MODERN SYSTEMS: EXTENSIBLE KERNELS AND CONTAINERS

CS6410

Hakim Weatherspoon

Motivation

- Monolithic Kernels just aren't good enough?
  - Conventional virtual memory isn't what userspace programs need (Appel + Li '91)
  - Application-level control of caching gives 45% speedup (Cao et al '94)
  - Application-specific VM increases performance (Krueger '93, Harty + Cheriton '92)
  - **Filesystems for databases** (Stonebraker '81)
  - And more...

Motivation

- Lots of problems…

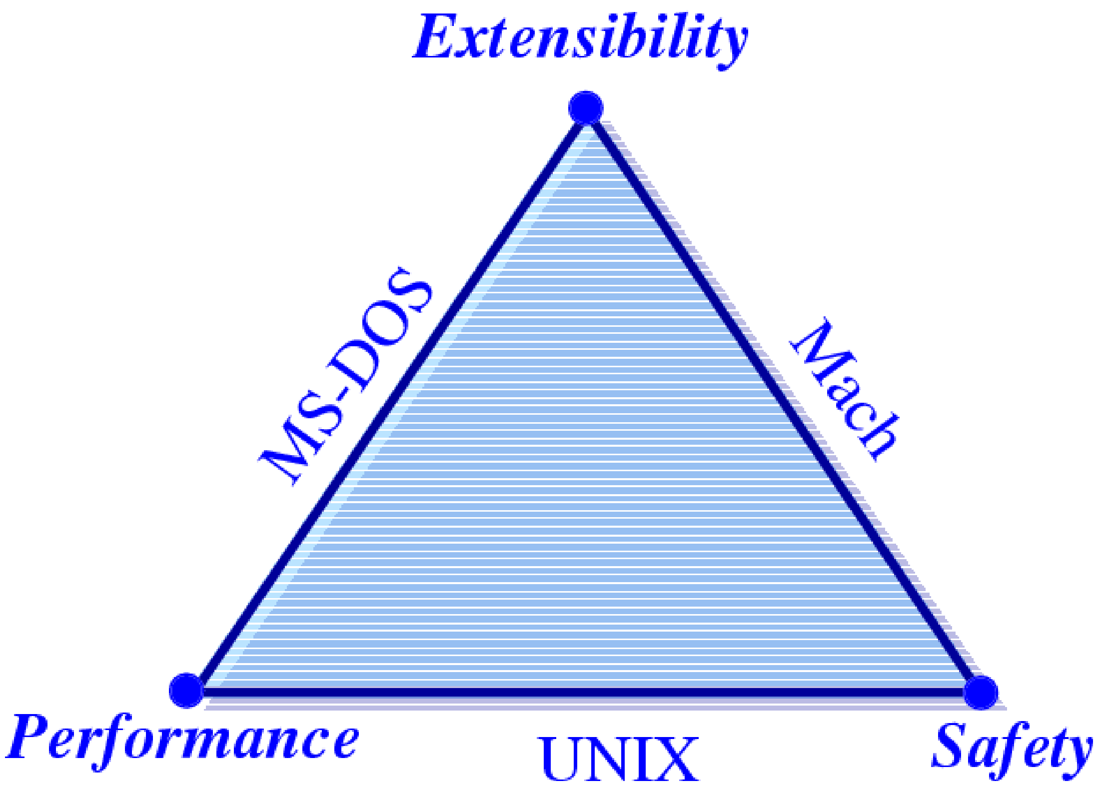
Motivation

- Lots of problems…Lots of design opportunities!

Motivation

- Extensibility
- Security
- Performance

Can we have all 3
in a single OS?



From Stefan Savage's SOSP 95 presentation

Context for these papers

- 1990's
  - Researchers (mostly) were doing special purpose OS hacks
  - Commercial market complaining that OS imposed big overheads on them
  - OS research community began to ask what the best way to facilitate customization might be. In the spirit of the Flux OS toolkit…
- 2010's
  - containers: single-purpose appliances
  - Unikernels: ("sealable") single-address space
    - Compile time specialized

Motivation

- 1988-1995: lots of innovation in OS development
  - Mach 3, the first "true" microkernel
  - SPIN, Exokernel, Nemesis, Scout, SPACE, Chorus, Vino,
  - Amoeba, etc...
  - And even more design papers

Motivation

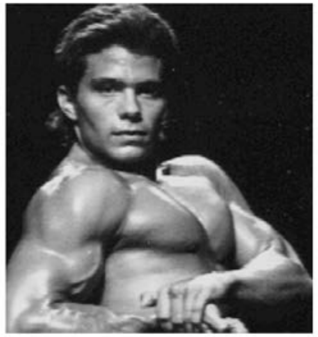
- Exploring new spaces
  - Distributed computing
  - Secure computing
  - **Extensible kernels** (exokernel, unikernel)
  - **Virtual machines** (exokernel)
  - **New languages** (spin)
  - **New memory management** (exokernel, unikernel)

Exokernel

- Dawson R. Engler, M. Frans Kaashoek and James O'Toole Jr.
- Engler's Master's Thesis.
- Follow-up publications on 1997 and 2002.
- Kaashoek later worked on Corey.

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

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



Exokernels - Motivation

- Existing Systems offer fixed high-level abstractions which is bad
  - Hurt app performance (generalization – eg: LRU)
  - Hide information (eg: page fault)
  - Limit functionality (infrequent changes – cool ideas don’t make it through)

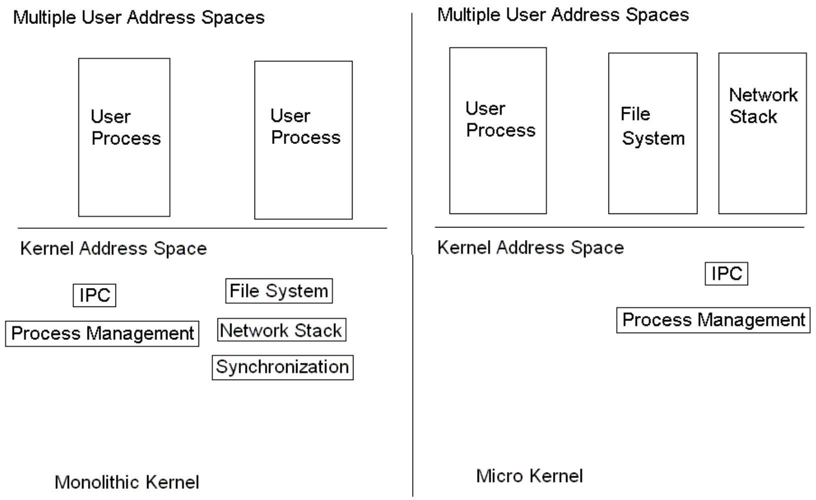
Motivation (cont.)

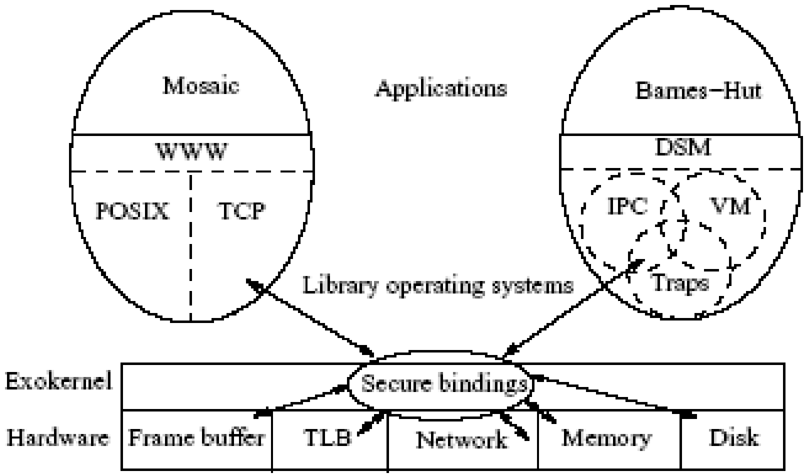
- Separate protection from management, mgmt in user space
- Apps should use domain specific knowledge to influence OS services
- Small and simple kernel – adaptable and maintainable

Exokernel

- Kernel only multiplexes hardware resources (Aegis)
- Higher-level abstractions in Library OS (ExOS)
- Secure binding, Visible resource revocation, Abort
- Apps link with the LibOS of their choice

OS Component Layout





Exokernel

Exokernel main ideas

- Kernel
  - Resource sharing, not policies
- Library Operating System
  - Responsible for the abstractions
    - IPC
    - VM
    - Scheduling
    - Networking

Lib OS and the Exokernel

- Lib OS (untrusted) can implement traditional OS abstractions (compatibility)
- Efficient (Lib OS in user space)
- Apps link with Lib OS of their choice
- Kernel allows LibOS to manage resources, protects LibOss

Exokernel vs Microkenels 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.

Design

- Application-level resource management
- Exports hardware resources
- Multiplexes access between processes
- Separates policy from management
  - avoid resource management!

What problems do we solve?

- High-level abstractions
  - hurt application performance
  - Hide information
  - Limit functionality
- Existing monolithic kernels
  - Encourage stable (archaic) interfaces
  - Difficult to extend with modern techniques

How do we solve them: Design

- Secure bindings
- Downloading code
- Visible resource revocation
- The abort protocol

How do we solve them: Design

- **Secure bindings**
- Downloading code
- Visible resource revocation
- The abort protocol

Secure bindings

- Decouples authorization from use
- Authorize once, at “bind time”
- Use transferable “capabilities” to check access
- Cache bindings in-kernel to decrease binding frequency
  - Example: huge software-based TLB

How do we solve them: Design

- Secure bindings
- **Downloading code**
- Visible resource revocation
- The abort protocol

Downloading code

- Userspace application produces kernel space code
- Access checks at download time
- Code is verified before being run, with JIT for speed

How do we solve them: Design

- Secure bindings
- Downloading code
- **Visible resource revocation**
- The abort protocol

Visible resource revocation

- Revocation traditionally invisible (or transparent)
  - Expensive: have to save entire state
- Try visible instead!
  - Save only the state you need
  - Kernel gives you a few microseconds to do it

How do we solve them: Design

- Secure bindings
- Downloading code
- Visible resource revocation
- **The abort protocol**

- Revocation: kernel asks process for resource
  - “relinquish page 5 please”
  - Process tracks state and returns resource
- Abort: kernel demands resource
  - “page 5 in 50 microseconds”
  - Takes resource “by force”
  - Invalidates credentials and bindings.
  - Notifies library operating system

Exokernel

- DEC MIPS
- Aegis: actual exokernel
  - Processor
  - Physical memory
  - TLB
  - Exceptions, Interrupts
- ExOS: library operating system
  - Processes, IPC, Virtual Memory, Network protocols

Microbenchmark results

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

ExOS Virtual Memory

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

+ Fast Sys call.
- Half the time in look-up (vector).

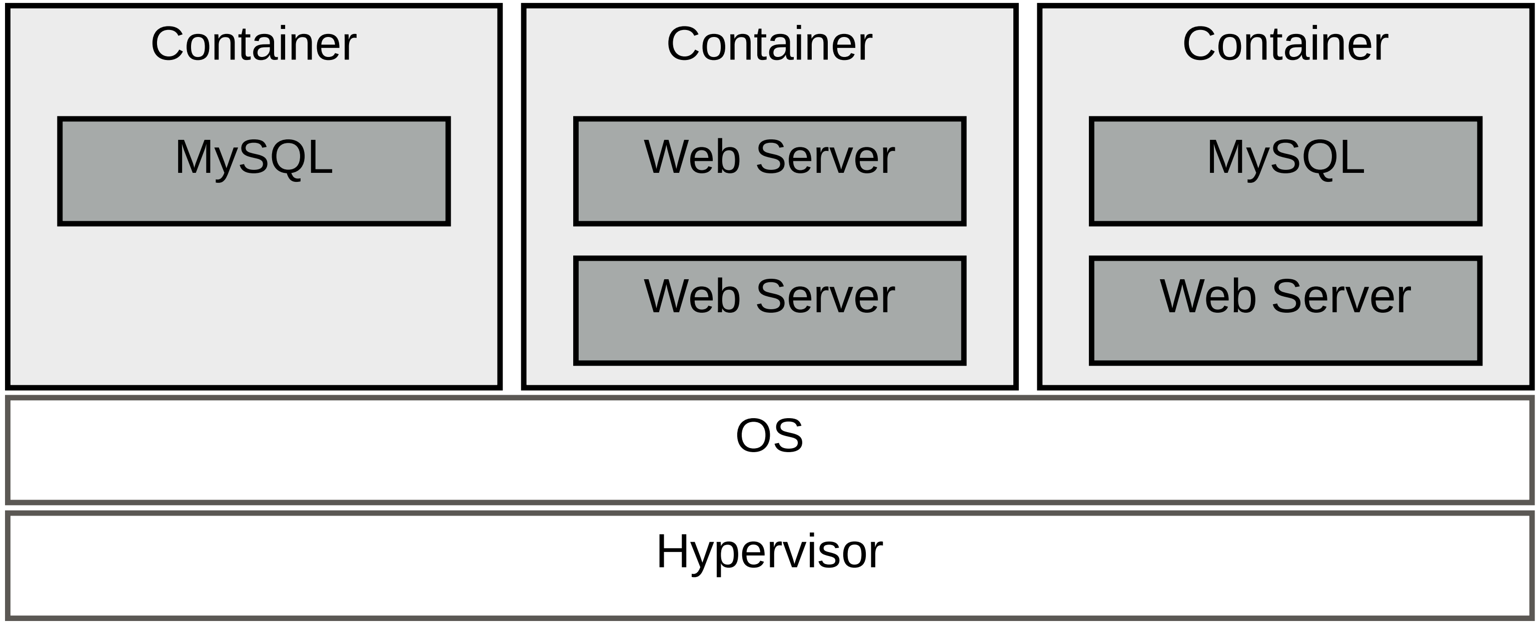
Repeated access to Aegis STLB and ExOS PageTable

Perspective

- Extensible kernels are actually fast.
- End-to-end arguments.
- Efficient implementations.
- Extensibility without loss of security or performance
  - Exokernels
    - Safely export machine resources
    - Decouple protection from management

Containers

- Grouping of processes
- Provide isolation between groups
- Containers cannot customize operating systems
- Isn’t this similar to the problem exokernels tried to solve?



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

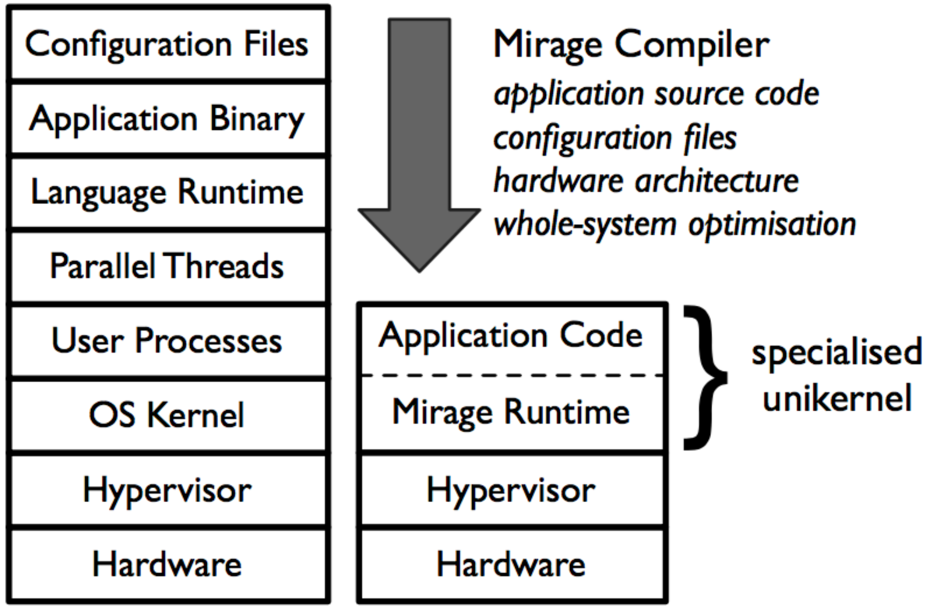
In Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems pg. 461–472.

Unikernel slides from Shannon Joyner

Unikernel = EXOKERNEL + CONTAINERs

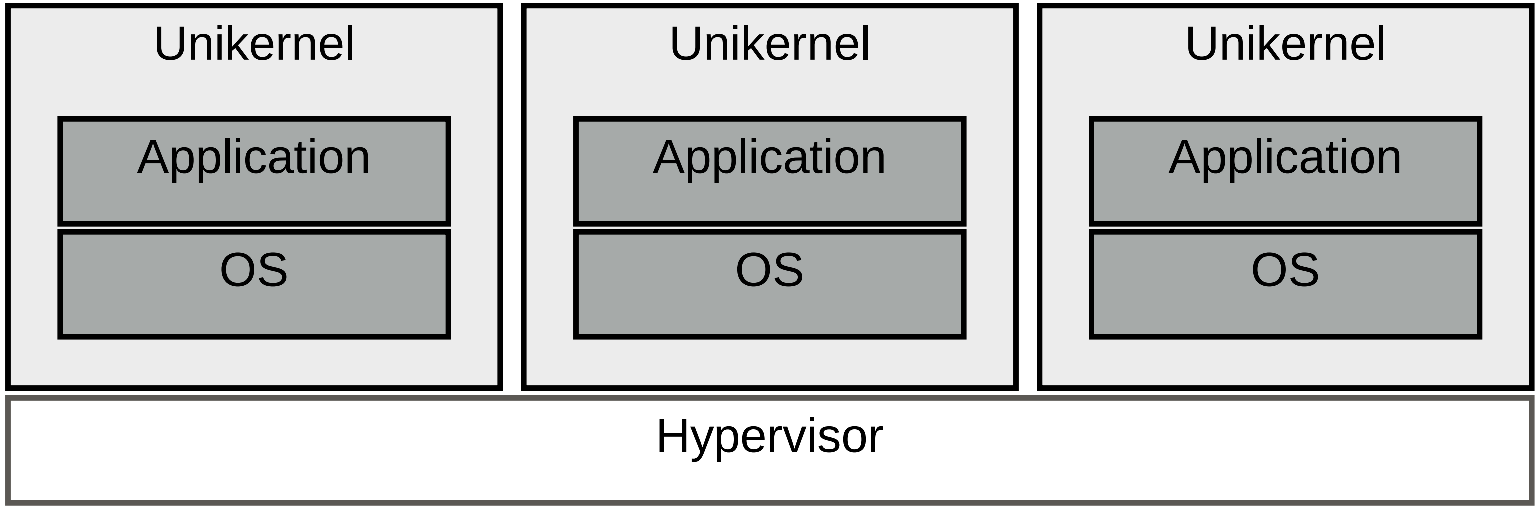
- Run one application per virtual machine
- One process per application
- Everything compiled into a VM image
- Do not compile unused code

Unikernel, Figure 1



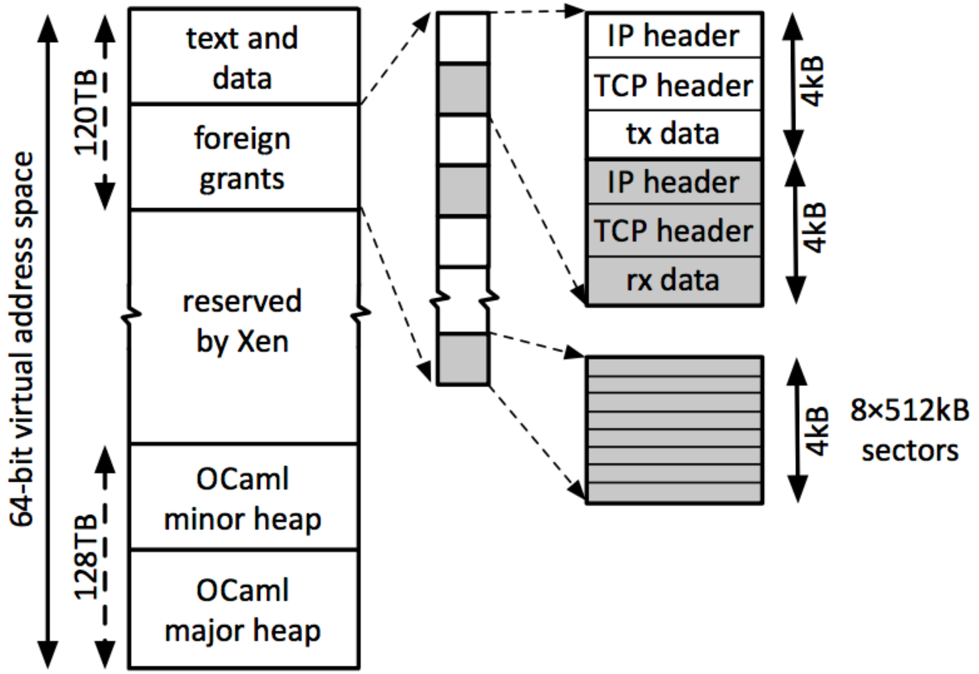
Unikernel

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



Mirage

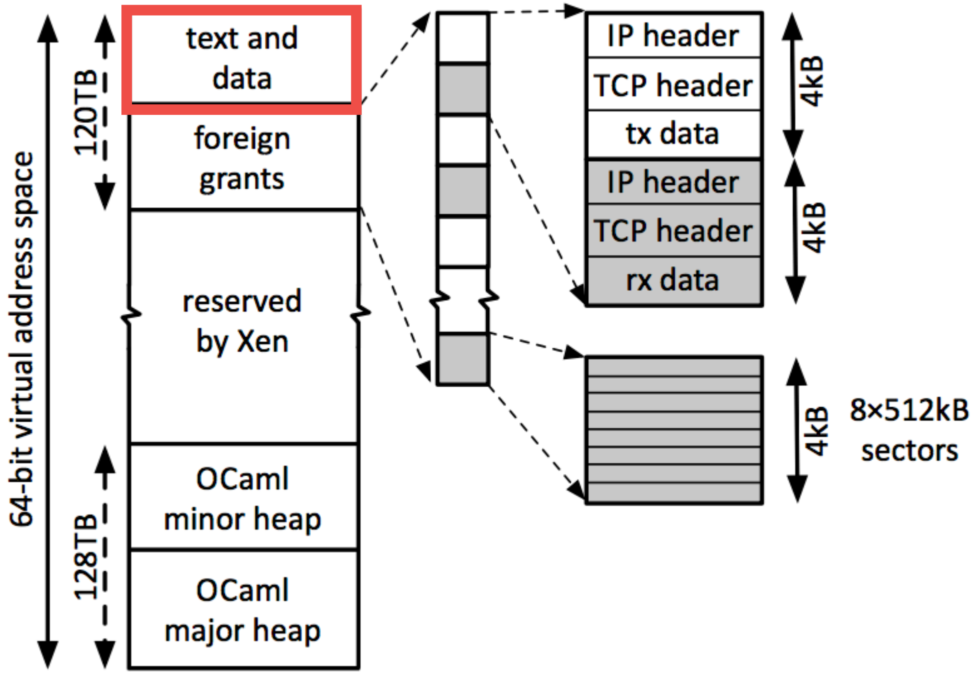
- Produces unikernels
- Compiles OCaml code to Xen VM image
- 4 main components
  - Text + Data segment
  - Foreign Grants
  - Minor Heap
  - Major Heap



Unikernel, Figure 2

Text and Data

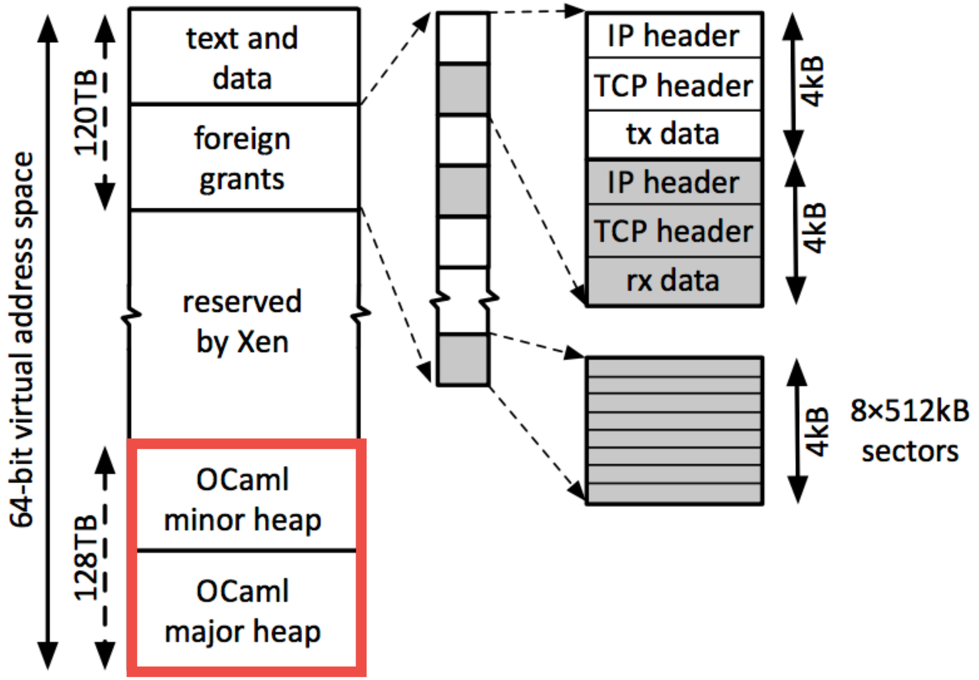
- OCaml Runtime
- PVBoot
  - Initializes VM



Unikernel, Figure 2

HEAP

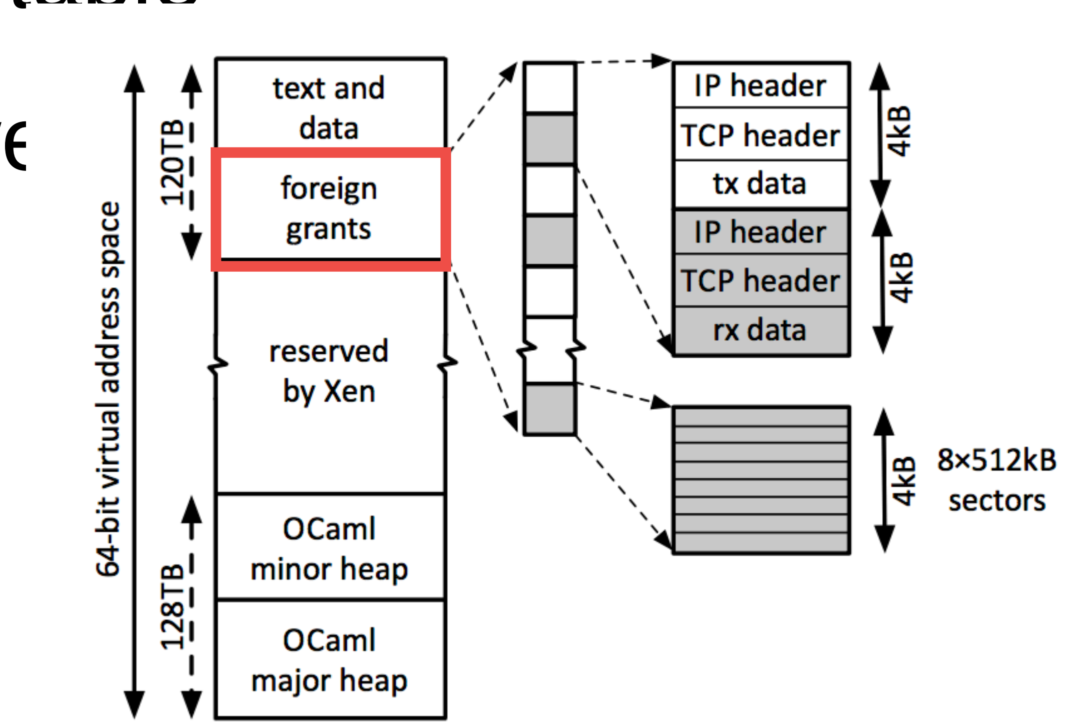
- Minor Heap
  - Short lived values in VM
  - Fast
- Major Heap
  - Long lived values



Unikernel, Figure 2

Foreign Grants

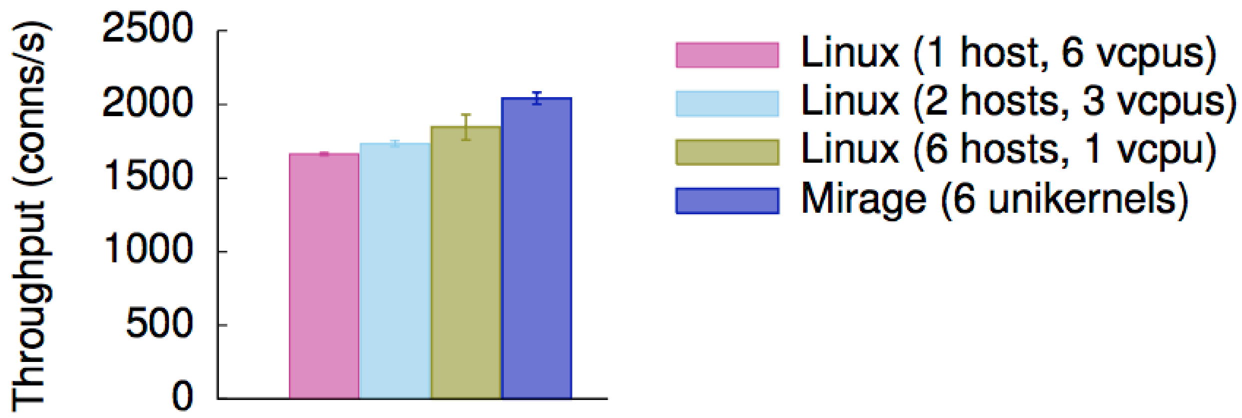
- Used for VM communication
- Write data to a grant table
- Exchange table betwe



Unikernel, Figure 2

ApACHE BENCHMARK

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



Unikernel, Figure 2

Exokernel versus Unikernel

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

Next Time

- Read and write review:
  - **The Origin of the VM/370 Time-Sharing System**, R. J. Creasy, *In IBM Journal of Research and Development*, 25(5):483-490, September 1981.
  - **Xen and the Art of Virtualization, Paul Barham, Boris Dragovic**, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. *19th ACM symposium on Operating systems principles (SOSP)*, October 2003, page 164--177.

Next Time

- MP1 part 2 due Friday
- Project Survey Paper proposals due next week
- Presentation schedule
- Check website for updated schedule