Twizzler: A Data-Centric OS for Persistent Memory

Daniel Bittman

Peter Alvaro   Pankaj Mehra   Darrell Long   Ethan Miller

Center for Research in Storage Systems

University of California, Santa Cruz

Hardware Trends

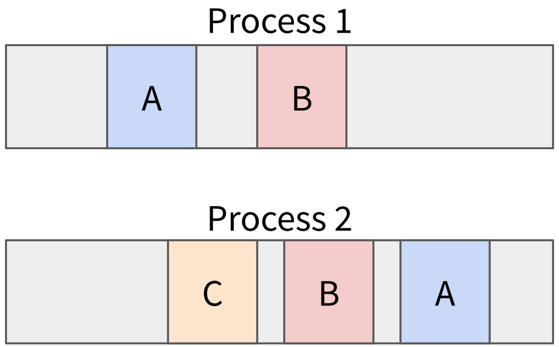
```
sys_read
```

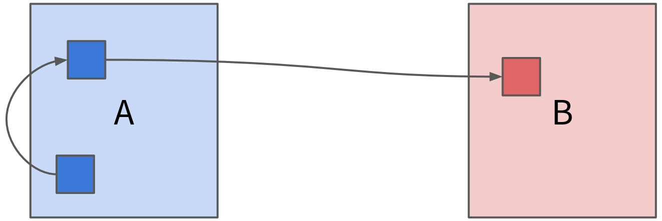
~100-300 ns	~1 us	~1-10 ms
Growing, becoming persistent	Outdated interface	Cannot compute on directly

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

Global Object Space: Abstract *References*

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





Existing approaches?

__POSIX__

Explicit persistence and data access

Multiple forms of data

Kernel involvement

mmap helps, but does not solve the virtual memory problem

__PMDK__

No OS support

Data sharing is hard

Slow pointers

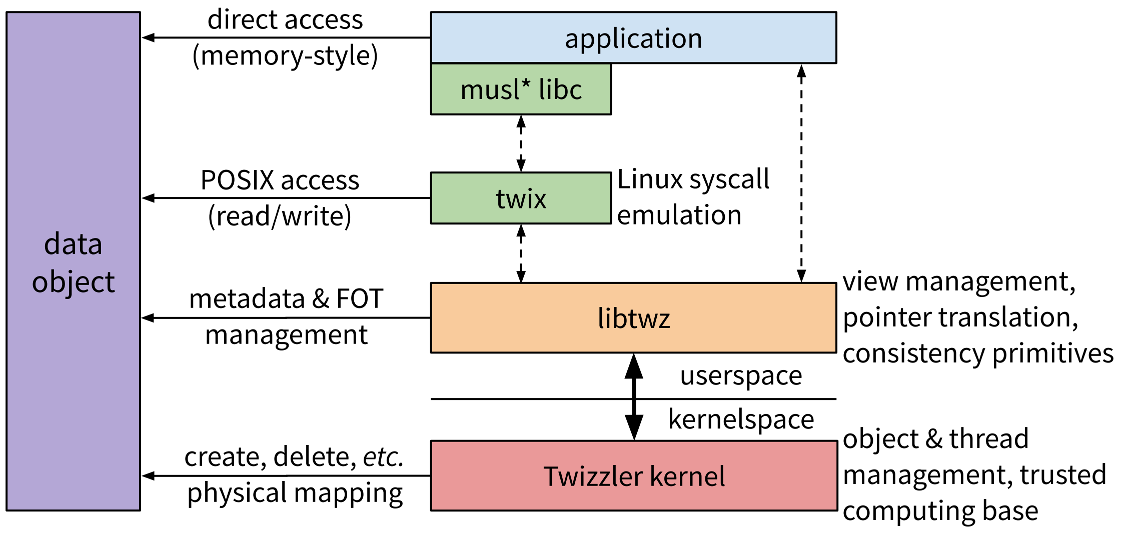
__Twizzler's goals__

Little kernel involvement

Pervasive support (security, sharing)

Low overhead persistent pointers

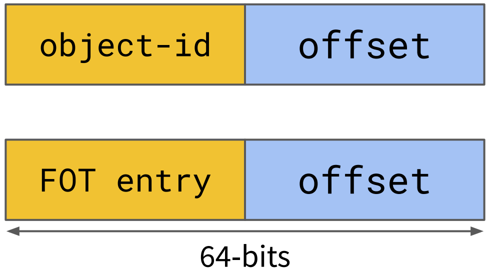
Design Overview


data object
direct access (memory-style)
application
musl* libc
POSIX access (read/write)
twix
Linux syscall emulation
metadata & FOT management
libtwz
view management, pointer translation, consistency primitives
userspace
kernelspace
create, delete, *etc.* physical mapping
Twizzler kernel
object & thread management, trusted computing base

* modified musl to change linux syscalls into function calls

Persistent Pointers

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

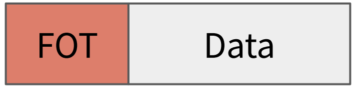


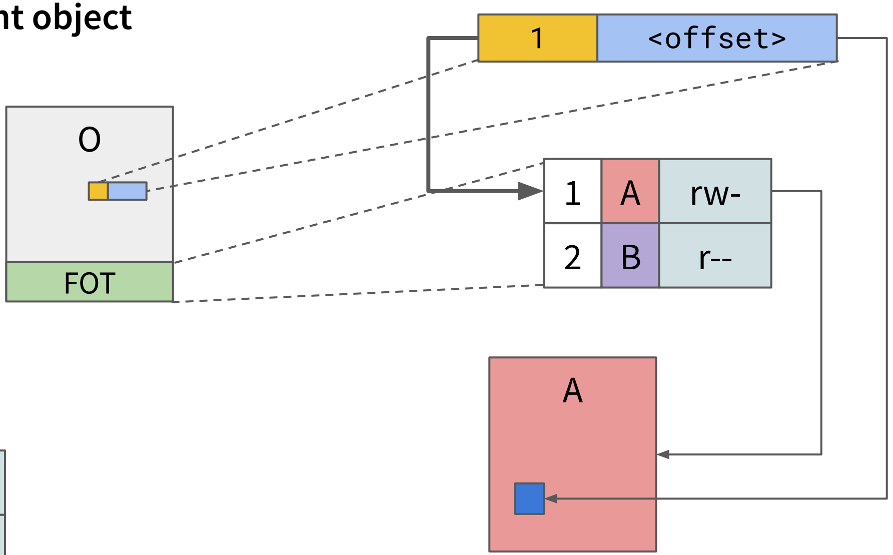
Foreign Object Table

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

...

Object Layout





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

Implications for Data and Sharing

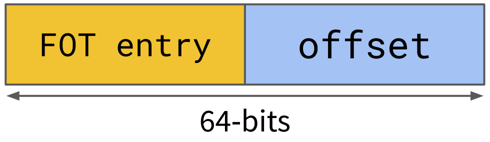
Objects are self-contained

Persistent pointers are based on *identity* not *location*

Persistent pointers can be operated on generically

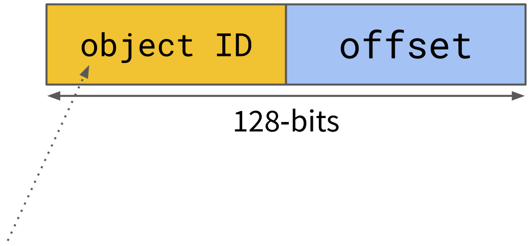
Objects can be *easily shared*

Pointers in Twizzler



Coordination free sharing

Pointers in PMDK



64-bit IDs require global coordination or collision management

Consistency and Security

Cryptographically signed capabilities for access control

The kernel cannot *create* capabilities, but it can (must) verify them.

All enforcement must be done by hardware.

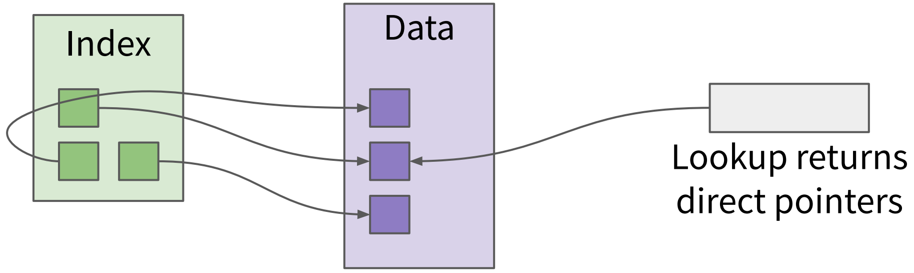
Implementation

More details available at twizzler.io

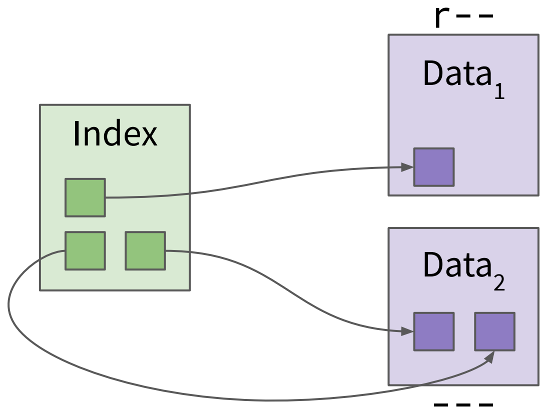
Evaluation Goals

Programmability, not performance (though, performance where we can get it)

Case Study: KVS



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



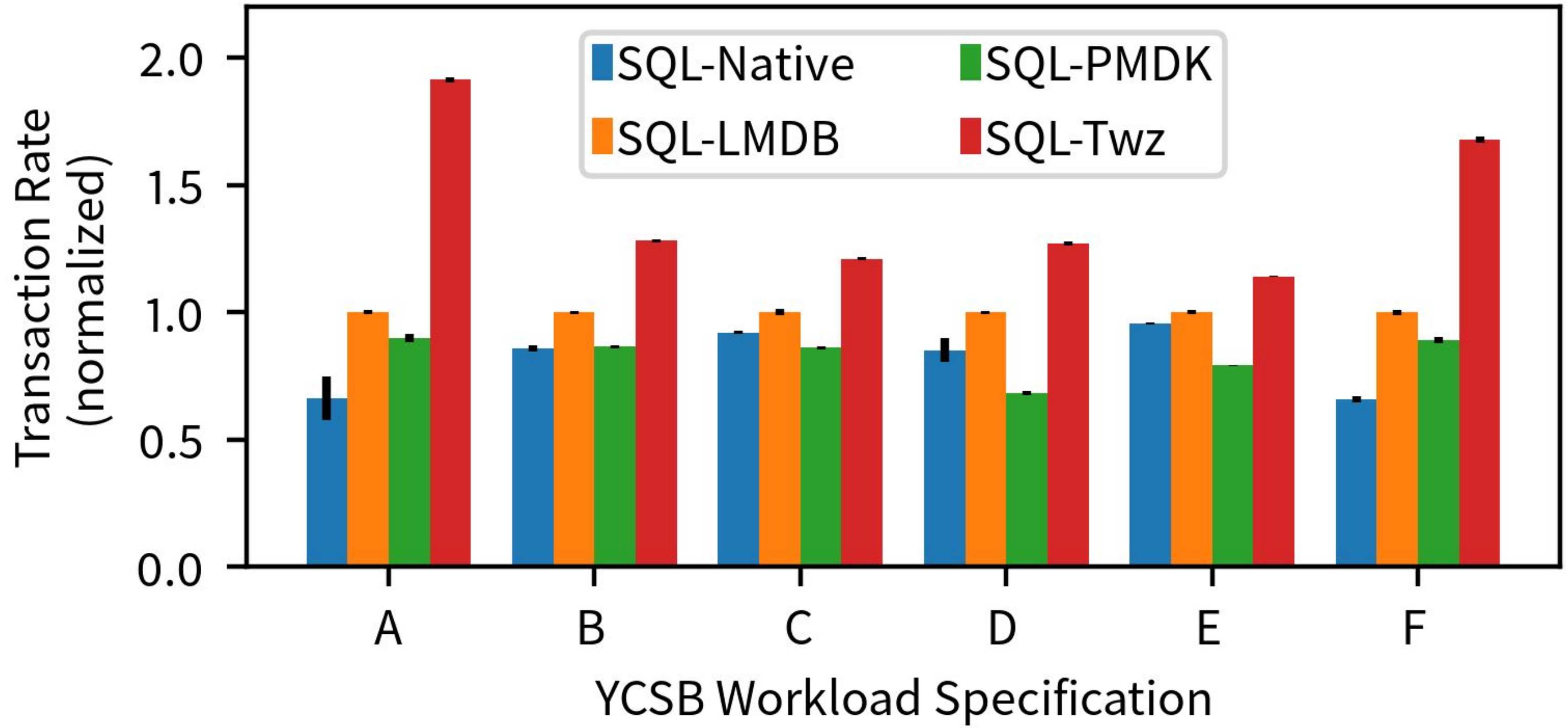
Evaluation

Dell R640 Servers with Intel Optane DC

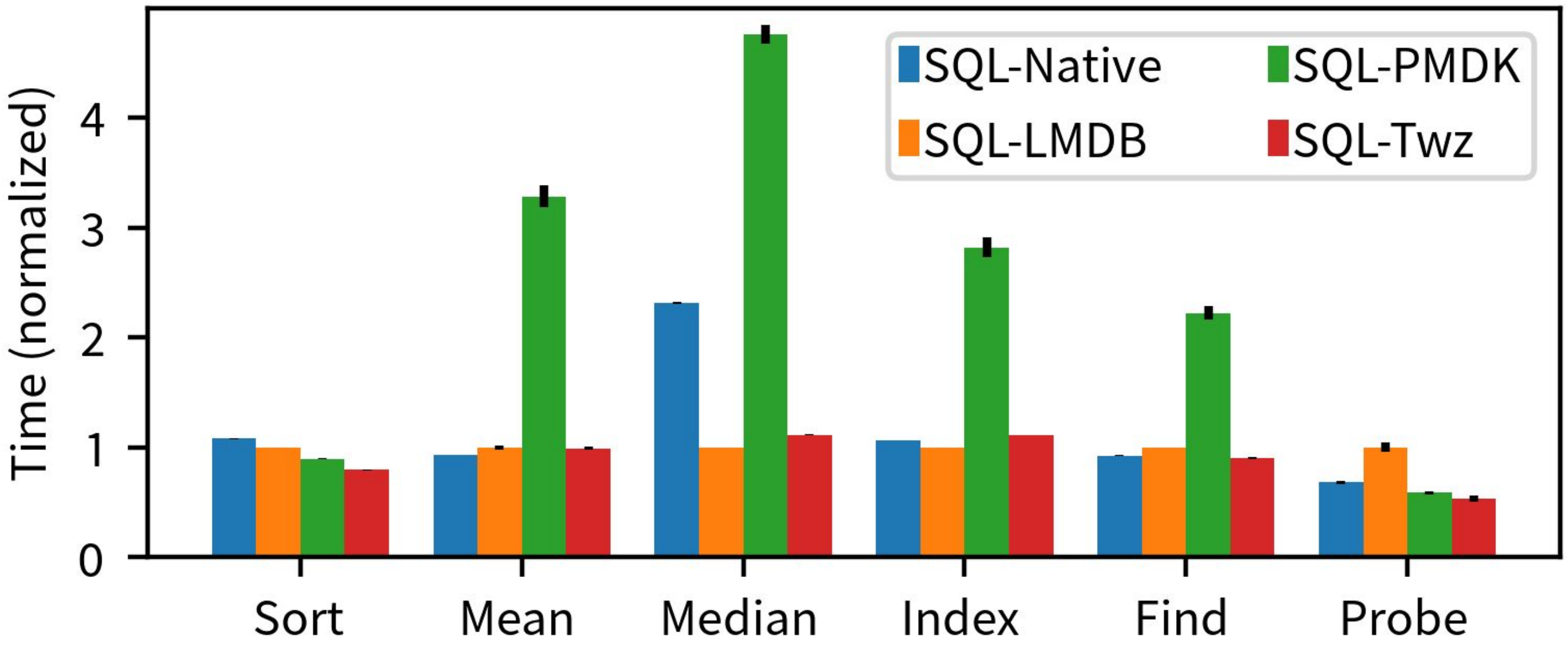
Ported SQLite to Twizzler and to PMDK

Compared to SQLite “native” and SQLite “LMDB” (mmap)

Performance: SQLite

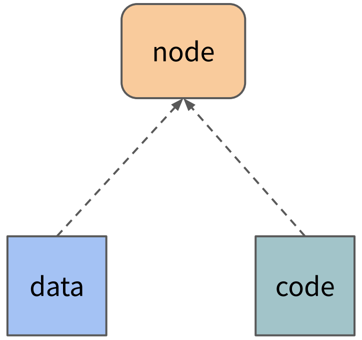


Performance: SQLite

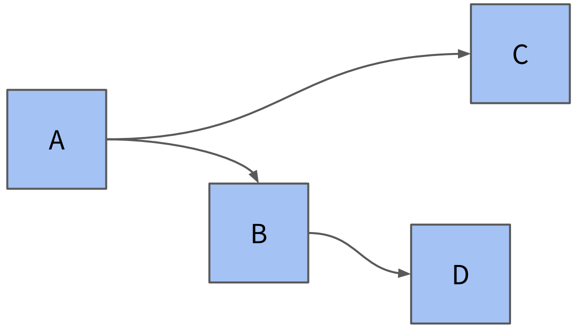


Future Work: Distributed Twizzler

It's a rendezvous problem



Explicit Relationships and the Object Graph



Conclusion

Center for Research in Storage Systems

Operating systems must evolve to support persistent data programming models directly

Cross-object pointers allow us to realize the power of UNIX in a data-centric model

Twizzler provides benefits for both NVM and traditional systems

Thank You! Questions?