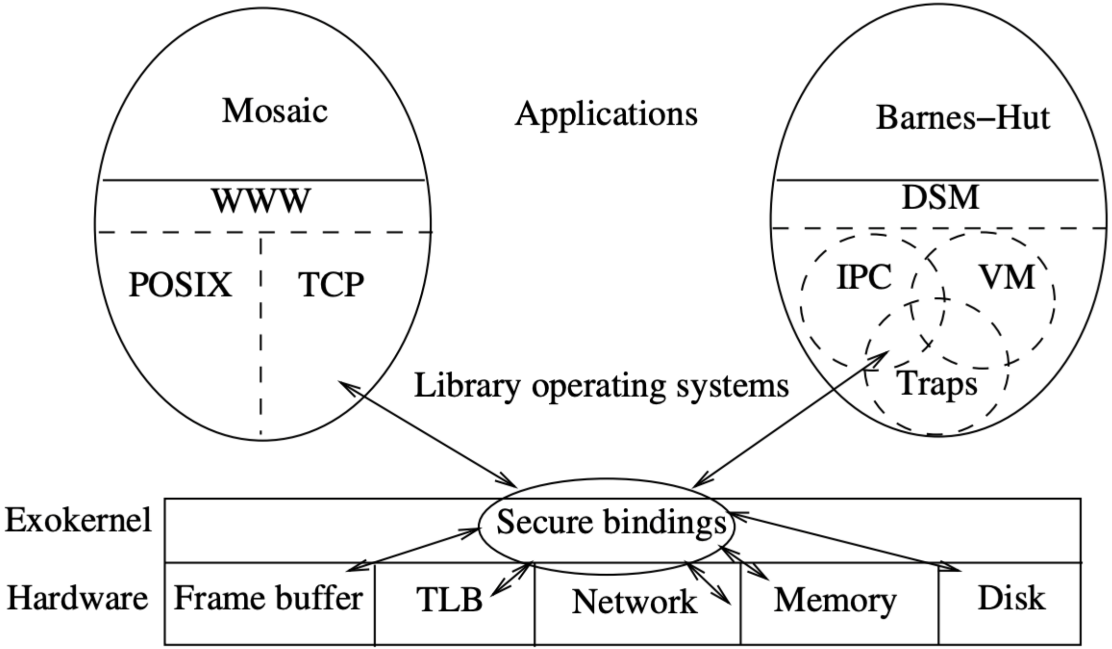
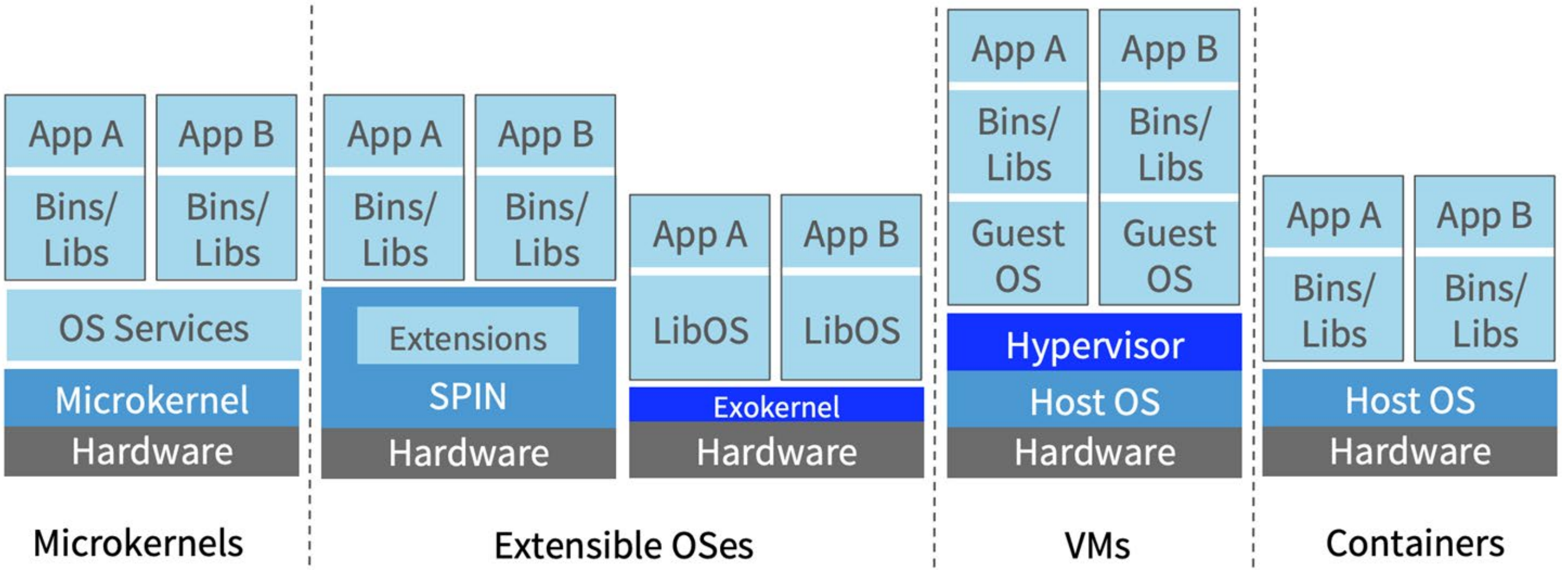


Fig 1 from SOSP '95

Exokernel vs Microkernels vs VM

- Exokernel defines only a low-level interface
- A microkernel also runs almost everything on user-level, but has fixed abstractions
- A VM emulates the whole machine, doesn’t provide direct access

Exokernel vs Microkernels vs VM



From CS262a Fall 2016, UC Berkeley

Discussion

- How compatible is the exokernel architecture with existing OS design?
- What happens to executables?
  (Consider size, library dependencies, portability)
- The authors claim that OS experiments are easier in the user-space. How would that scale commercially? Do they want to make kernel hackers out of application developers?

Exokernel: Design

- Kernel multiplexes hardware
  - Doesn’t govern authorization
- Maximise application control
  - Expose allocation and kernel data structures
  - Expose names
  - Access to hardware
- Revoke resources from applications
  1. “Request” access to resource
  2. Repossess resource if unresponsive

Exokernel: Memory

- Guard physical memory access
  - Examine Lib. OS capability before granting access
- Software TLB caches access bindings
  - Large TLB; improves performance
- “Read-only” access to page table for applications
  - Applications can share resources easily
- To break binding: flush TLB, dequeue DMA requests

Exokernel: Scheduling

- Round-robin scheduling
- Library OS implements context-switching
- The kernel is unforgiving; applications that take too much time are killed

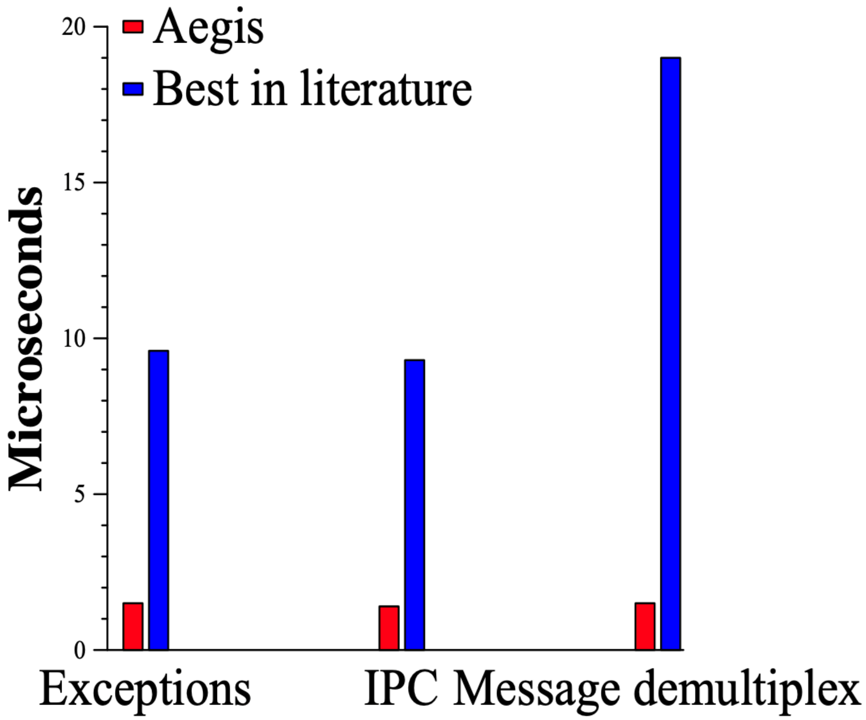
Exokernel: Network

- Dynamic Packet Filters
- Fast de-multiplexing of traffic
- Code can be ‘downloaded’ into kernel
    - Application-specific Safe Handlers (ASH)

Exokernel: Implementation

- On MIPS-based DECstations
- Aegis: an exokernel in practice
  - Physical memory
  - TLB entries
  - Time slices
  - Network
- ExOS: Rudimentary UNIX-like library
  (Processes, Virtual Memory, Network protocols)

Results on DEC5000/25Mhz



From Engler's talk at SOSP '95

Microbenchmarks

Machine	OS	pipe	pipe'	shm	lrpc
DEC2100	Ultrix	326.0	n/a	187.0	n/a
DEC2100	ExOS	30.9	24.8	12.4	13.9
DEC3100	Ultrix	243.0	n/a	139.0	n/a
DEC3100	ExOS	22.6	18.6	9.3	10.4
DEC5000	Ultrix	199.0	n/a	118.0	n/a
DEC5000	ExOS	14.2	10.7	5.7	6.3

Machine	OS	Roundtrip latency
DEC5000/125	ExOS/ASH	259
DEC5000/125	ExOS	320
DEC5000/125	Ultrix	3400
DEC5000/200	Ultrix/FRPC	340

Discussion

- We see clear examples of Exokernel being more performant, but little in the way of commercial adoption. Why is that?
- What ideas from extensibility, if not the kernel itself, are part of modern systems?

Agenda

- Exokernel: An Operating System architecture for Application-Level Resource Management
- Extensibility, Safety and Performance in the SPIN Operating System

SPIN

- University of Washington.
- Brian N. Bershad, Stefan Savage, Emin Gun Sirer, Marc E. Fiuczynski, David Becker, Craig Chambers, Susan Eggers
- 1997 IEEE Symposium on Security and Privacy

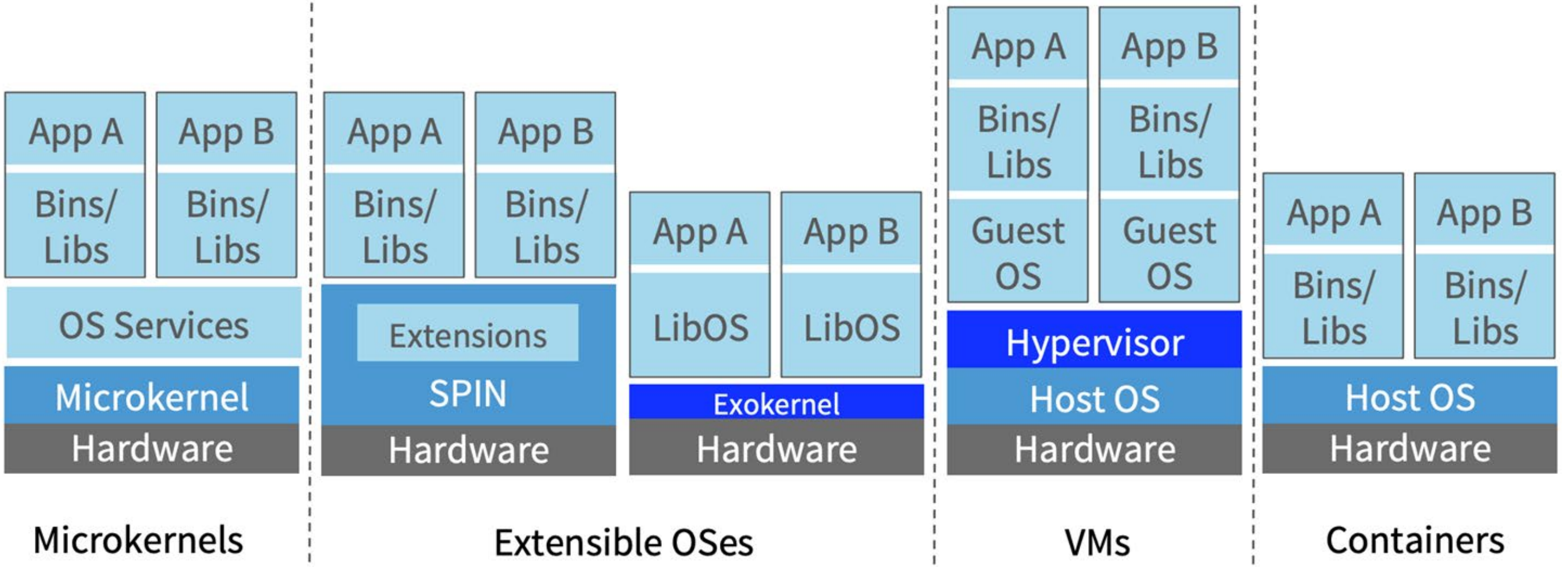
SPIN

- Similar ideas about extensibility
- **Extensibility**: Extensions dynamically linked and bound at runtime
- **Safety**:
  - Written in Modula-3
  - Statically verified. Type safe.
- **Performance**:
  - Extensions run inside the kernel. No overhead for traps.

SPIN vs Exokernel

- SPIN relies on safety mechanisms of the Modula-3 language
- Extensions execute in response to system events (e.g. page fault, thread scheduling)

In a nutshell



From CS262a Fall 2016, UC Berkeley

To summarize…

- Exokernels are fast and simple
- OS abstractions can be implemented as libraries
- Extensiblity in the kernel gives significant performance benefits