A look at exokernels, today

Léo Benito & Daniel Frédéric

History

- Introduced by MIT in 1995
- Exokernel: An Operating System Architecture for Application-Level Resource Management
- A new kernel architecture
- Disclaimer: not a general purpose OS

Motivation

"Traditional operating systems limit the performance, flexibility, and functionality of applications by fixing the interface and implementation of operating system abstractions such as interprocess communication and virtual memory."

Problem of abstraction

"The lower the level of a primitive, the more efficiently it can be implemented, and the more latitude it grants to implementors of higher-level abstraction"

Abstraction problem

- Denies domain-specific optimizations
- Discourage change of implementation of existing abstractions
- Restricts the flexibility of application builders

Exokernel architecture

- Abstraction through untrusted libOS (library operating system)
- A small kernel that multiplexes hardware resources
- Provide application-level physical resource management
- Developper can choose preferred libOS

Exokernel architecture

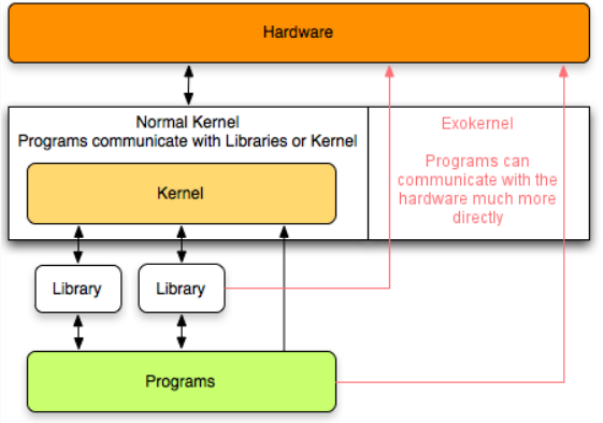

Figure 1: Exokernel architecture

Design principles

- Securely expose hardware: Multiplexed by the kernel
- Allow physical resources request: Mapping physical buffers in LibOS address space
- Expose physical names: Using devices id or easily translatable names, limiting time spent in systems like VFS. Better Resilience
- Expose revocation: Allowing LibOS to manage their accesses to the hardware and give an interface to other address spaces. Allow to unset permissions

Results

Machine	OS	Procedure call	Syscall (getpid)
DEC2100	Ultrix	0.57	32.2
DEC2100	Aegis	0.56	3.2 / 4.7
DEC3100	Ultrix	0.42	33.7
DEC3100	Aegis	0.42	2.9 / 3.5
DEC5000	Ultrix	0.28	21.3
DEC5000	Aegis	0.28	1.6 / 2.3

Figure 2: getpid performance comparison

- 10x faster than Ultrix for getpid syscall

Machine	OS	unalign	overflow	coproc	prot
DEC2100	Ultrix	n/a	208.0	n/a	238.0
DEC2100	Aegis	2.8	2.8	2.8	3.0
DEC3100	Ultrix	n/a	151.0	n/a	177.0
DEC3100	Aegis	2.1	2.1	2.1	2.3
DEC5000	Ultrix	n/a	130.0	n/a	154.0
DEC5000	Aegis	1.5	1.5	1.5	1.5

Figure 3: Exception dispatch performance comparison

- 100x faster than Ultrix for dispatching exceptions

Results

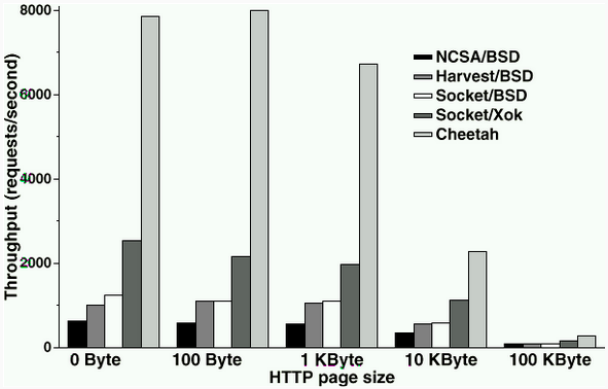

Figure 4: HTTP server performance comparison

- High performance HTTP server using exokernel primitives

Syscalls are slow

- Syscall instructions
- vDSO
- Caching
- Execution time is not dominated by syscall

API improvements

- Direct I/O
- io_uring

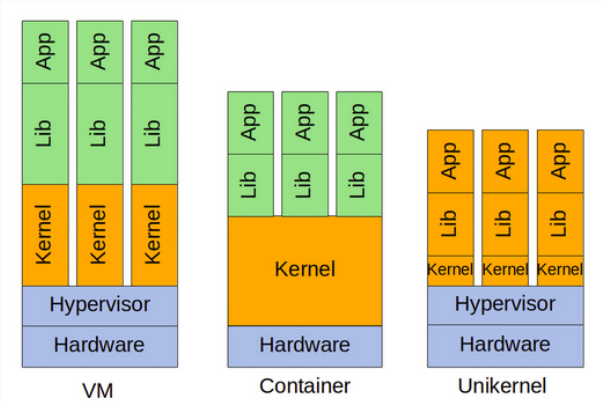

Figure 5: Unikernel architecture

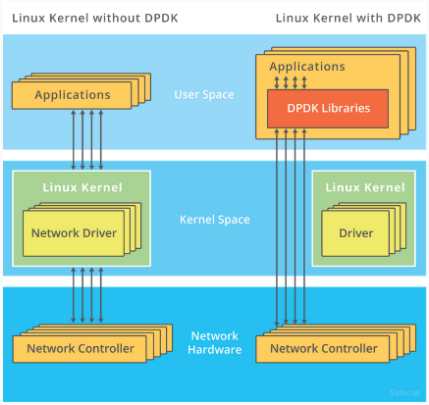

Figure 6: kernel bypass

- Bypass kernel in hardware IOs, user-defined abstraction, lockfree algorithms
- FreeBSD & Linux but originally bare metal (unikernel or exokernel)

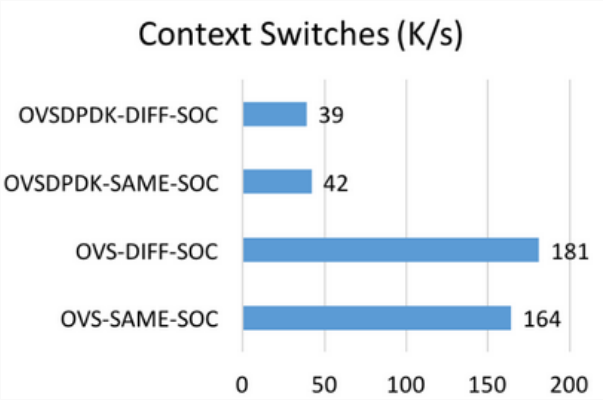

Figure 7: DPDK performances

- 5 to 6 times faster on context switches

Our implementation

- WIP
- ATM hard to customize, for instance there is a fixed entry point for LibOSes, but a system config file in ramdisk is WIP.
- Capabilities
- Cooperative Scheduling first
- Shared lockfree queue for inter libOS communication
- LibOS registration and maintenance of a table holding capabilities informations (registration etc...)
- Best bet using bitfield on capability (ids are a power of two), else linear complexity (Constant with less than 64 capabilities, linear elsewhere)

Questions

Do you have any questions ?