SHANNON JOYNER

MODERN SYSTEMS: EXTENSIBLE KERNELS AND CONTAINERS

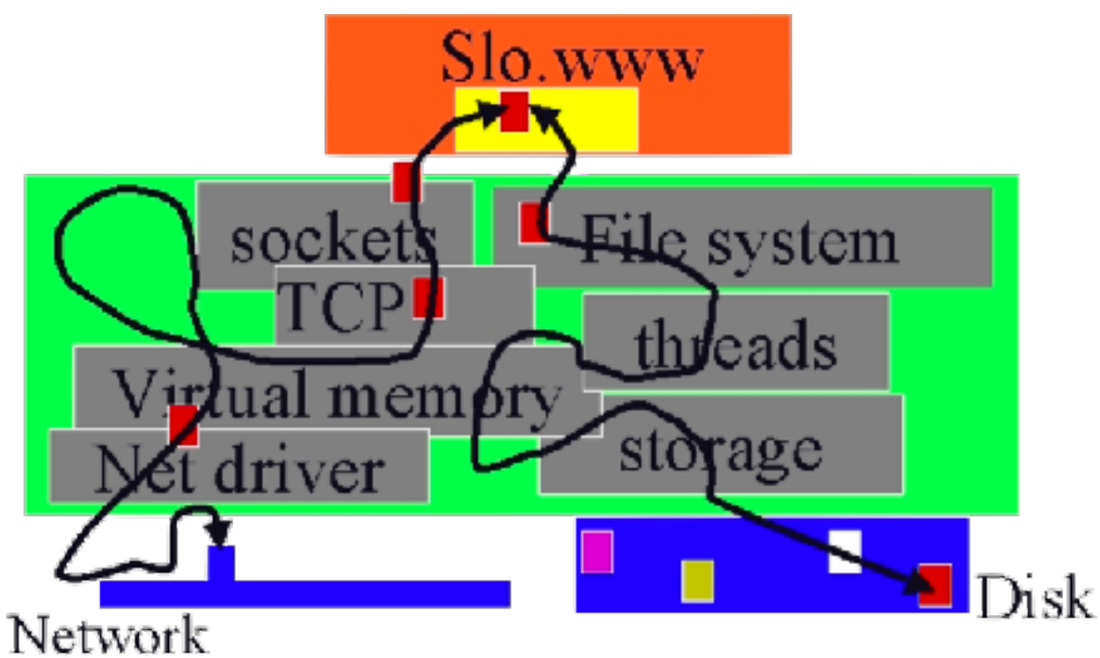
MOTIVATION

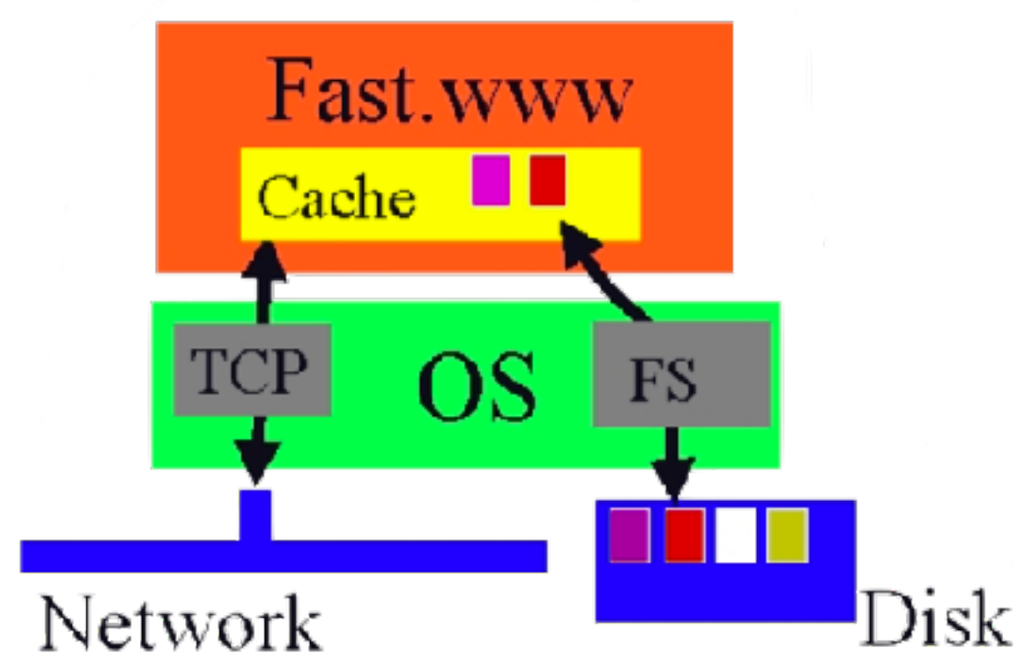
- Applications must conform to operating system interface
  - One operating system implementation is not ideal for everyone
- Cannot optimize for a given application
  - OSs did not work well for database management systems [1]

[1] Operating System Support for Database Management. Michael Stonebraker. Communications of the ACM.

MOTIVATION

- Operating systems are complex
  - Accessing resources require many system calls [1]
- This complexity in UNIX lead to Mach
  - Give application more control over system





[1] Mach: A New Kernel Foundation for UNIX Development. Mike Acetta et al. https://pdos.csail.mit.edu/archive/exo/exo-slides/sld001.htm

Exokernel: An Operating System Architecture for Application-Level Resource Management

Dawson R. Engler, M. Frans Kaashoek, and James O'Toole Jr.

M.I.T. Laboratory for Computer Science

Proc. 15th ACM Symposium on Operating Systems Principles (SOSP). Pages 251-266.

EXOKERNEL GOALS

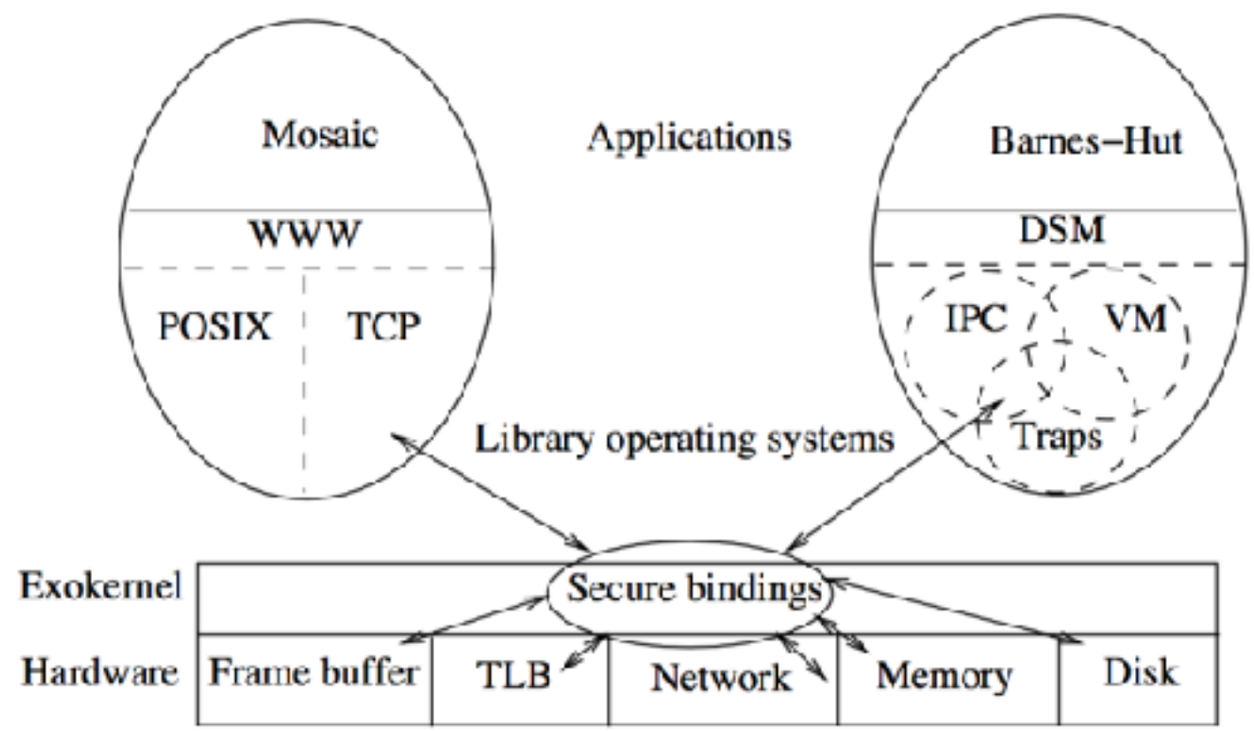
- Make small, fast kernel implementation by implementing simple primitives
- Secure, fast way to access hardware resources
- System abstractions can be implemented efficiently by application
- Applications can have fast, specialized implementations

EXOKERNEL

- Make small, fast kernel implementation by implementing simple primitives
  - Exokernel
- Secure, fast way to access hardware resources
  - TLB + Secure Bindings
- System abstractions can be implemented efficiently by application
  - LibOS + Exokernel
- Applications can have fast, specialized implementations
  - LibOS

EXOKERNEL

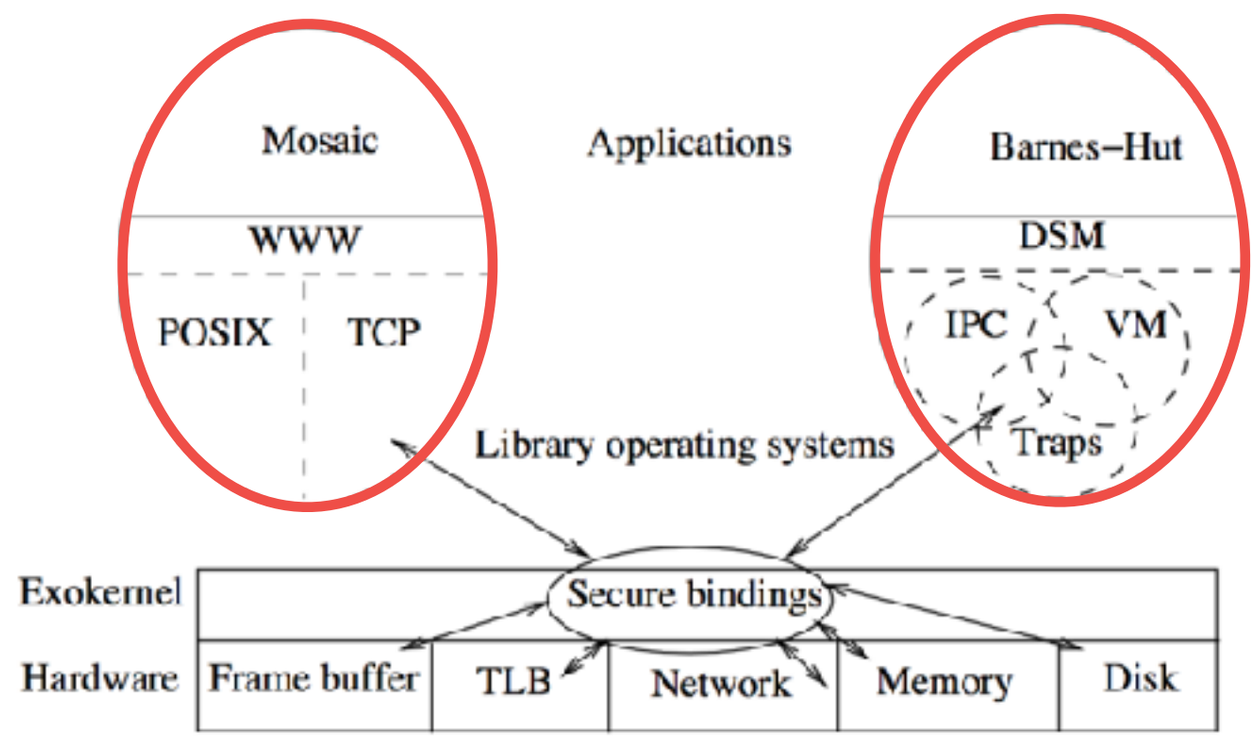
- LibOS
- Secure Bindings
- TLB
- Exokernel



Exokernel, Figure 1

LIBRARY OPERATING SYSTEMS

- Implements OS besides interaction with hardware[1]
- Customize based on application needs
- Manages resource policies
- Example LibOS customizations
  - Virtual Memory
  - Scheduling
  - Networking

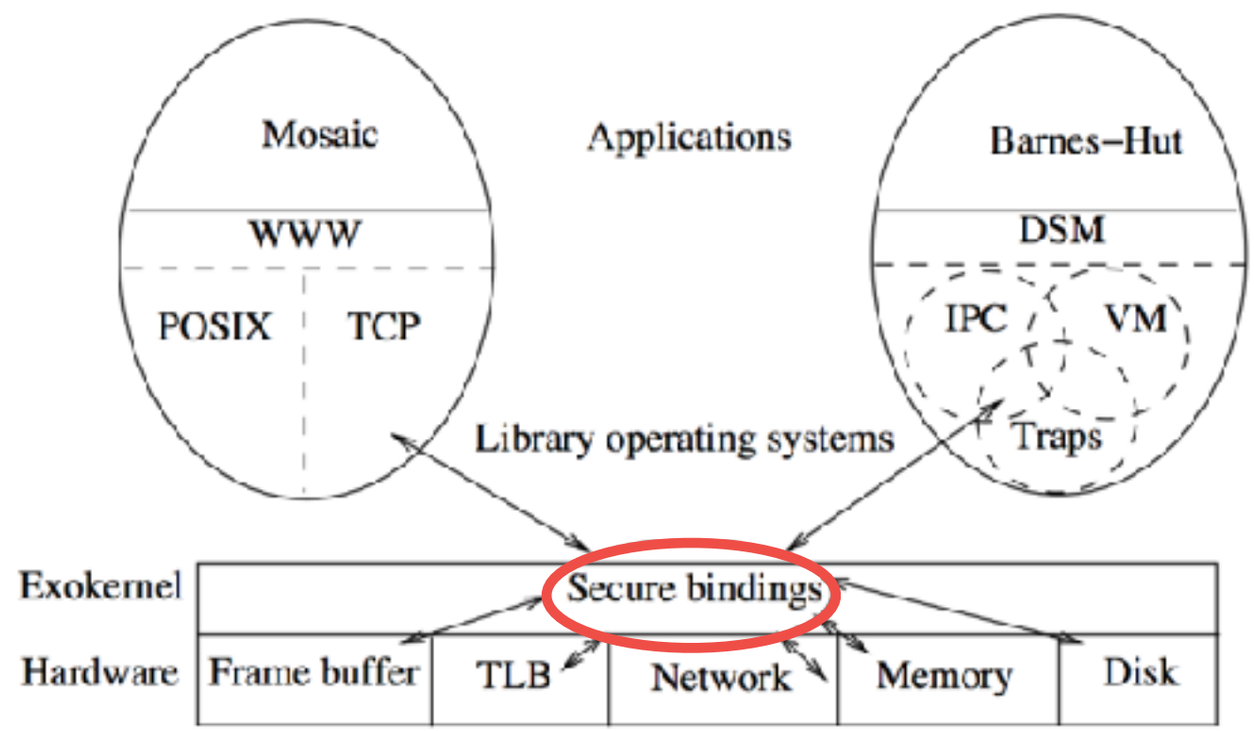


Exokernel, Figure 1

[1] http://www.cs.cornell.edu/courses/cs6410/2010fa/lectures/08-extensible-kernels.pdf

SECURE BINDINGS

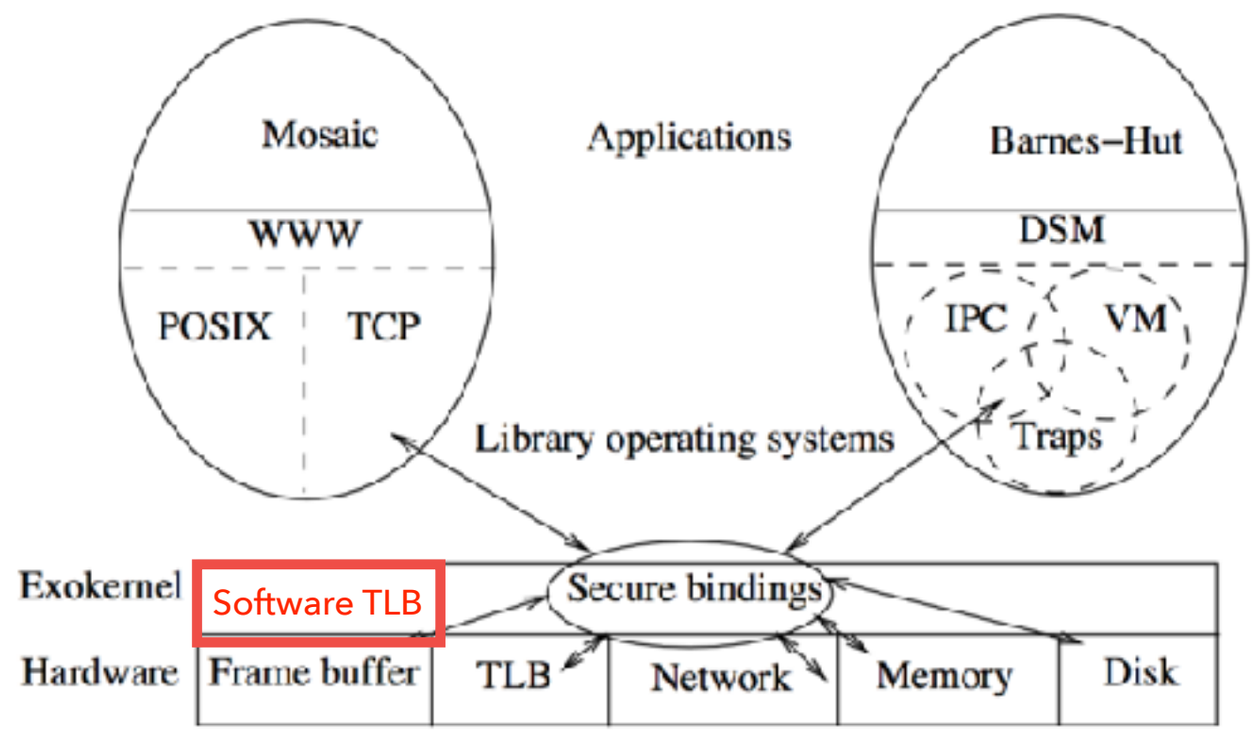
- Used by LibOSes to access resources
  - Provides connection to resource
- Decouples authorization from use of resource
- Hardware can implement protection checks quickly



Exokernel, Figure 1

SOFTWARE TRANSLATION LOOK-ASIDE BUFFER (TLB)

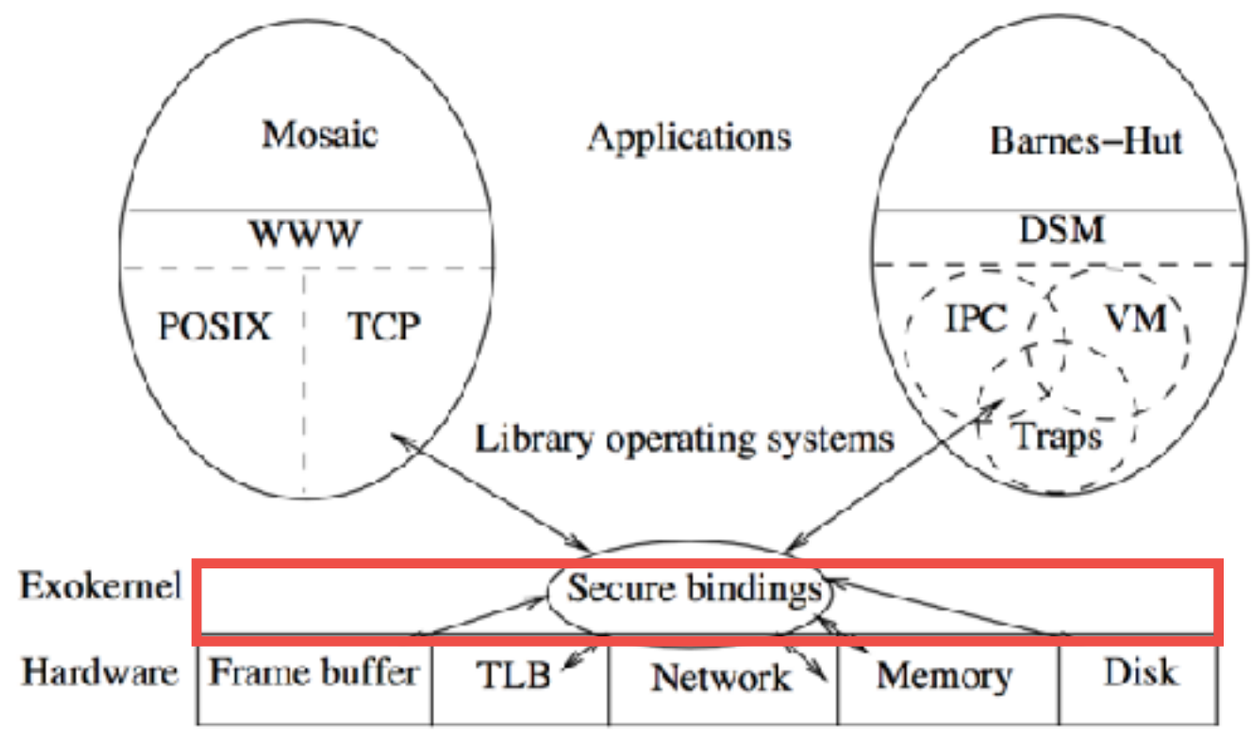
- Caches virtual memory to physical memory translations
- Cache secure bindings in TLB to reduce number of binding connections
  - Improves performance



Exokernel, Figure 1

EXOKERNEL

- Only handles resource sharing
  - Rest left to application
- Does not handle resource polices



Exokernel, Figure 1

EXOKERNEL

- Make small, fast kernel implementation by implementing simple primitives
  - Exokernel
- Secure, fast way to access hardware resources
  - TLB + Secure Bindings
- System abstractions can be implemented efficiently by application
  - LibOS + Exokernel
- Applications can have fast, specialized implementations
  - LibOS

IMPLEMENTATION

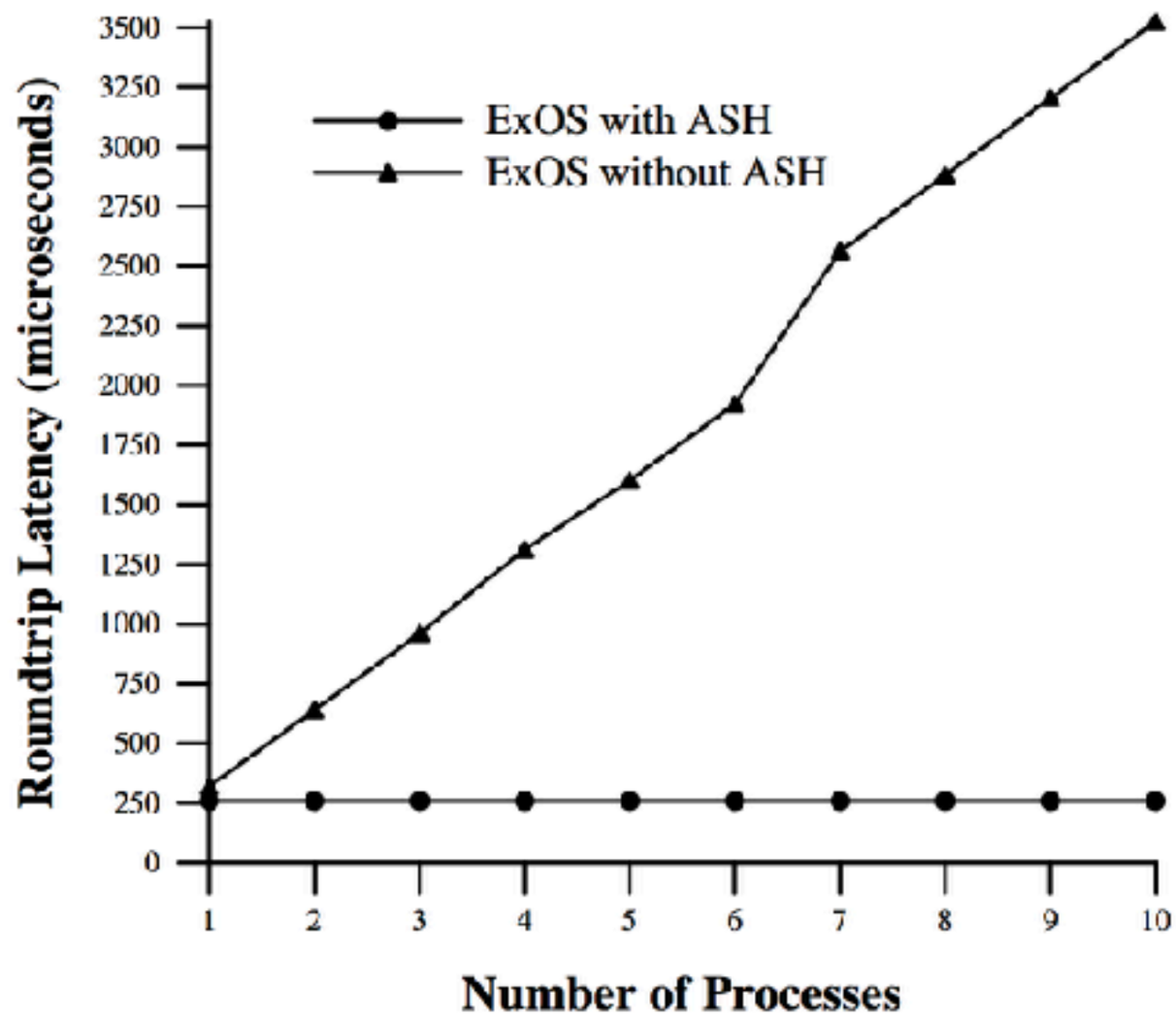
- Aegis: Exokernel [Page 7]
  - Processor Time Slices [Page 7]
  - Exceptions [Page 8]
  - Protected Control Transfer (PCT) [Page 9]
  - Dynamic Packet Filter (DPF) [Page 10]

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

Exokernel, Dispatch Exceptions, Table 5

IMPLEMENTATION

- ExOS: LibOS [Page 10]
  - Interprocess communication [Page 10]
  - Application Specific Safe Handlers [Page 11]
  - Virtual Memory [Page 11]



Exokernel, Figure 2

QUESTIONS

- Who handles resource policies?
- Is there a problem with how exokernels handle resource policies?
- Why do we need secure bindings?

QUESTIONS

- Who handles resource policies?
- Is there a problem with how exokernels handle resource policies?
- Why do we need secure bindings?

QUESTIONS

- Who handles resource policies?
  - Application handles resource policies.

QUESTIONS

- Who handles resource policies?
- Is there a problem with how exokernels handle resource policies?
- Why do we need secure bindings?

QUESTIONS

- Is there a problem with how exokernels handle resource policies?
  - Poor isolation. Applications can have conflicting policies.

QUESTIONS

- Who handles resource policies?
- Is there a problem with how exokernels handle resource policies?
- Why do we need secure bindings?

QUESTIONS

- Why do we need secure bindings?
  - Protection / authorization of resources. Faster to do in kernel and kernel does not need to understand resources.

EXOKERNEL TAKEAWAYS

- Strengths
  - Minimal kernel and customizable operating system
  - Fast
- Weakness
  - Poor isolation
    - Each application implements own LibOS
    - No way to prevent systems from conflicting
  - Hardware compatibility
    - Need to change LibOS depending on hardware interface

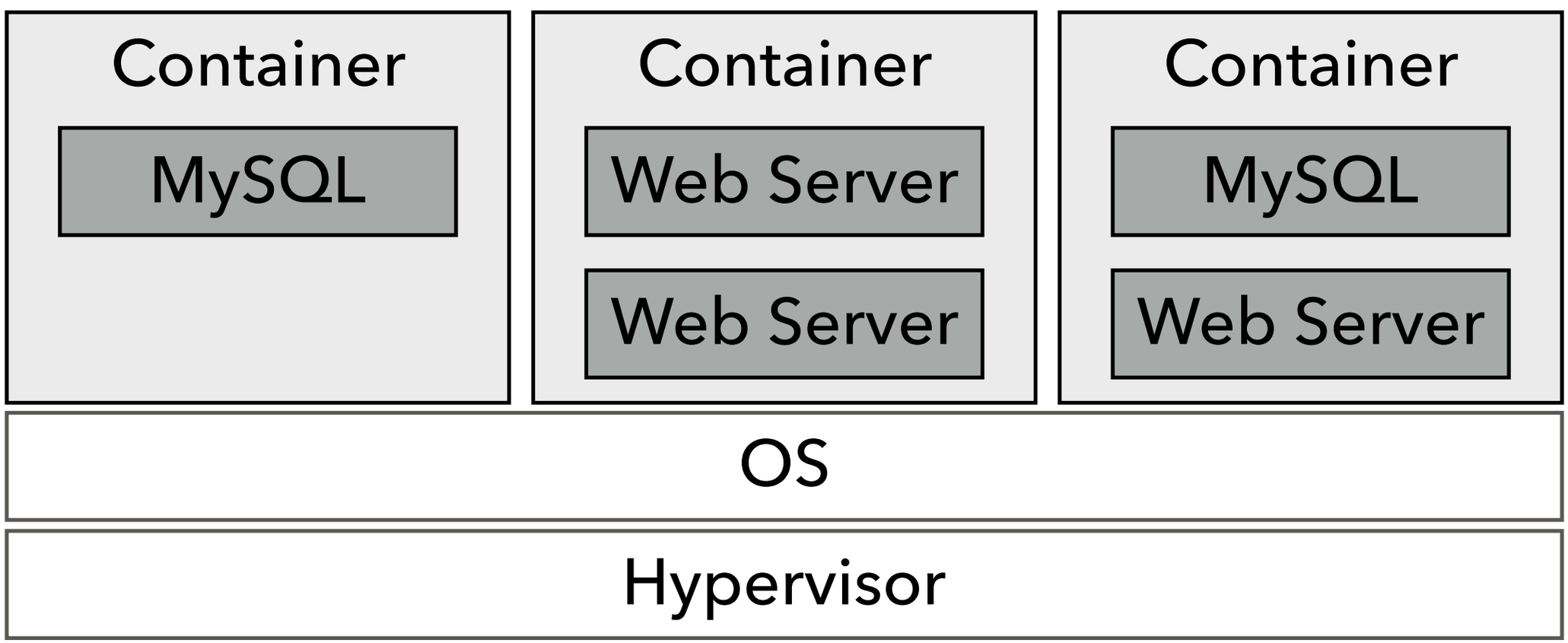
`

- Hypervisor
  - Hardware abstraction
- Previous virtual machines ran on top of hypervisors
  - No isolation

[1] Container-based Operating System Virtualization: A Scalable, High performance Alternative to Hypervisors
Stephen Soltesz, Herbert Potzi, Marc Fiuczynski, Andy Bavier, Larry Peterson. EuroSys '07.

CONTAINERS

- Grouping of processes
- Strength
  - Provide isolation between groups
- Weakness
  - Containers cannot customize operating systems



Unikernel: Library Operating Systems for the Cloud

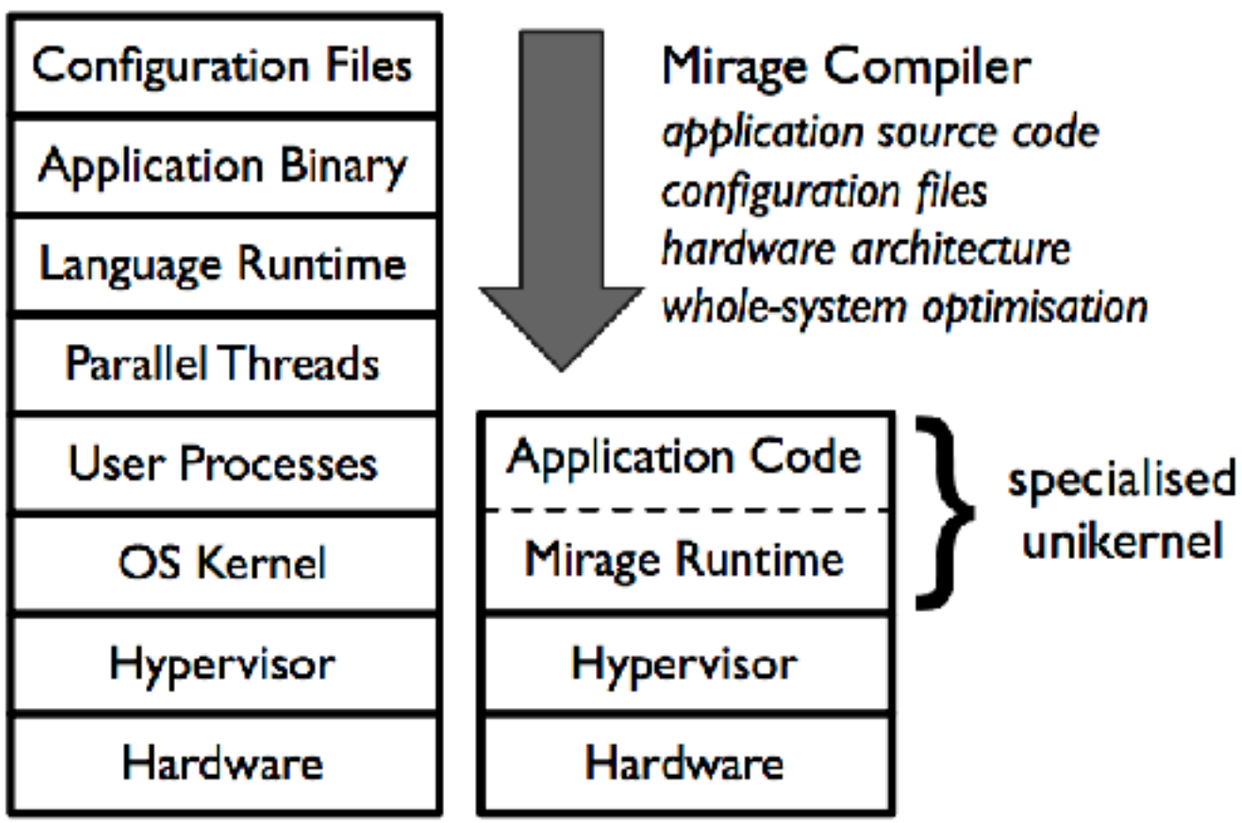
Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Ralraj Singh, ThomasGazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft

University of Cambridge, University of Nottingham, Citrix Systems Ltd, OCamlPro SAS

In Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems pg. 461–472.

UNIKERNEL = EXOKERNEL + CONTAINERS

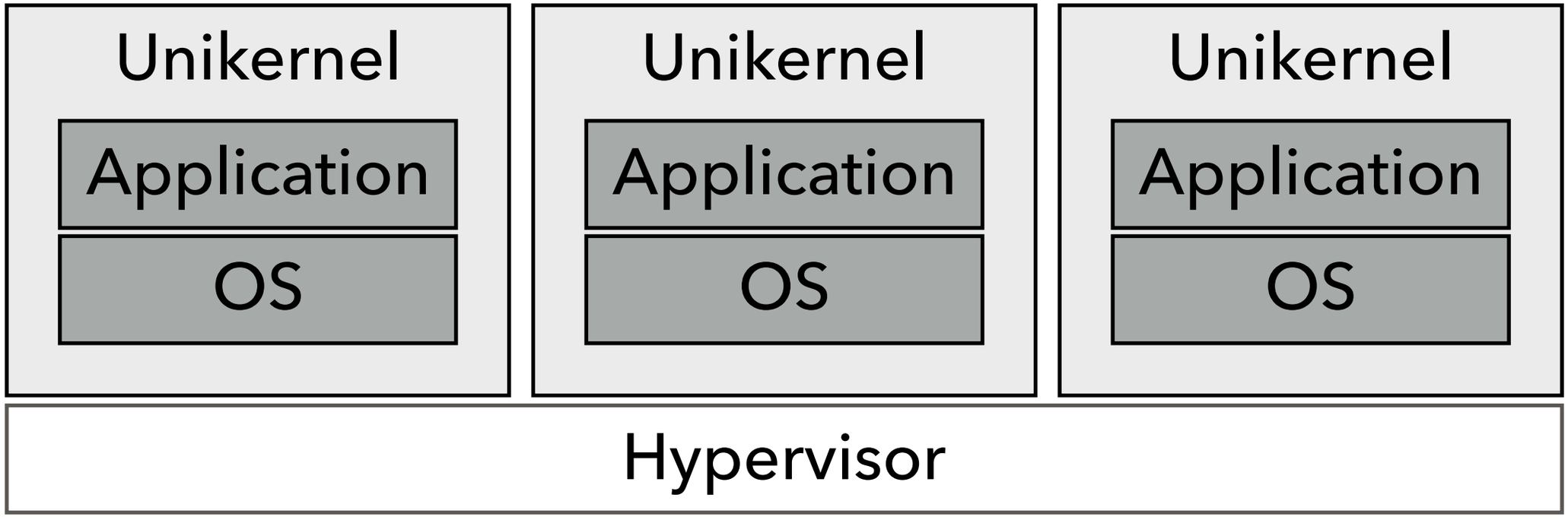
- Run one application per virtual machine
- One process per application
- Everything compiled into a VM image
- Do not compile unused code



Unikernel, Figure 1

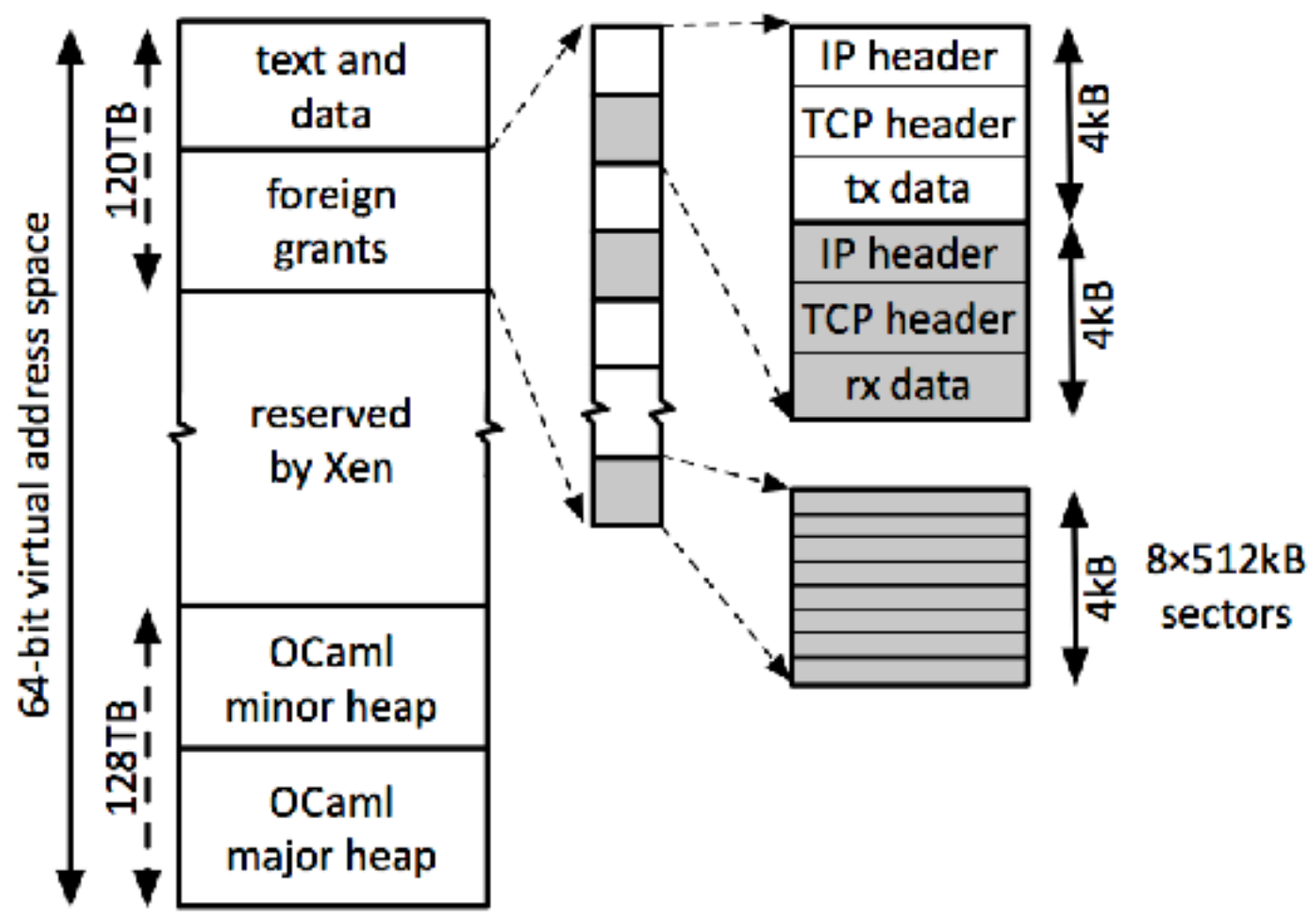
UNIKERNEL

- Run directly on top of standard hypervisor
- Can run multiple unikernels on the same hypervisor



MIRAGE

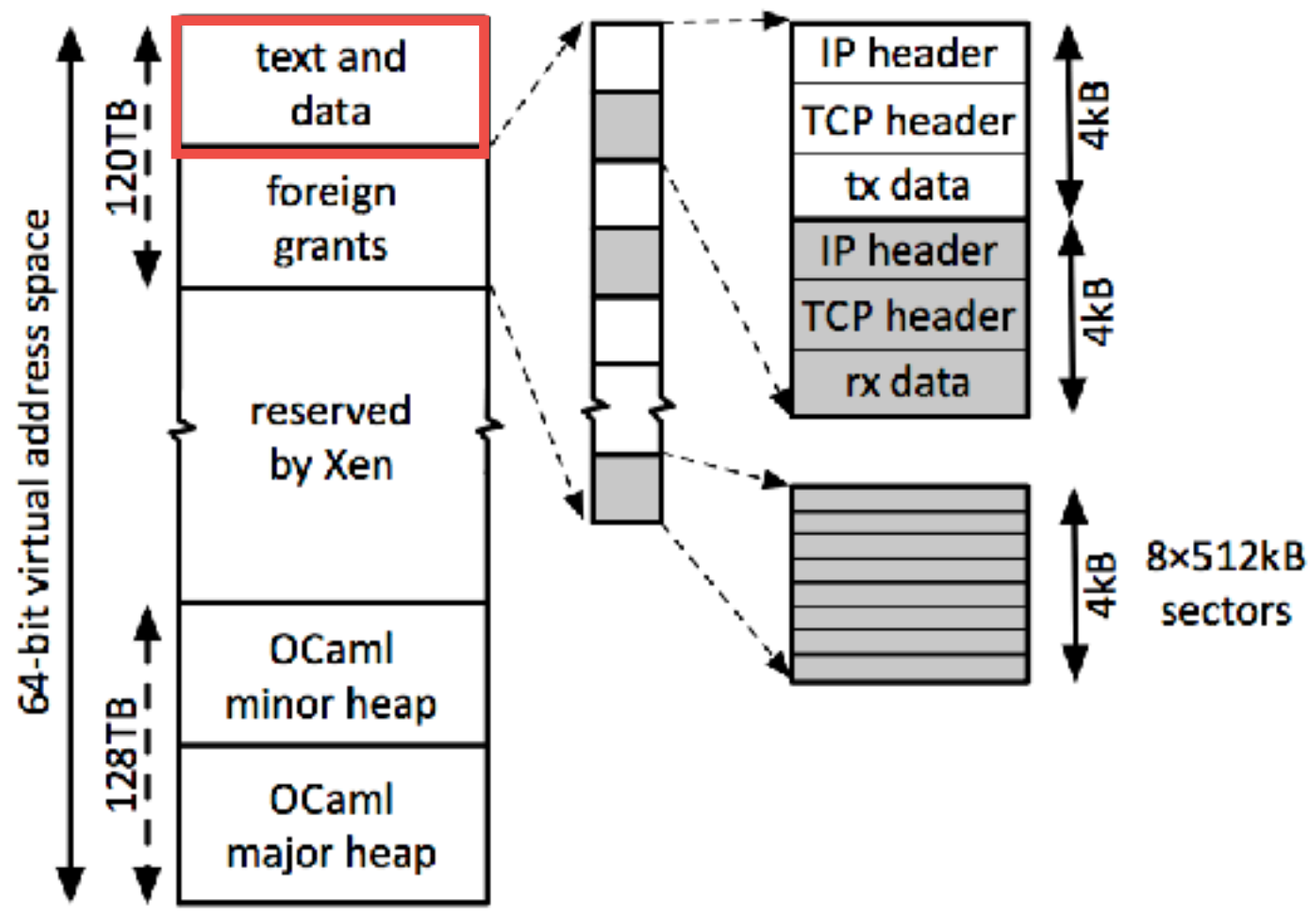
- Produces unikernels
- Compiles OCaml code to Xen VM image
- 4 main components
  - Text + Data segment
  - Foreign Grants
  - Minor Heap
  - Major Heap



Unikernel, Figure 2

TEXT AND DATA

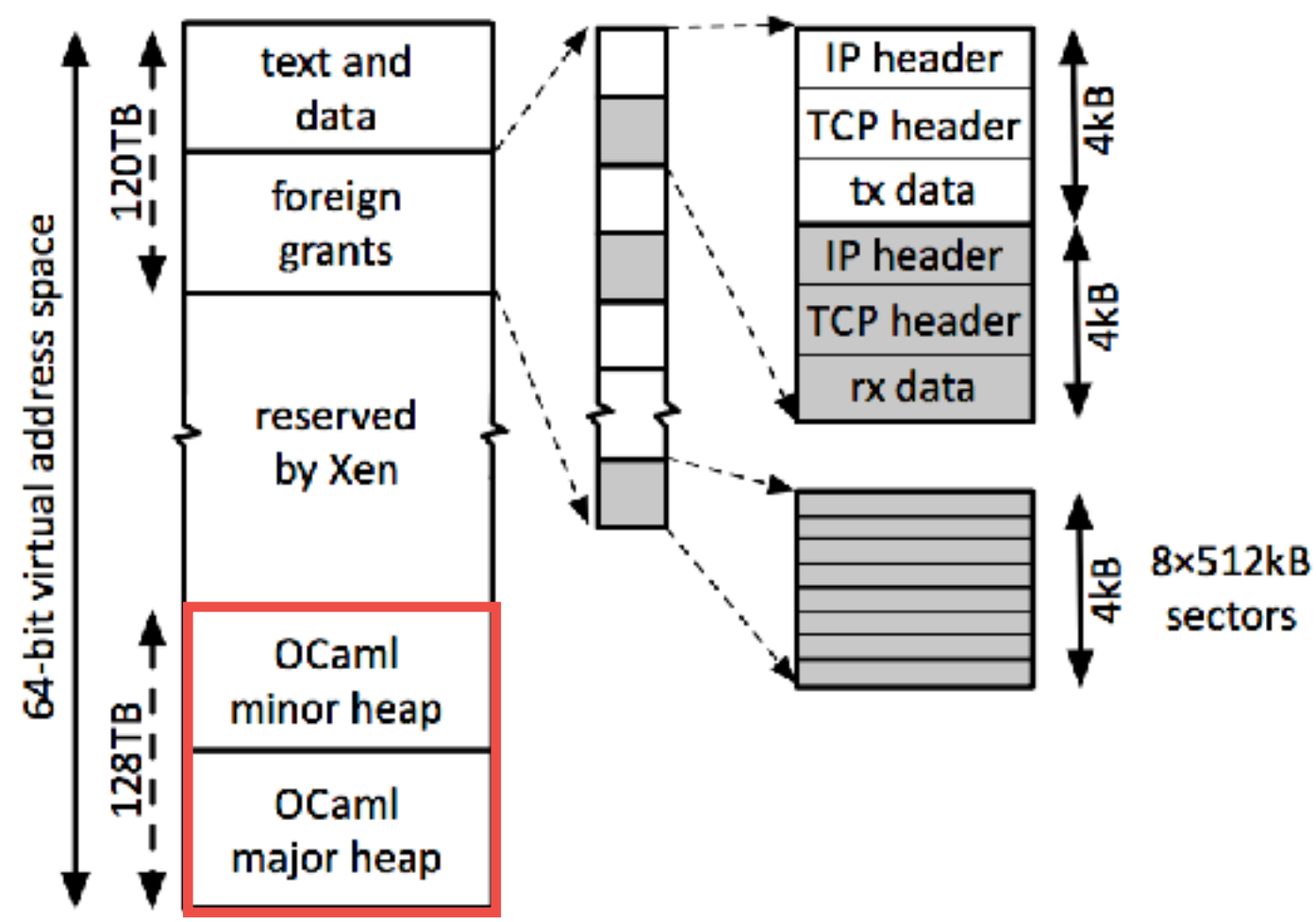
- OCaml Runtime
- PVBoot
  - Initializes VM



Unikernel, Figure 2

HEAP

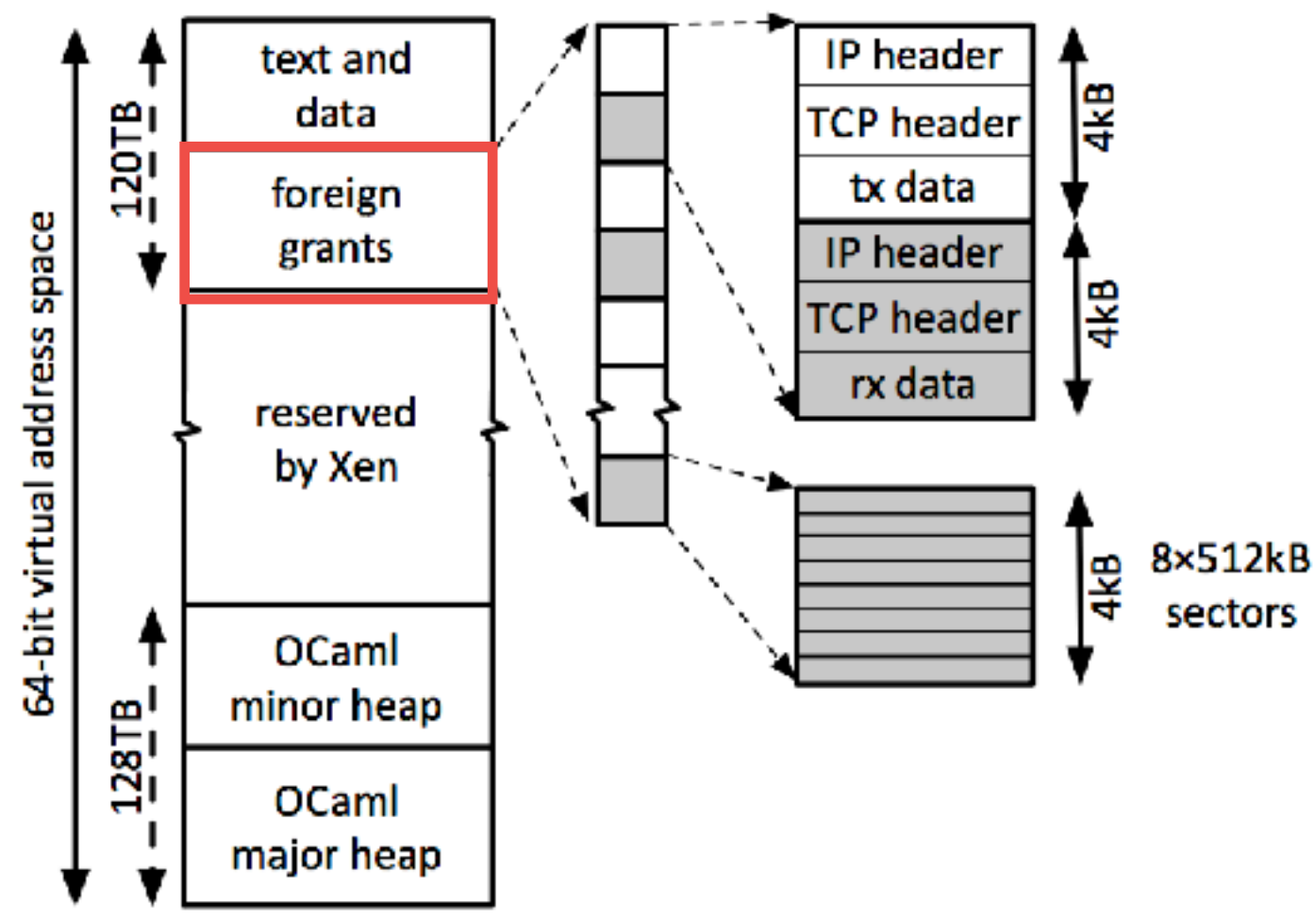
- Minor Heap
  - Short lived values in VM
  - Fast
- Major Heap
  - Long lived values



Unikernel, Figure 2

FOREIGN GRANTS

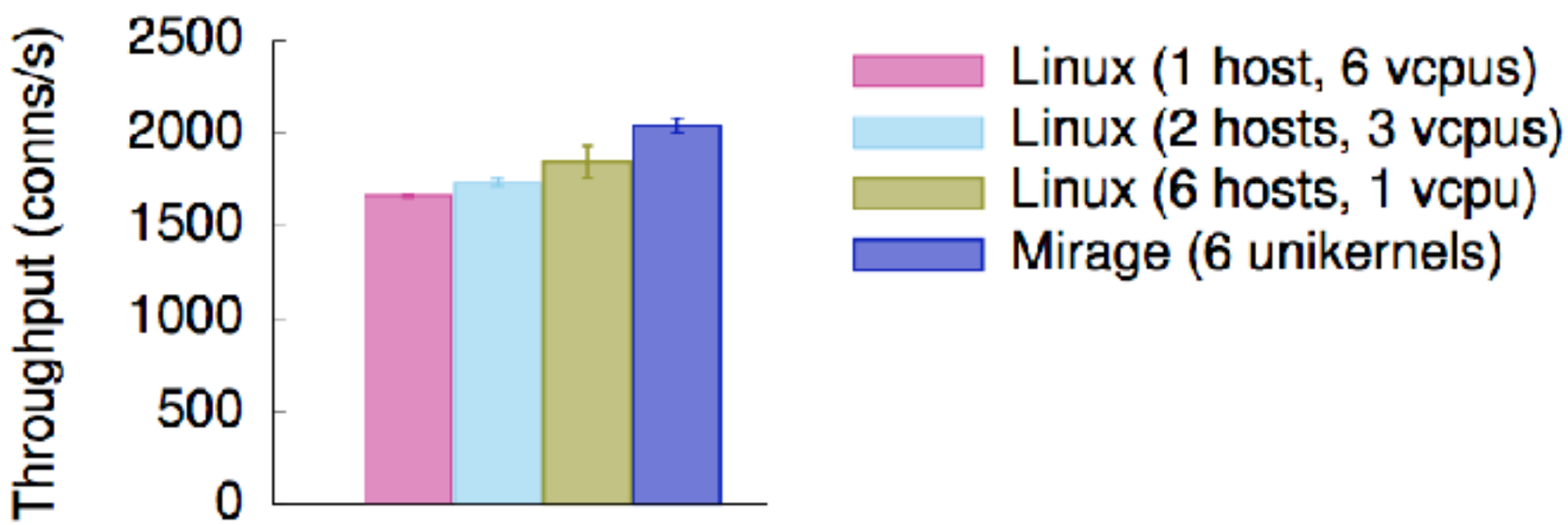
- Used for VM communication
- Write data to a grant table
- Exchange table between VM address spaces



Unikernel, Figure 2

APACHE BENCHMARK

- Mirage unikernel improvements result in better performance than having multiple cores



Unikernel, Figure 13

EXOKERNEL VERSUS UNIKERNEL

- Exokernel
  - Fast and customizable
  - All applications on same system
  - Poor isolation
- Unikernel
  - Fast and customizable
  - Single application per system
  - Better isolation

ACKNOWLEDGEMENTS

- Thanks to Hakim for helping me prepare for this presentation!

- Operating System Support for Database Management. Michael Stonebraker. Communications of the ACM.
- Mach: A New Kernel Foundation for UNIX Development. Accetta, Mike, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young. 1986.
- Exokernel: An Operating System Architecture for Application-Level Resource Management. Dawson R. Engler, M. Frans Kaashoek, and James O'Toole Jr. Proc. 15th ACM Symposium on Operating Systems Principles (SOSP). Pages 251-266.
- http://www.cs.cornell.edu/courses/cs6410/2010fa/lectures/08-extensible-kernels.pdf
- Container-based Operating System Virtualization: A Scalable, High performance Alternative to Hypervisors Stephen Soltesz, Herbert Potzi, Marc Fiuczynski, Andy Bavier, Larry Peterson. EuroSys '07.
- Unikernel: Library Operating Systems for the Cloud. Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Ralraj Singh, ThomasGazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. University of Cambridge, University of Nottingham, Citrix Systems Ltd, OCamlPro SAS. In Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems pg. 461–472.

PROTECTED CONTROL TRANSFER

- Implementation of interprocess communication
  - Put the messages in the receiver process's context
- Asynchronous: Rest of sender process's time slice goes to receiver
- Synchronous: All future time slices go to receiver process
- 7x faster than best reported implementation

OS	Machine	MHz	Transfer cost
Aegis	DEC2100	12.5MHz	2.9
Aegis	DEC3100	16.67MHz	2.2
Aegis	DEC5000	25MHz	1.4
L3	486	50MHz	9.3 (normalized)

Exokernel, Table 6

TIME TO PERFORM NULL PROCEDURE AND SYSTEM CALLS

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

Exokernel, Table 5

TIME FOR IPC

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

Exokernel, Table 8

TIME TO PERFORM VM OPERATIONS

Machine	OS	dirty	prot1	prot100	unprot100	trap	appel1	appel2
DEC2100	Ultrix	n/a	51.6	175.0	175.0	240.0	383.0	335.0
DEC2100	ExOS	17.5	32.5	213.0	275.0	13.9	74.4	45.9
DEC3100	Ultrix	n/a	39.0	133.0	133.0	185.0	302.0	267.0
DEC3100	ExOS	13.1	24.4	156.0	206.0	10.1	55.0	34.0
DEC5000	Ultrix	n/a	32.0	102.0	102.0	161.0	262.0	232.0
DEC5000	ExOS	9.8	16.9	109.0	143.0	4.8	34.0	22.0

Exokernel, Table 10

TIME TO PERFORM VM OPERATIONS (TWO DIFFERENT PAGE-TABLES)

Machine	Method	dirty	prot1	prot100	unprot100	trap	appel1	appel2
DEC2100	Original page-table	17.5	32.5	213.	275.	13.9	74.4	45.9
DEC2100	Inverted page-table	8.0	23.1	253.	325.	13.9	54.4	38.8
DEC3100	Original page-table	13.1	24.4	156.	206.	10.1	55.0	34.0
DEC3100	Inverted page-table	5.9	17.7	189.	243.	10.1	40.4	28.9

Table 3, Exokernel

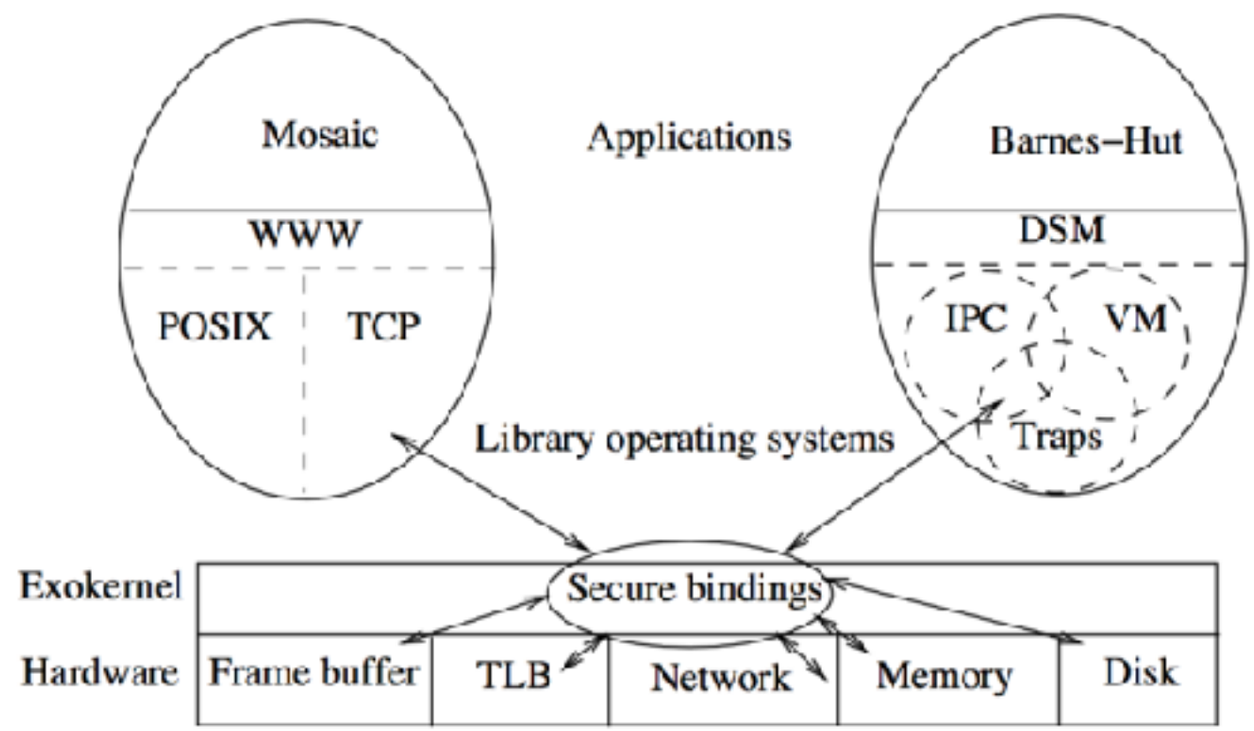
ROUNDTRIP LATENCY OVER ETHERNET

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

Exokernel, Table 11

EXOKERNEL: SEPARATE PROTECTION FROM MANAGEMENT

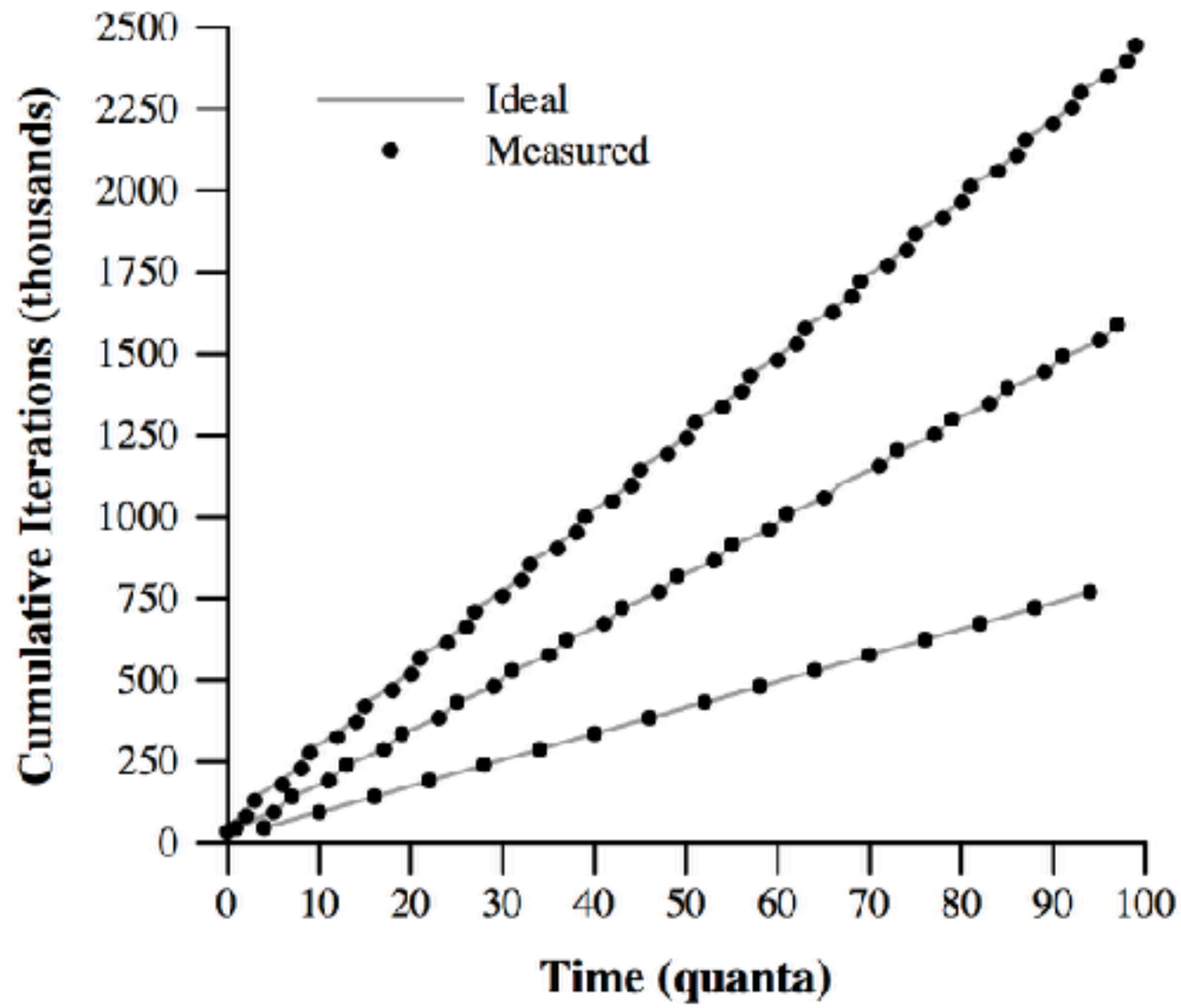
- Tracking ownership of resources
- Protect bindings and resource usage
  - Ex. Accessing memory not accessible to application
- Revoking resource privileges



Exokernel, Figure 1

EXOS APPLICATION-LEVEL STRIDE SCHEDULER

- Each process given a fixed proportion of resources
- Implemented counter program



Exokernel, Figure 3

TIME TO PERFORM LRPC EXTENSIONS

Machine	lrpc	tlrpc
DEC2100	13.9	8.6
DEC3100	10.4	6.4
DEC5000	6.3	2.9

Table 12

TIME TO CLASSIFY TCP/IP HEADERS

Filter	Classification Time
MPF	35.0
PATHFINDER	19.0
DPF	1.5

Exokernel, Table 7

TIME TO PERFORM 150X150 MATRIX MULTIPLICATION

Machine	OS	matrix
DEC2100	Ultrix	7.1
DEC2100	ExOS	7.0
DEC3100	Ultrix	5.2
DEC3100	ExOS	5.2
DEC5000	Ultrix	3.8
DEC5000	ExOS	3.7

Exokernel, Table 9

SAMPLE OF AEGIS'S CALL INTERFACE

System call	Description
`Yield`	Yield processor to named process
`Scall`	Synchronous protected control transfer
`Acall`	Asynchronous protected control transfer
`Alloc`	Allocation of resources (*e.g.*, physical page)
`Dealloc`	Deallocation of resources

Table 2

MACHINE CONFIGURATION FOR EXOKERNEL EXPERIMENTS

Machine	Processor	SPEC rating	MIPS
DEC2100 (12.5 MHz)	R2000	8.7 SPECint89	~ 11
DEC3100 (16.67 MHz)	R3000	11.8 SPECint89	~ 15
DEC5000/125 (25 MHz)	R3000	16.1 SPECint92	~ 25

Table 1

SAMPLE OF AEGIS'S PRIMITIVE OPERATIONS

Primitive operations	Description
`TLBwr`	Insert mapping into TLB
`FPUmod`	Enable/disable FPU
`CIDswitch`	Install context identifier
`TLBvadelete`	Delete virtual address from TLB

Table 3

EXOKERNEL

- Provides interface to interact with hardware
- Physical resources managed at application-level
- Most of the system implemented using a Library Operating System
- Separate resource protection from resource management

SECURE BINDINGS

- Used by LibOSes to access resources
- Decouples authorization from use of resource
- Performs authorization at bind time
- Hardware can implement protection checks quickly

LIBRARY OPERATING SYSTEM (LIBOS)

- Use hardware interface
- Implement rest of the operating system
- Implementation can be specialized to meet needs of applications