11/11 CS240 - Exokernels

Announcements

For next class (Thursday 11/13)

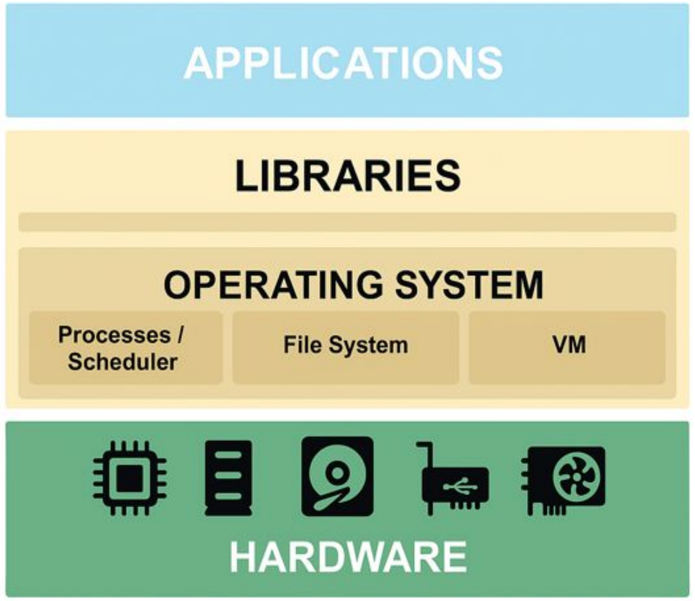
1. Read: Dune: Safe User-level Access to Privileged CPU Features
2. Submit answers to reading questions (see course schedule) before class

Paper

- Application Performance and Flexibility on Exokernel Systems
  - SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles
  - Two future Stanford faculty: Dawson Engler & David Mazières
  - Follow up paper with more experience and evaluation
- Paper reports experience with building Exokernels
  - Radical idea explored: OS abstractions consider harmful
- Idea itself failed to catch on but techniques developed in use did
  - Download code, Wakeup predicates (Linux eBPF, `io_uring`, and `userfaultfd`)

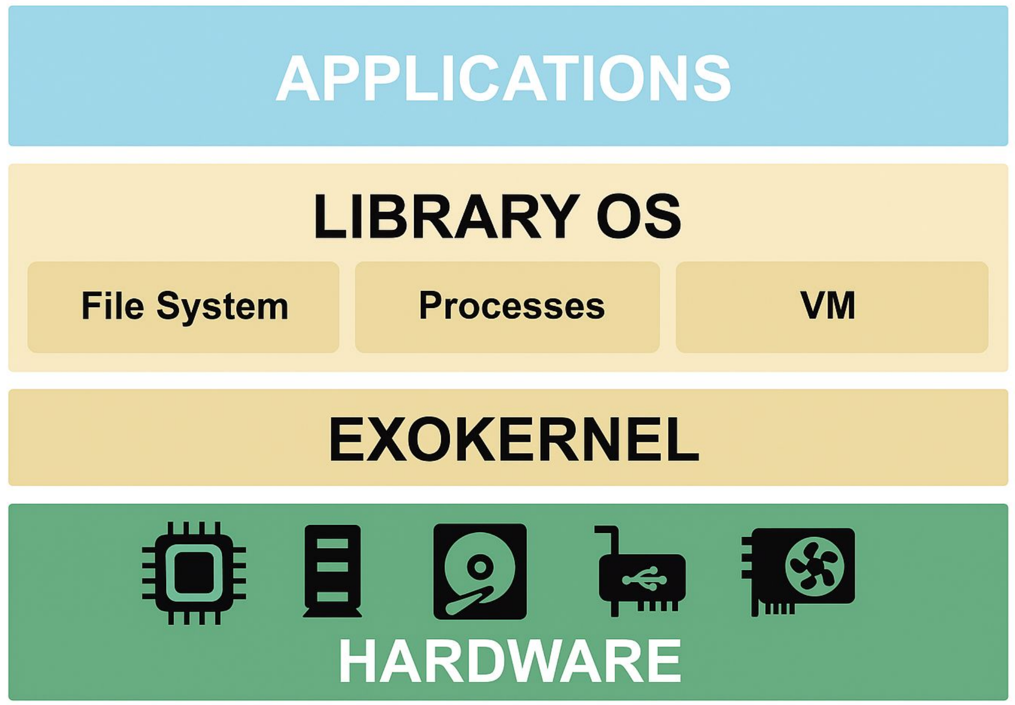
Operating System Structure

- What do typical operating system kernels do?
- According to the paper, what's wrong with this picture?
- How do libraries fit into this?
  - Does code run faster in a library or kernel?
- Give DBMS example - bypass



Exokernel principles

- Separate protection from management
- Expose names
- Expose revocation
- Expose information



Exokernel kernel interface

- Interface controls access to low level resources using high-level abstraction?
- Xok design techniques:
    - Same access control across all resources?
    - Software abstractions to bind hardware resources?
    - Code download?

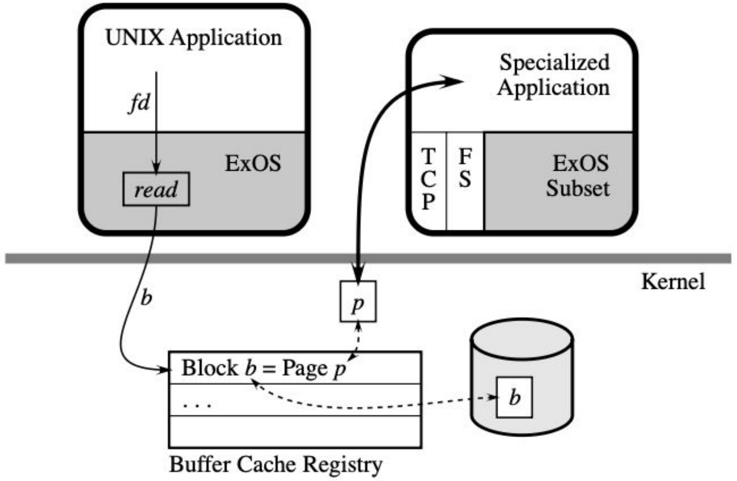
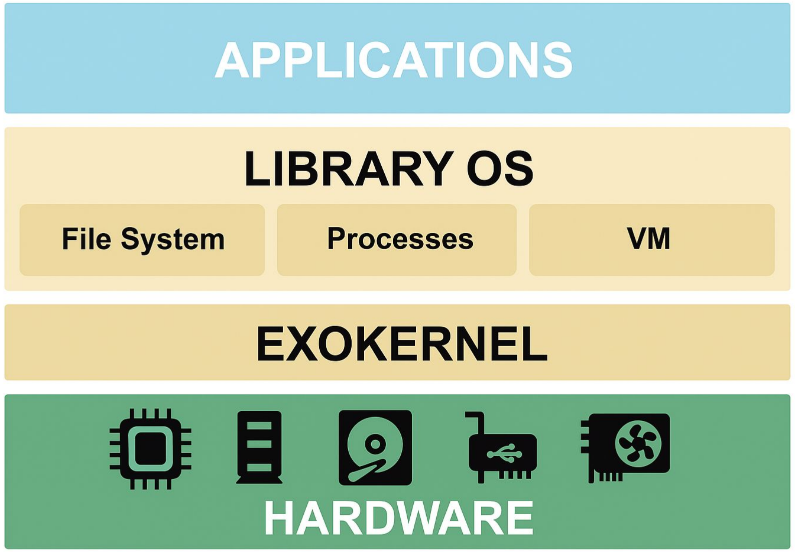


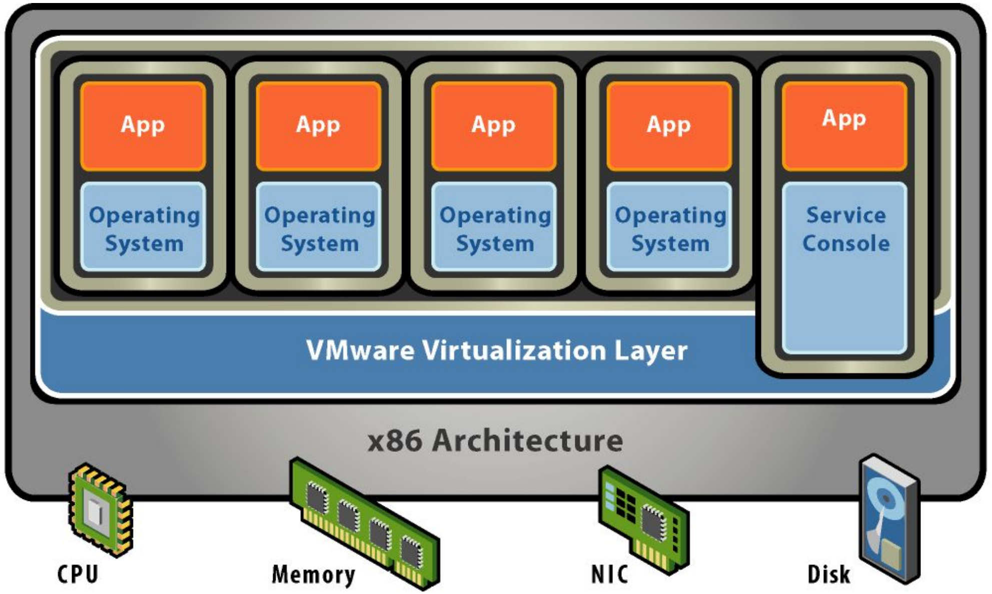
Figure 1: A simplified exokernel system with two applications, each linked with its own libOS and sharing pages through a buffer cache registry.

Protected Sharing

- Trust model
  - Application <-> Application's libOS
  - libOSes <-> Xok
  - Application A libOS <-> Application B libOS
- Mechanisms:
  - Software regions
  - Capabilities
  - Wakeup predicates
  - Robust critical sections
- Trust optimization:
  - Mutual trust (ptrace?)
  - Unidirectional trust
  - Mutual distrust

How does Xok compare with VMM like VMware?





XN - Multiplexing Stable Storage

- Untrusted deterministic functions (UDFs)?
  - Templates *owns-udf*$_T$()
- Ordered disk writes
  - 3 ordering rules make crash recover fast
    - Nullify all points before reusing a block
    - Don't create pointer before initialized block
    - Don't old pointer in persistent storage before the new one has been set
- Buffer cache registry

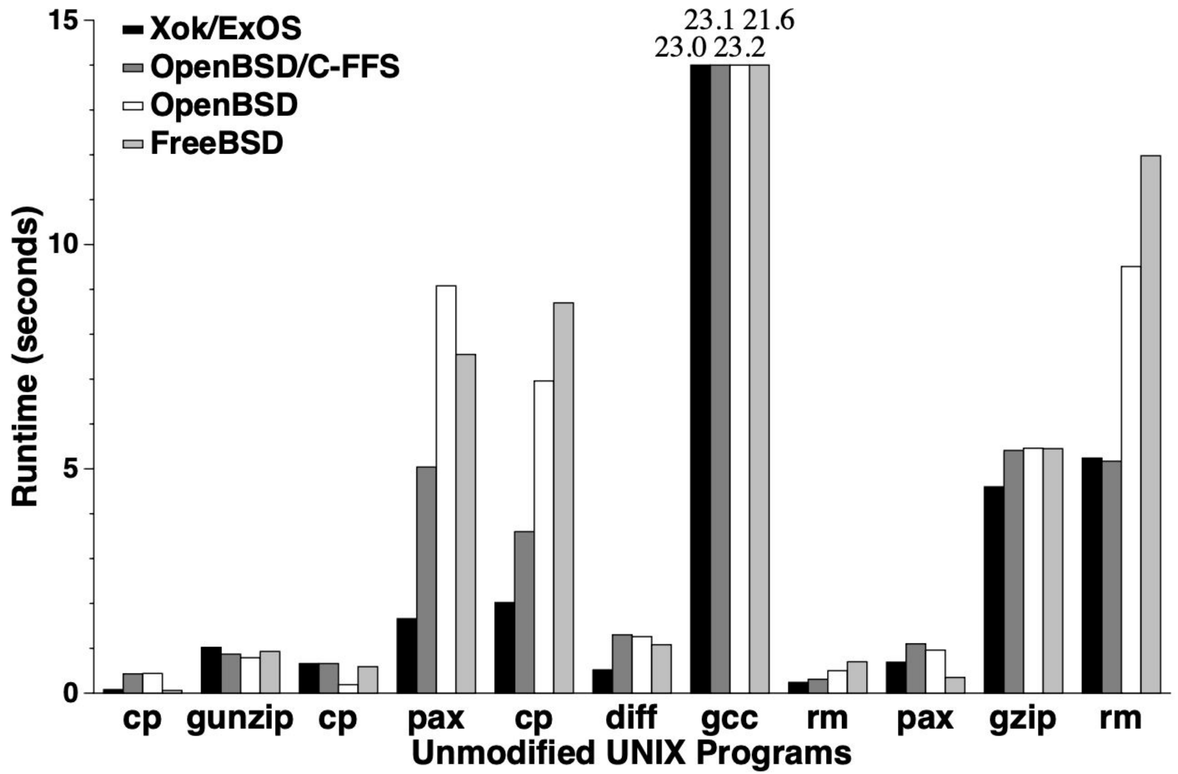
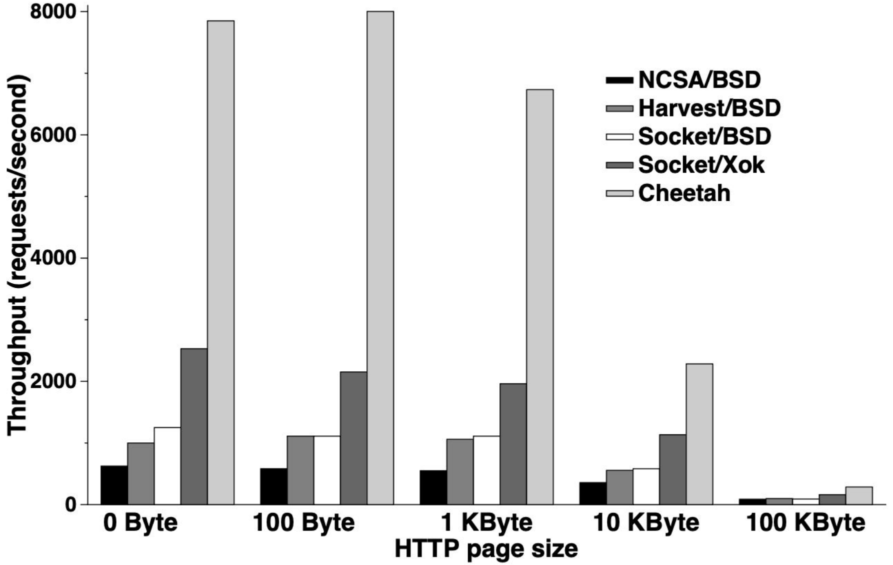


Figure 2: Performance of unmodified UNIX applications. Xok/ExOS and OpenBSD/C-FFS use a C-FFS file system while Free/OpenBSD use their native FFS file systems. Times are in seconds

Exploiting Extensibility

- Binary emulation
- XCP: "zero-touch" file copying
  - 3x faster
- Cheetah web server
  - Merge file cache and retransmission pool
  - Knowledge-based packet merging
  - HTML-based file grouping



Global Performance

Experience

- Wins
  - Exposing kernel data structures to libOSes
  - Libraries are simpler than kernels
  - Downloading code is powerful
- Losses
  - Exokernel interface is a mess
  - Information loss
  - Paging libOS

Lessons

- Provide space for application data in kernel structures
- Fast applications do not require good microbenchmark performance
- Inexpensive critical sections are useful for LibOSes
- User-level page tables