A Tale of Two Abstractions

The Case for Object Space

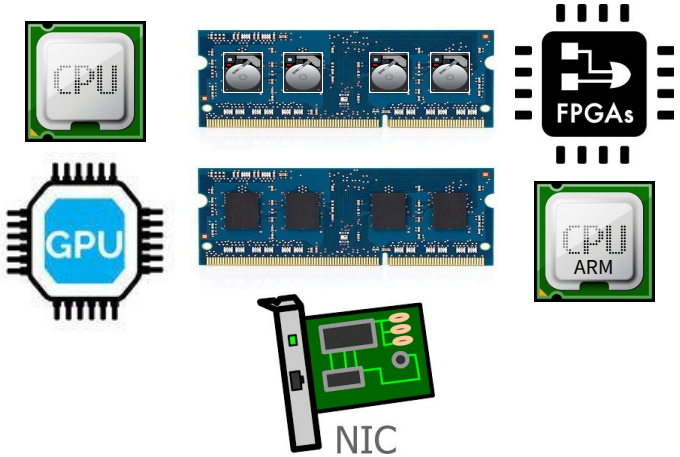
Daniel Bittman Peter Alvaro Darrell D. E. Long Ethan L. Miller

UC Santa Cruz

HotStorage '19
2019-07-08

Hardware Trends

Byte-addressable Non-volatile Memory
(actual implementations may vary)



Multiplicity of Computing Devices and Heterogeneous Memory

Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency		
In-memory Data Structures		
Data Lifetime and Persistent Data References		
Memory Heterogeneity and Data Movement		

Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures		
Data Lifetime and Persistent Data References		
Memory Heterogeneity and Data Movement		

Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures		✓
Data Lifetime and Persistent Data References		
Memory Heterogeneity and Data Movement		

~~Serialization Cost + Two different data representations~~

Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures	X	✓
Data Lifetime and Persistent Data References		
Memory Heterogeneity and Data Movement		

Hardware's Needs vs. Software's Needs

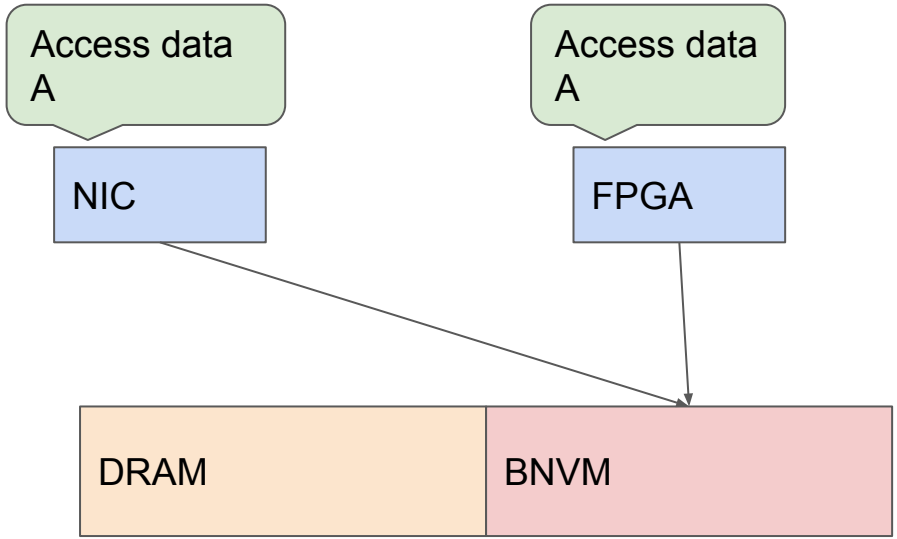
New challenges

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures	X	✓
Data Lifetime and Persistent Data References	X	✓
Memory Heterogeneity and Data Movement		

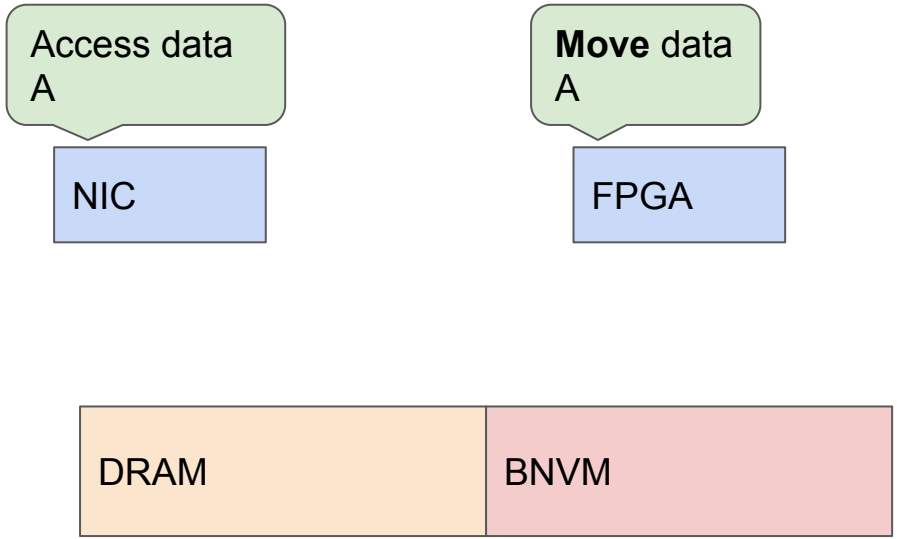
Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures	X	✓
Data Lifetime and Persistent Data References	X	✓
Memory Heterogeneity and Data Movement	✓	

Heterogeneity and Autonomy



Data Movement



Hardware's Needs vs. Software's Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures	X	✓
Data Lifetime and Persistent Data References	X	✓
Memory Heterogeneity and Data Movement	✓	X

In short...

Software cares about **long-lived data relationships**, even across program runs.

Hardware must provide **consistent data access**, even if it moves in memory.

Virtual memory is the **wrong** abstraction.

Virtual memory is fine.

Software is easier to change than hardware

Two Abstractions

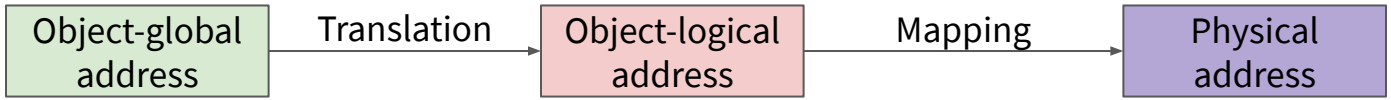
Global Object Space

Provides long-term data references (persistent pointers)

Logical Object Space

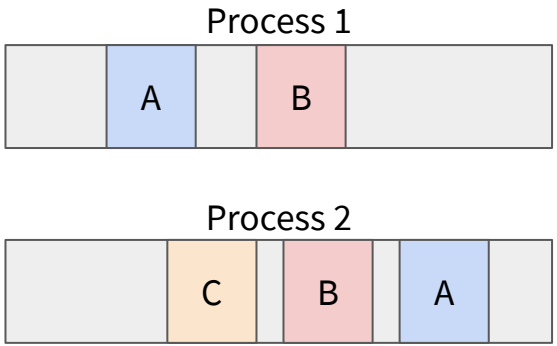
Abstracts physical location from hardware to enable correct access to objects

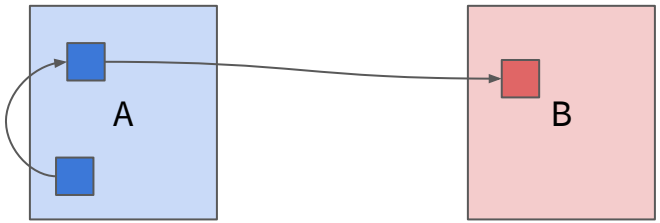
Common ground: organize data into *objects*.



Global Object Space: Abstract *References*

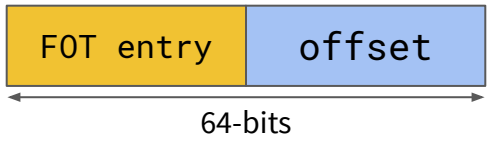
Persistent data should be operated on *directly* and *like memory*





Pointers may be *cross-object*: referring to data within a different object

Global Object Space: Abstract *References*

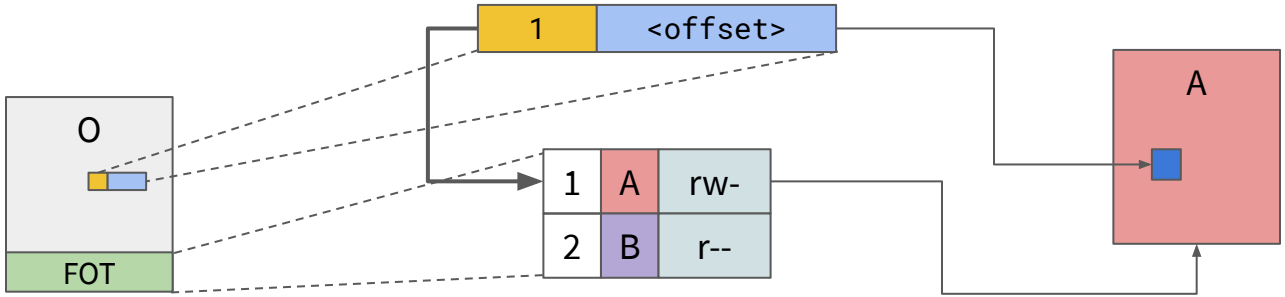


Foreign Object Table

1	object ID or Name	Name Resolver	flags
2	object ID or Name	Name Resolver	flags
	...		

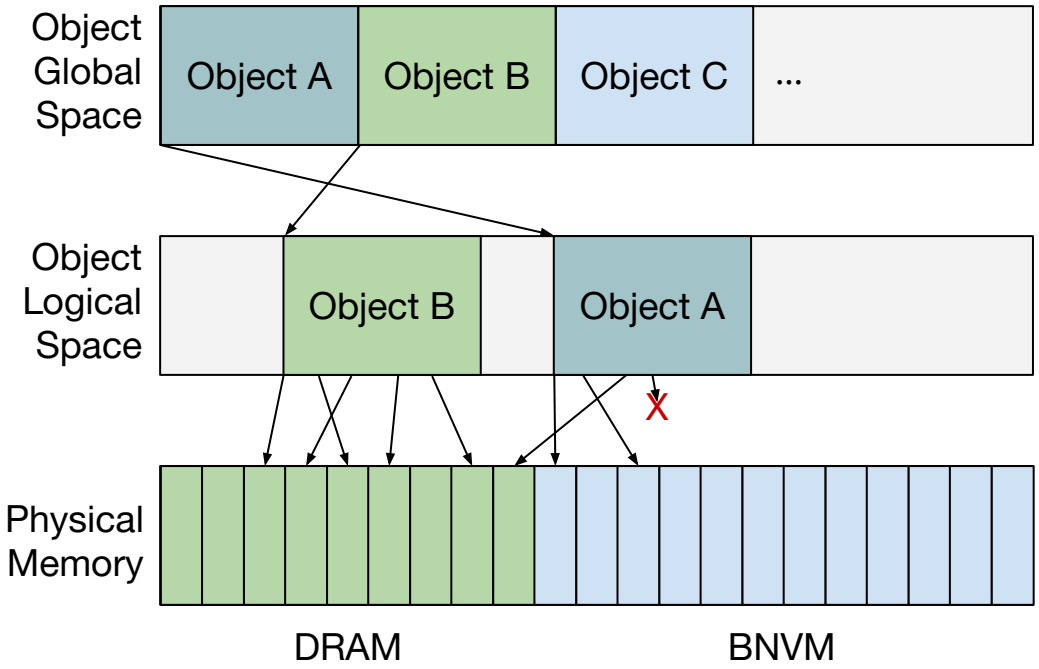
Object Layout

Global Object Space: Abstract *References*



FOT entry of >0 means “cross-object”—points to a different object.

Logical Object Space: Abstract *Location*



Software sees global space of ALL objects.

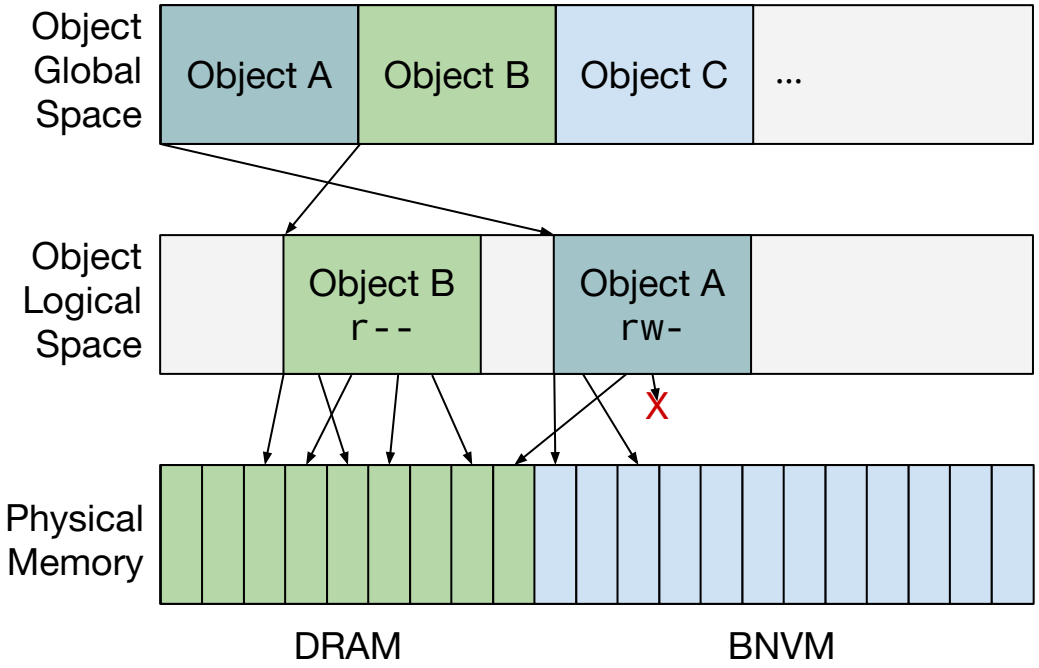
Hardware sees logical space of currently accessible, active objects

Implications for Operating Systems

Greatly simplified mapping management

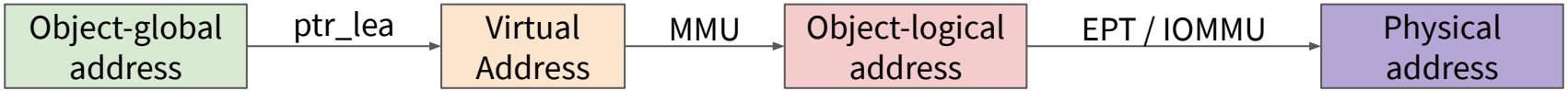
The kernel is “out of the way”

Implications for Operating Systems



Security Contexts!

Implementation Details



EPT Drawbacks

- Longer page walking
- Additional switching

Optimizations

- `vmfunc` enables faster EPT switching
- Virtualization exceptions allow guest to handle EPT violations

Where do we go from here?

We're building a new OS, Twizzler, around BNVM and heterogeneous memory.

Initial results show negligible impact from using VT-x hardware, and a very small overhead on translating persistent pointers.

We plan to explore implications for distributed computing, distributed memory and storage, and resumability under power cycles.

Remember: Different Needs

Consideration	Hardware	Software
Latency	✓	✓
In-memory Data Structures	X	✓
Data Lifetime and Persistent Data References	X	✓
Memory Heterogeneity and Data Movement	✓	X

Remember: Two Abstractions

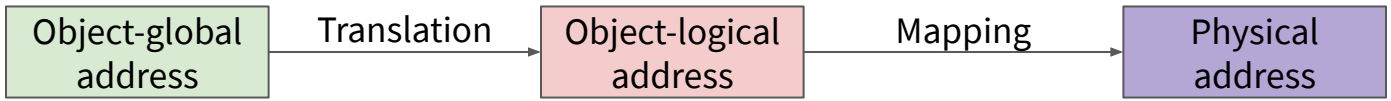
Global Object Space

Provides long-term data references (persistent pointers)

Logical Object Space

Abstracts physical location from hardware to enable correct access to objects

Common ground: organize data into *objects*.



Remember: Implications

Software should operate on in-memory data structures.

Hardware should have abstracted view of memory.

We’re building Twizzler to explore the implications.

Thank You!

Questions / Discussion

Daniel Bittman
dbittman@ucsc.edu
@danielbittman

Peter Alvaro
palvaro@ucsc.edu

Darrell D. E. Long
darrell@ucsc.edu

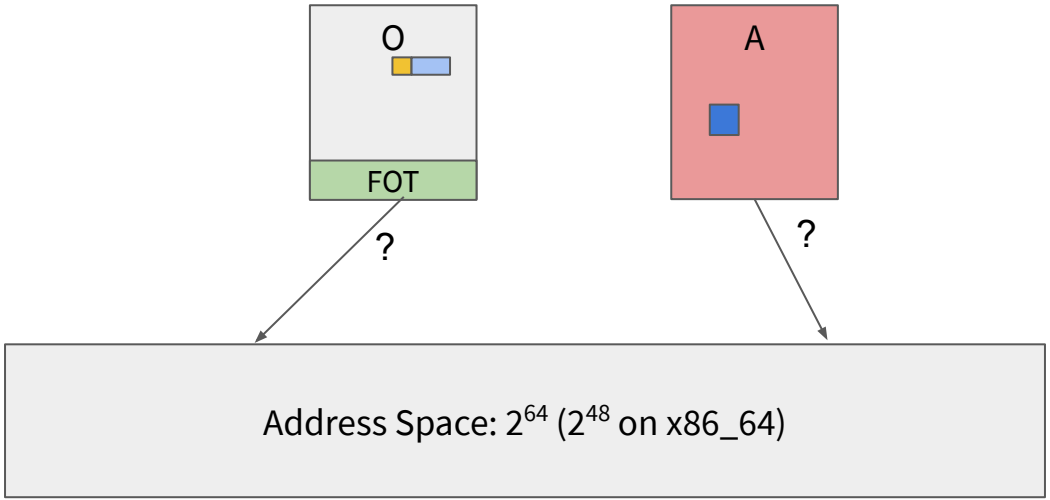
Ethan L. Miller
elm@ucsc.edu

Implementation of Global Space

Mapping should be transparent to application.

The virtual address space abstraction does not fit with the object:offset model

LibOS handles address space management



Implementation of Global Object Space