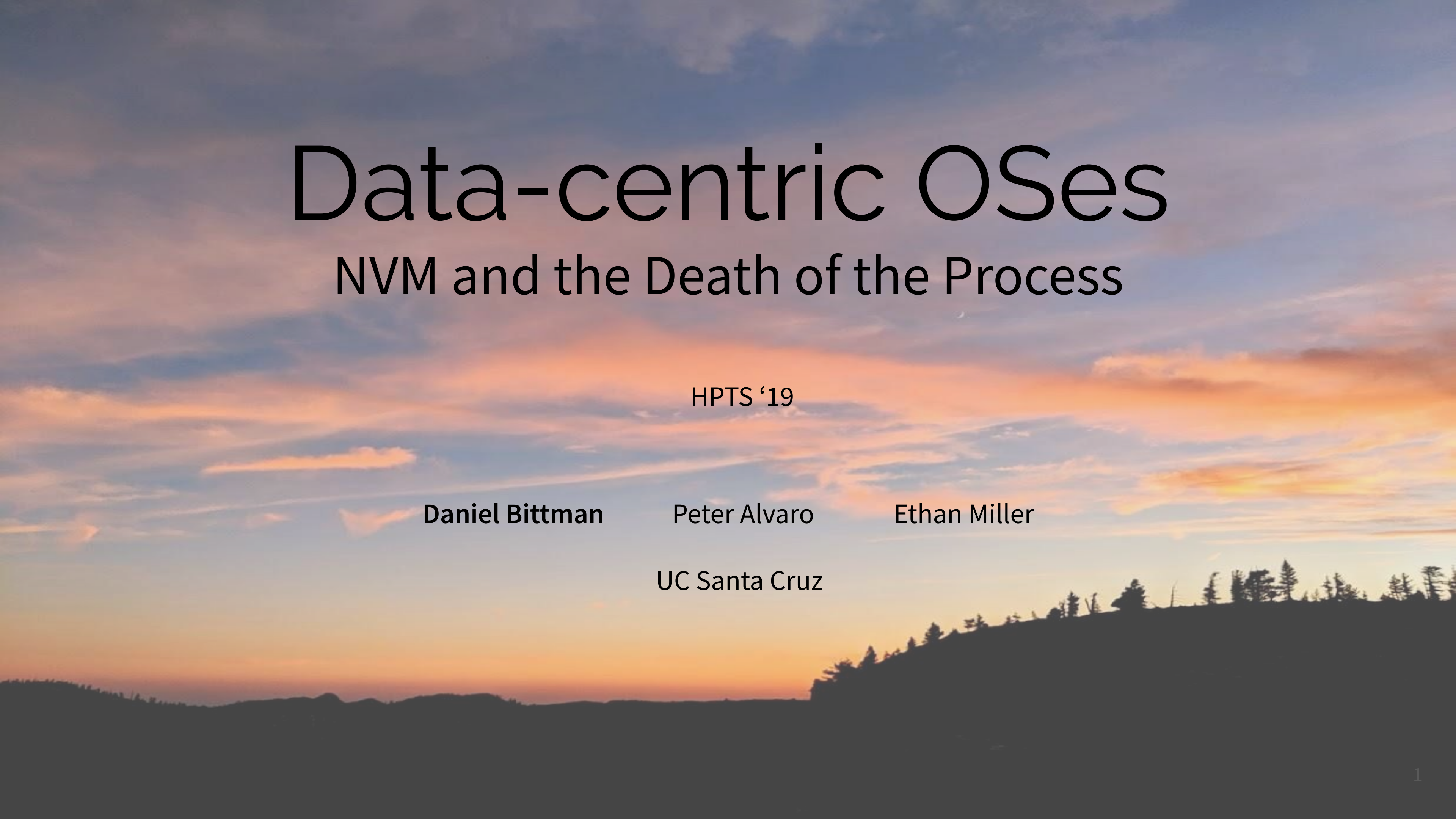


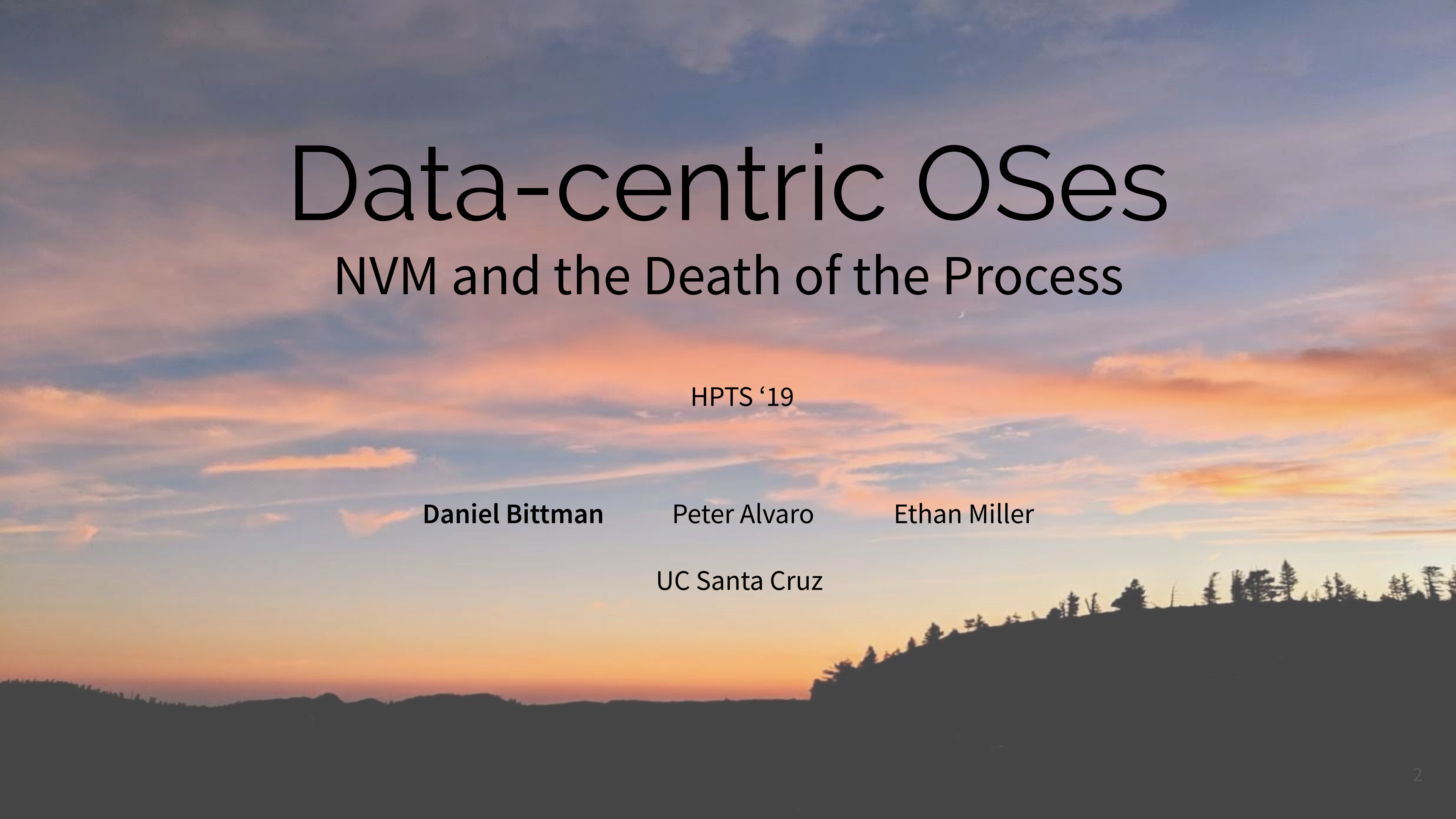
Data-centric OSes

NVM and the Death of the Process

HPTS ‘19

Daniel Bittman Peter Alvaro Ethan Miller

UC Santa Cruz



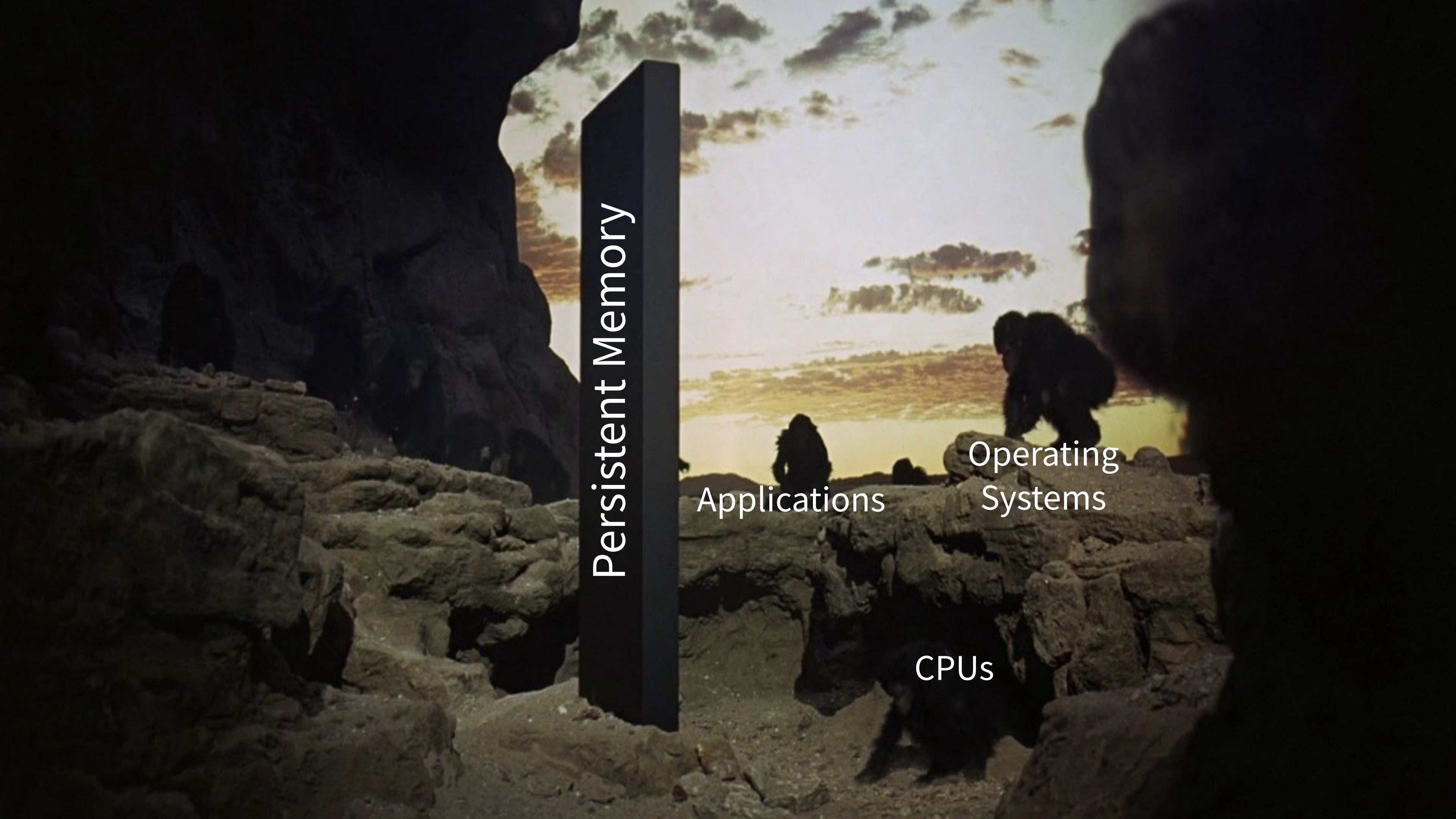
Data-centric OSes

NVM and the Death of the Process

HPTS ‘19

Daniel Bittman Peter Alvaro Ethan Miller

UC Santa Cruz


Persistent Memory
Applications
Operating Systems
CPUs

Hardware Trends

(artistic rendering;
actual implementation may vary)

`sys_read`

~300 ns

Growing, becoming persistent

~1 us

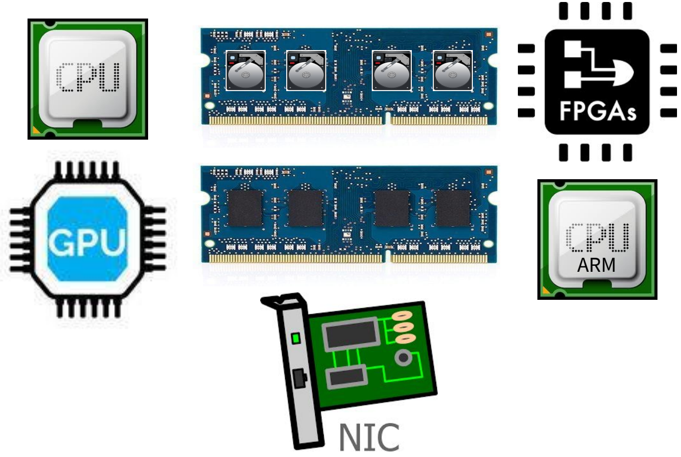
Outdated interface

~1-10 ms

Cannot compute on directly

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

Hardware Trends

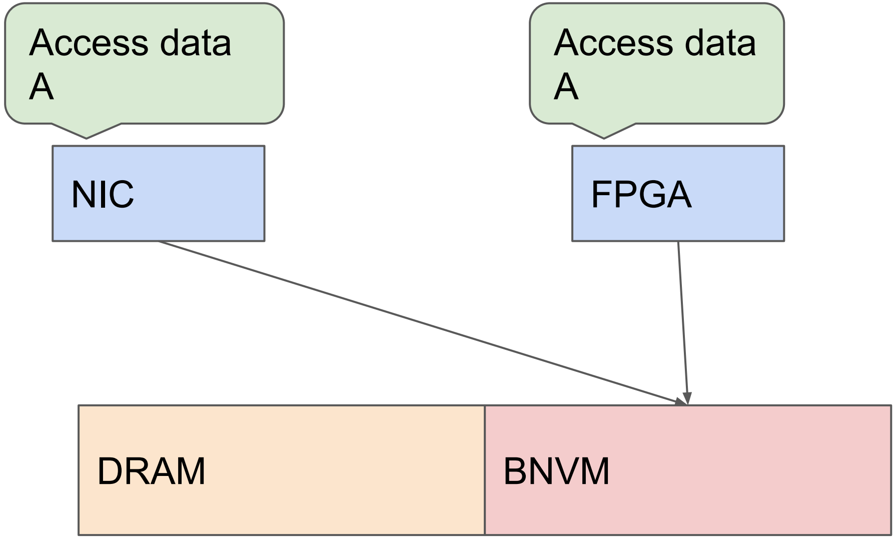


Multiplicity of Computing Devices and Heterogeneous Memory

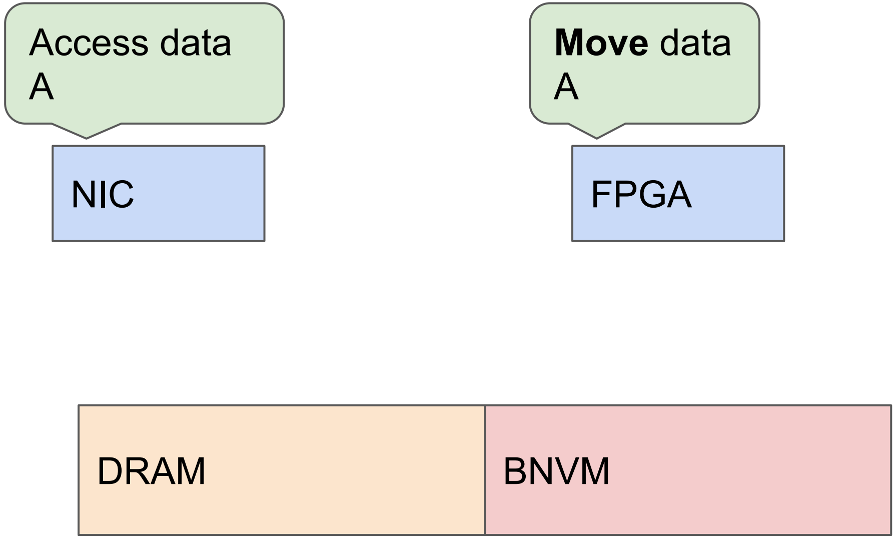
Hardware's Needs vs. Software's Needs

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

Heterogeneity and Autonomy



Data Movement



Access data A

NIC

Move data A

FPGA

DRAM

BNVM

In short…

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

Virtual memory is the **wrong** abstraction.

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

Virtual memory is fine.

Software is easier to change than hardware

Twizzler: A new OS

The kernel is “out of the way”

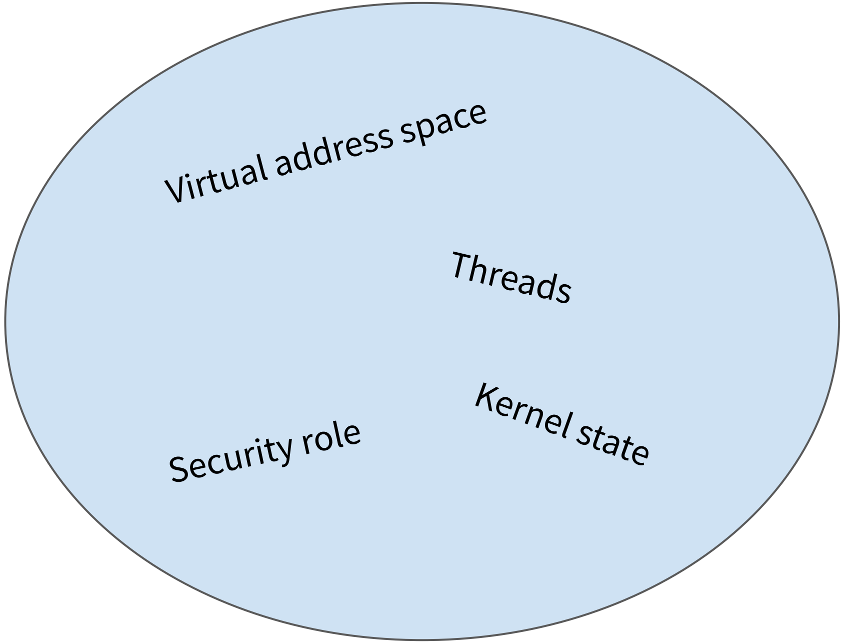
Presents a unified interface for data sharing, security, and persistent pointers

Operating systems COULD BE WORSE, CALVIN.
Operating systems COULD BE A LOT BETTER, TOO!

OS Community

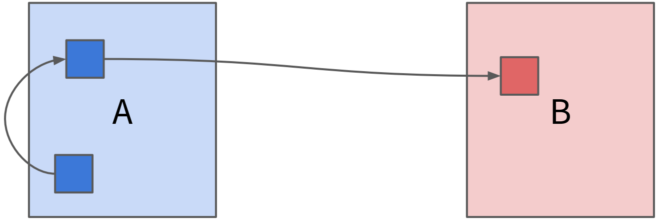
DB Community

The Death of the Process



A global object space

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

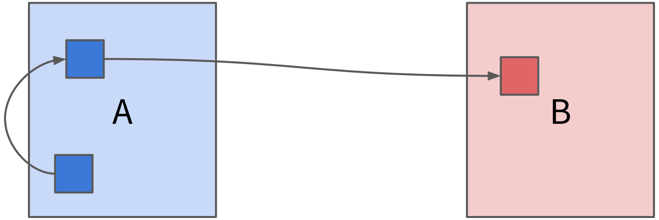


An object is a unit of semantically similar information

E.g. a b-tree, or part of one.

A global object space

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



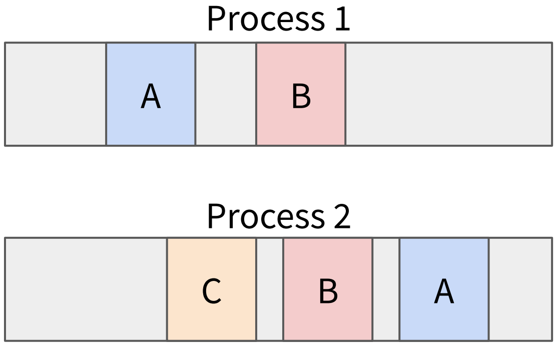
An object is a unit of semantically similar information

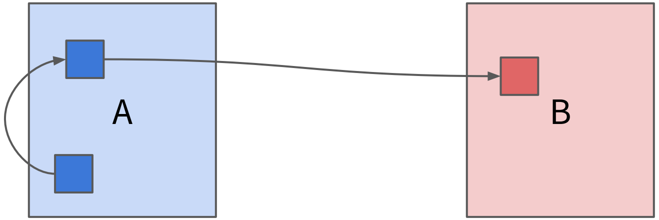
E.g. a b-tree, or part of one.

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

Persistent pointers in Twizzler

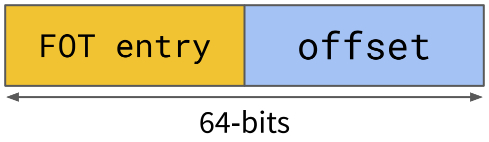
Virtual addresses are the *wrong* abstraction





object-id	offset

Twizzler's pointers



FOT entry | offset

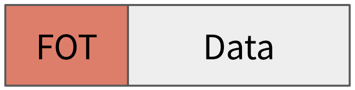
64-bits

Foreign Object Table

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

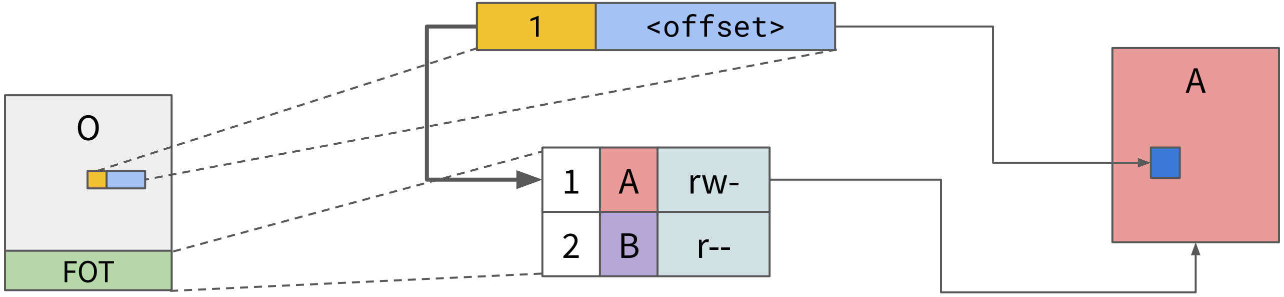
...

Object Layout



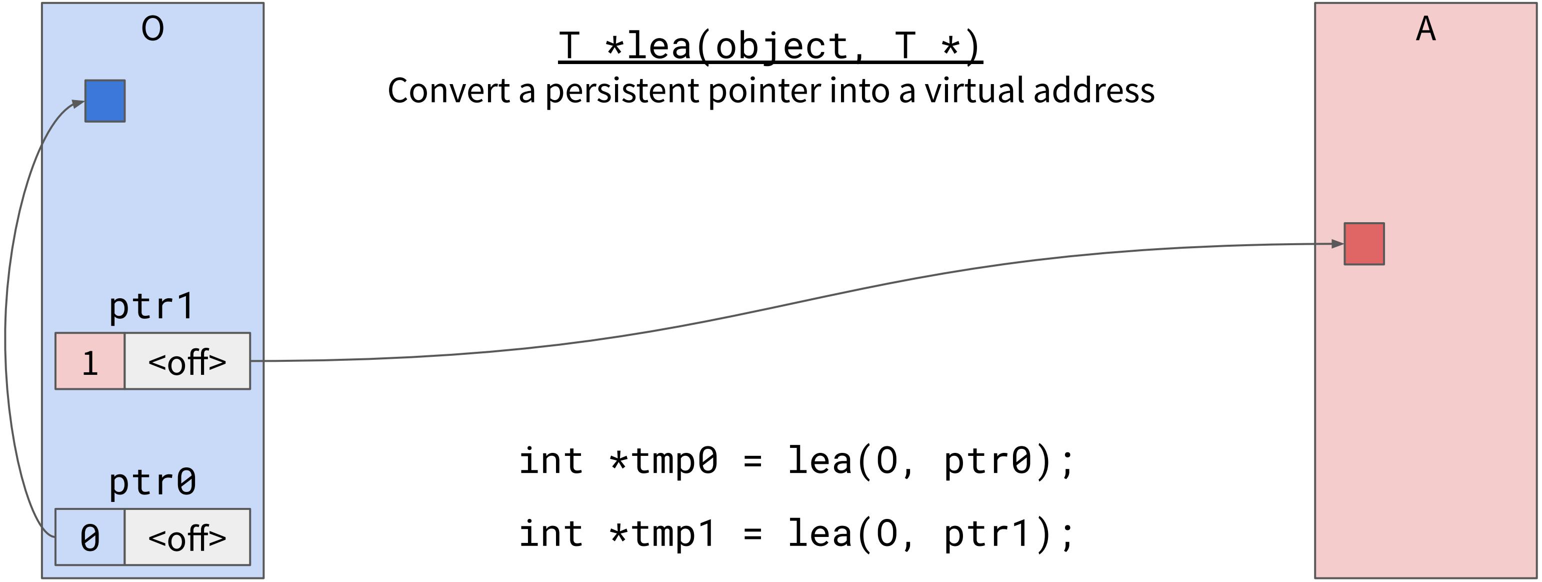
FOT | Data

Example pointer resolution



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

Pointer implementation



T *lea(object, T *)

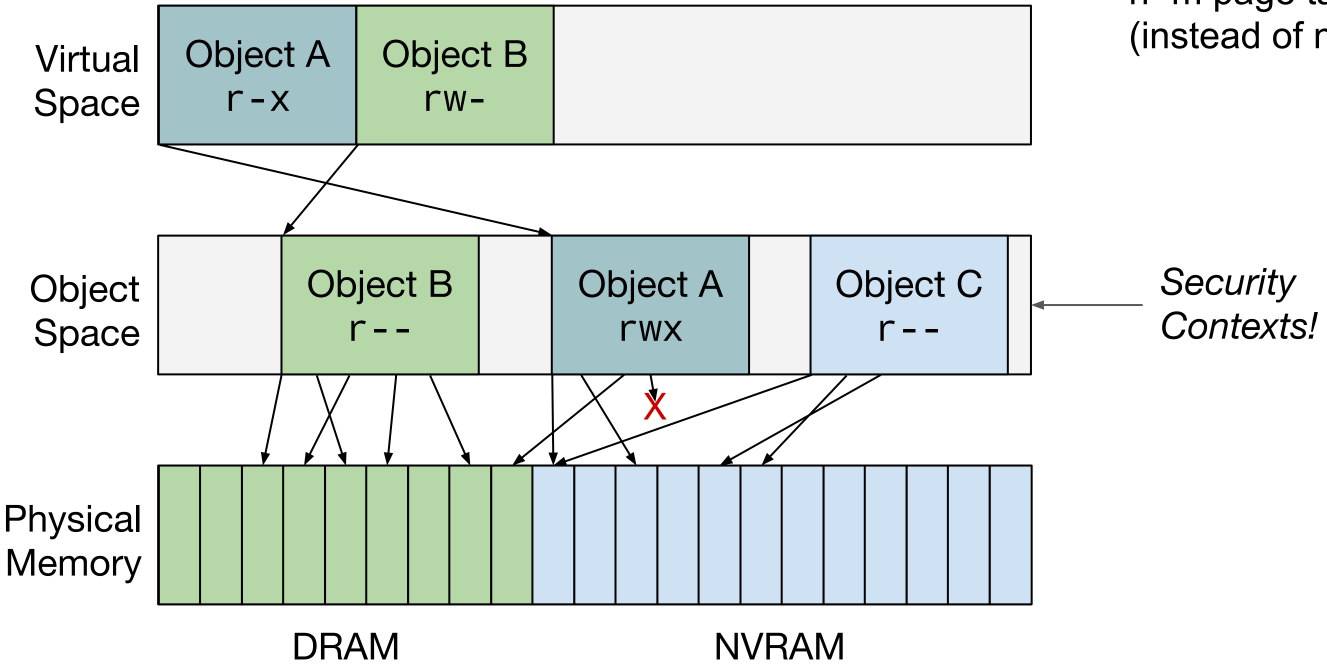
Convert a persistent pointer into a virtual address

```
int *tmp0 = lea(O, ptr0);
int *tmp1 = lea(O, ptr1);
x = *tmp1; y = *tmp0;
```

O’s FOT

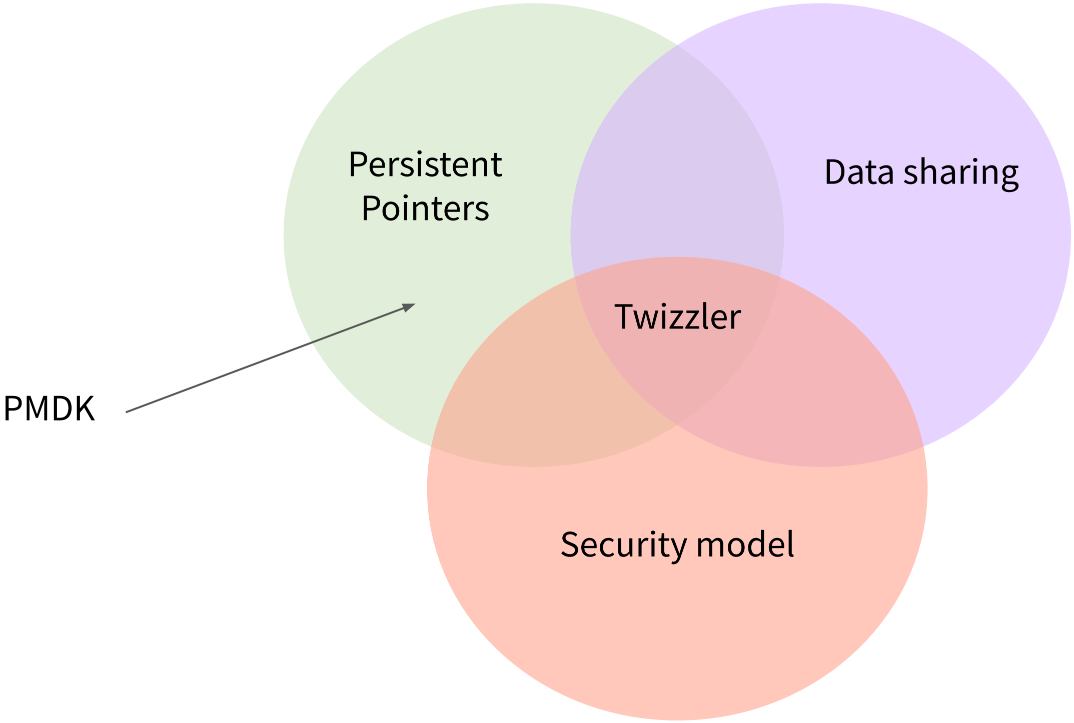
1 → A

Two-level Mapping

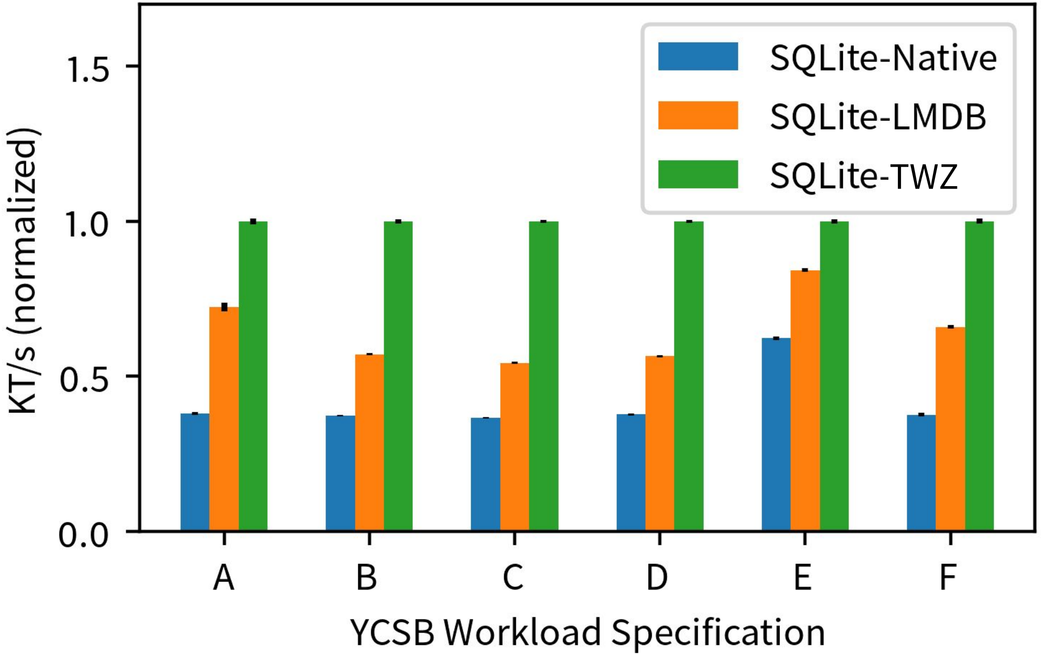


n+m page tables!
(instead of n*m)

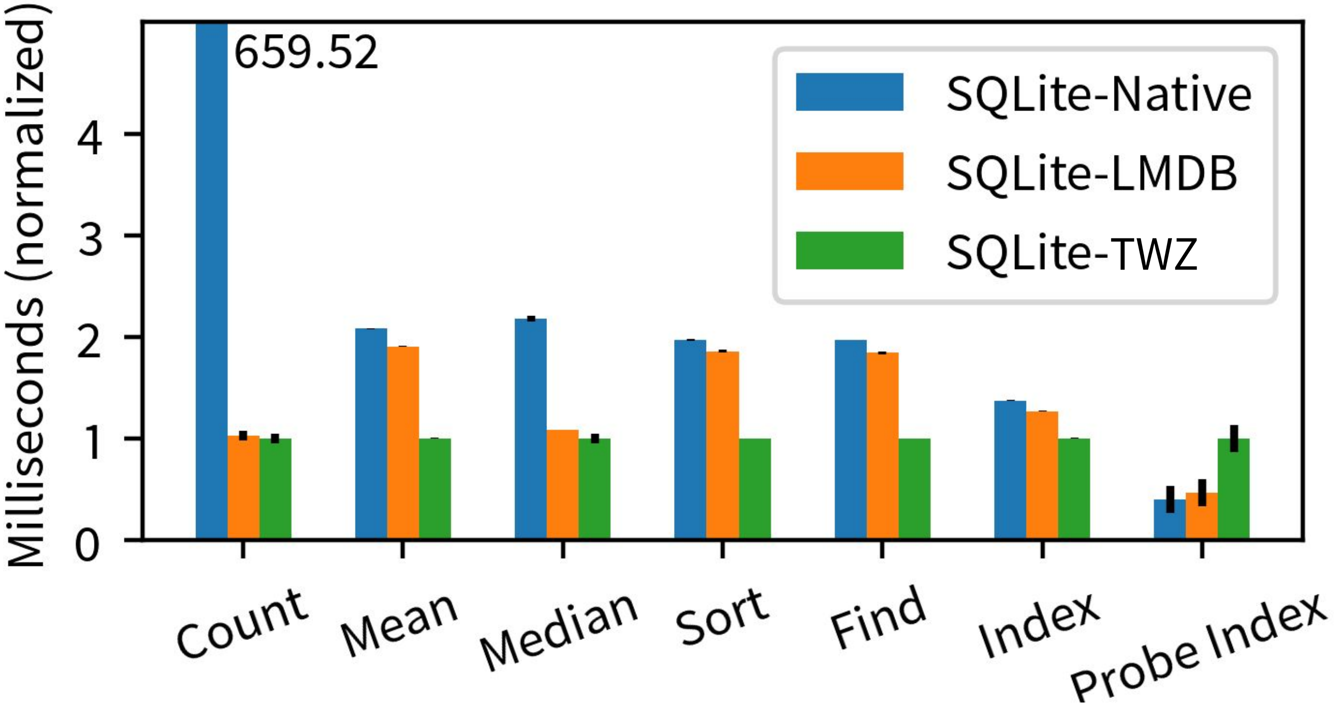
Hey look it's a Venn Diagram



Benchmark: SQLite, throughput

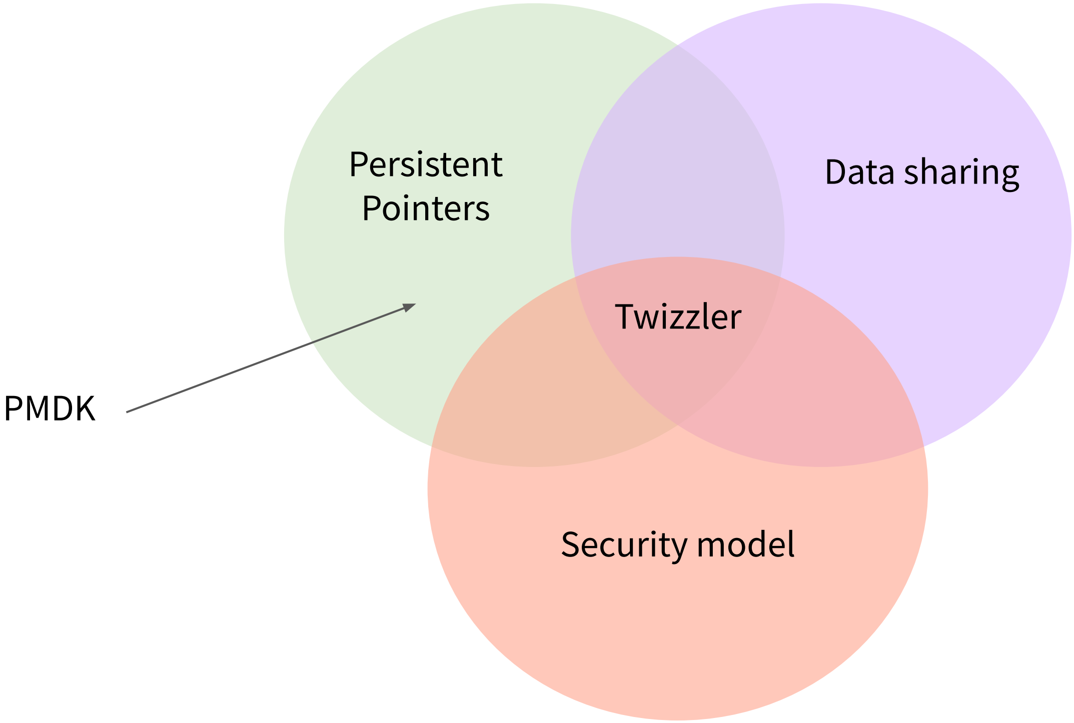


Benchmark: SQLite, latency



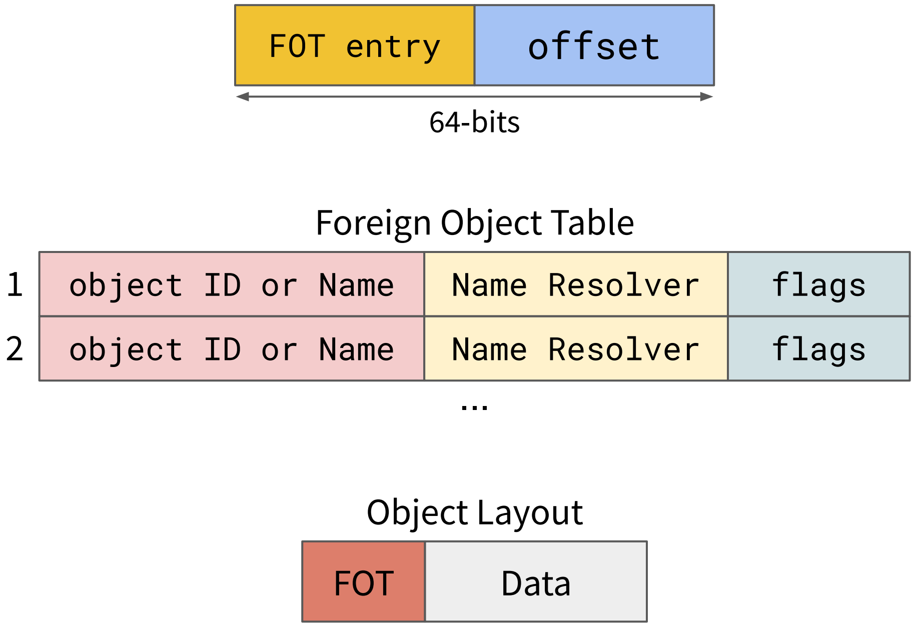
Takeaways - 1

We need to consider persistent memory programming in the context of sharing and security



Takeaways - 2

A *flexible* persistent pointer design enables sharing, upgrades, and late-binding



Foreign Object Table

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

...

Object Layout

Takeaways - 3

We are building Twizzler to explore new programming models for NVM

We must evolve our storage models for new technology

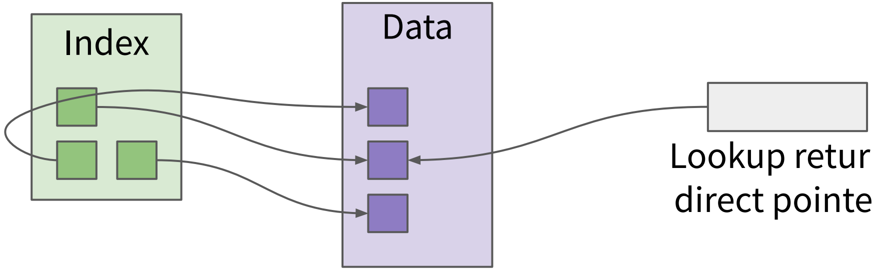
Thank You!
questions / discussion

Daniel Bittman
dbittman@ucsc.edu
@danielbittman

Peter Alvaro
palvaro@ucsc.edu

Ethan L. Miller
elm@ucsc.edu

Case Study: KVS



Lookup returns direct pointers

```
          insert(key, value)
bucket = get_bucket(key)
bucket.ptr = store(Index, value.ptr)
bucket.len = value.len
```

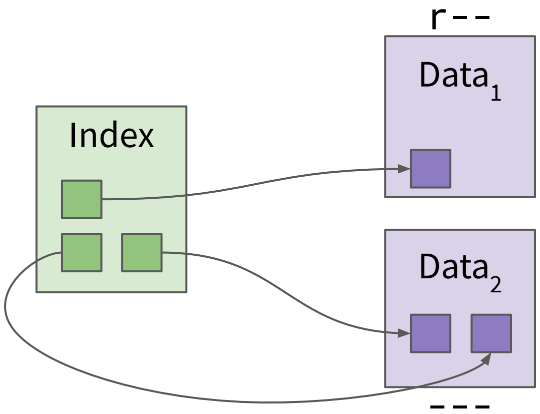
```
          lookup(key)
bucket = get_bucket(key)
item.ptr = lea(Index, bucket.ptr)
item.len = bucket.len
```

250 lines of simple C code is *all you need*

(store is the reverse of lea: convert a virtual address into a persistent pointer)

Cast Study: KVS

Add access control to the existing design

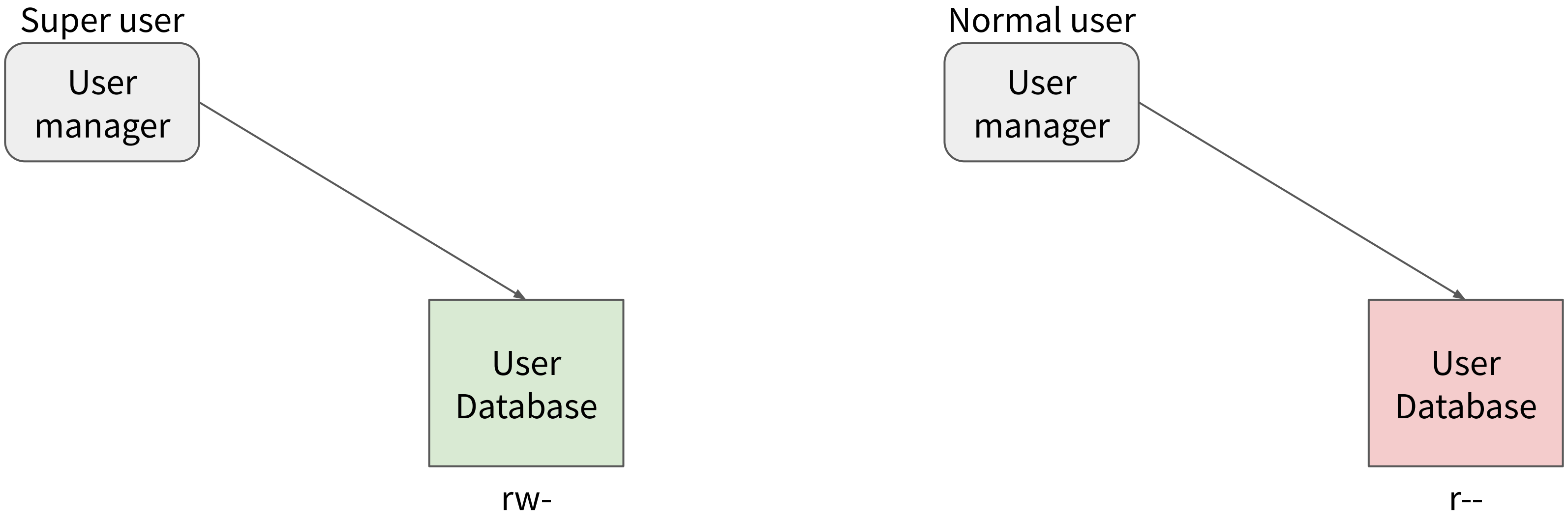


```
bucket = get_bucket(key)
item.ptr = lea(Index, bucket.ptr)
item.len = bucket.len
```

Index points to **different data objects** with **different access control**.

Can hand out pointers to these objects, which can **only be dereferenced with proper permissions**.

Late-binding of access control


Super user
User manager
User Database
rw-
Normal user
User manager
User Database
r--