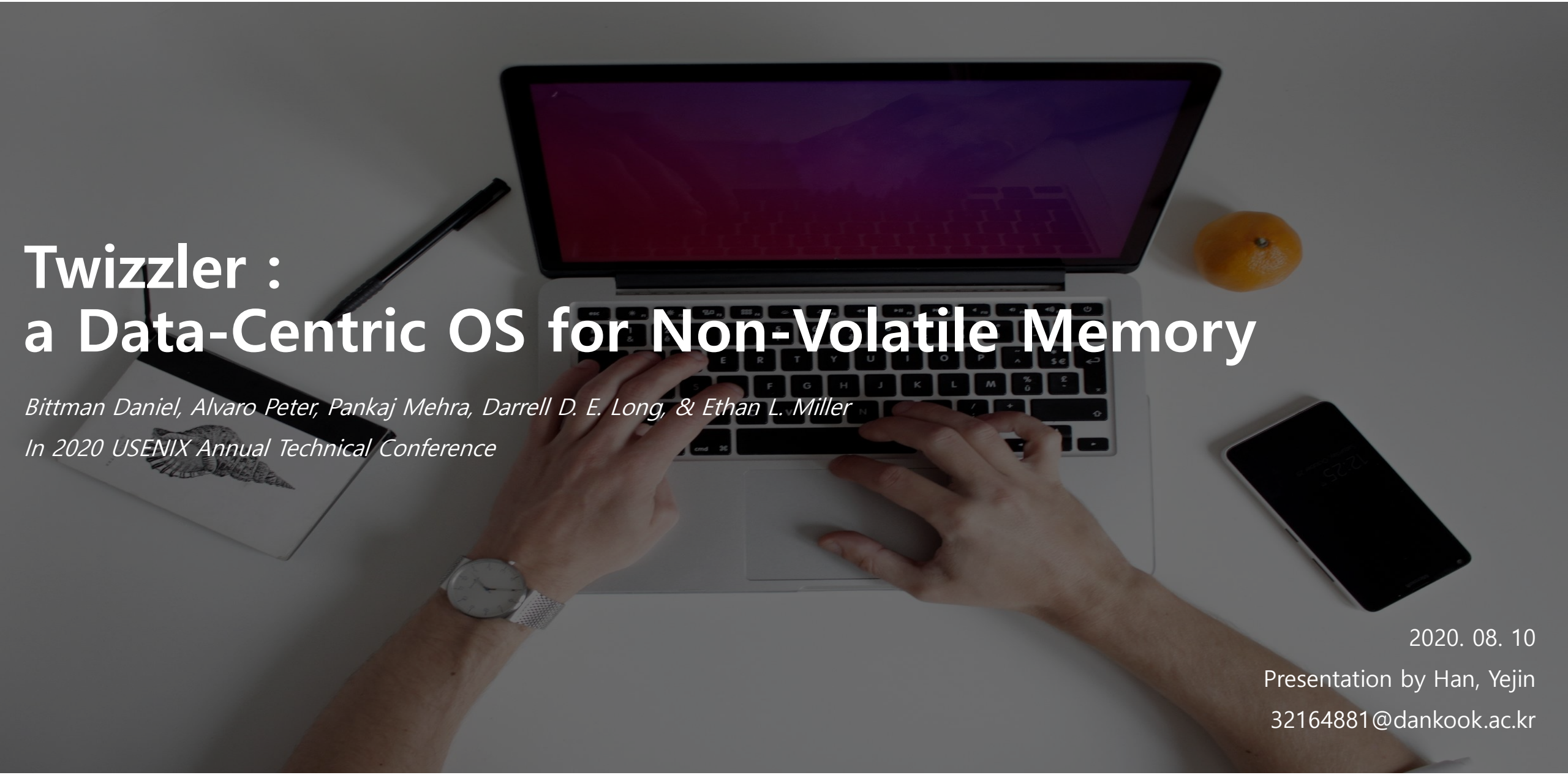


Twizzler : a Data-Centric OS for Non-Volatile Memory

Bittman Daniel, Alvaro Peter, Pankaj Mehra, Darrell D. E. Long, & Ethan L. Miller

In 2020 USENIX Annual Technical Conference

2020. 08. 10

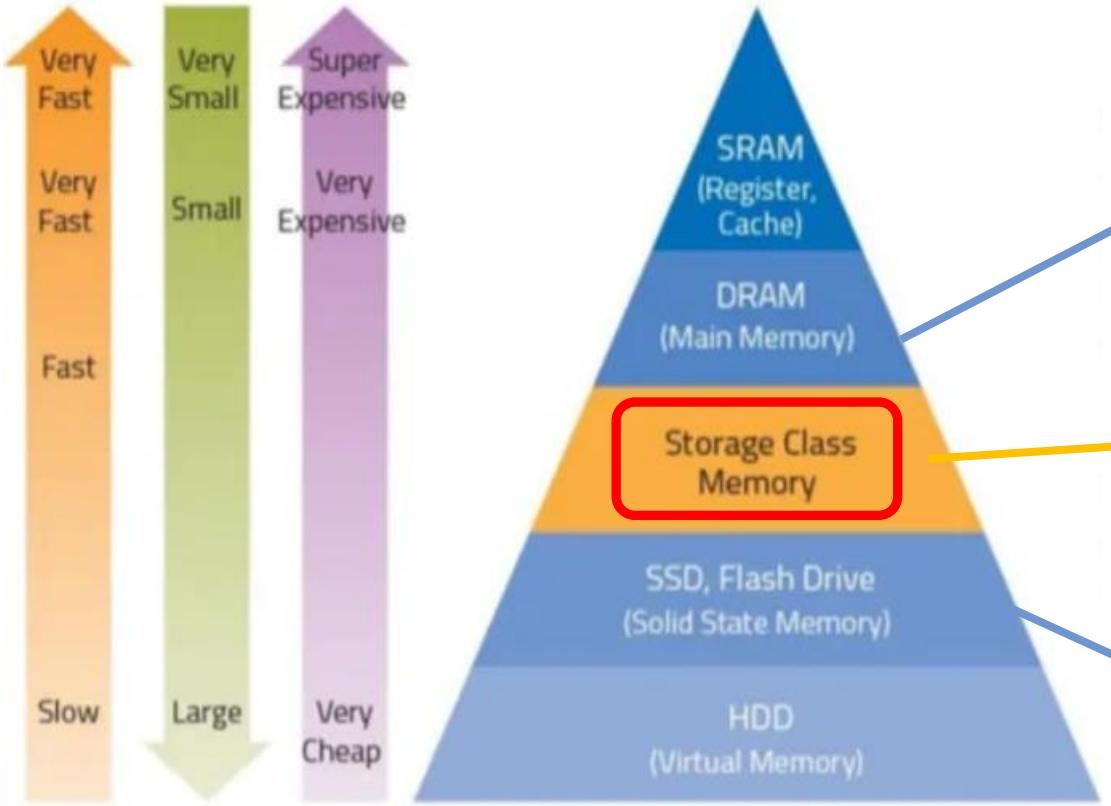
Presentation by Han, Yejin

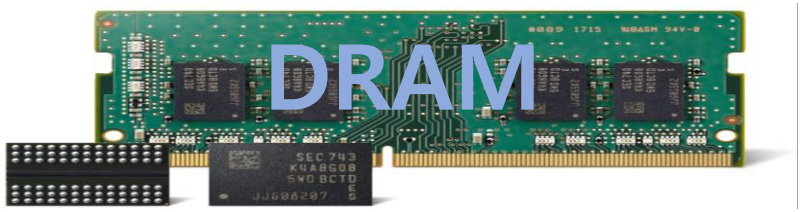
32164881@dankook.ac.kr

Contents

Hardware Trends

- Non-Volatile Memory
- Performance, Capacity, Cost





NVM characteristics

- ✓ Non-volatile
- ✓ R/W Performance
- ✓ Byte-addressable

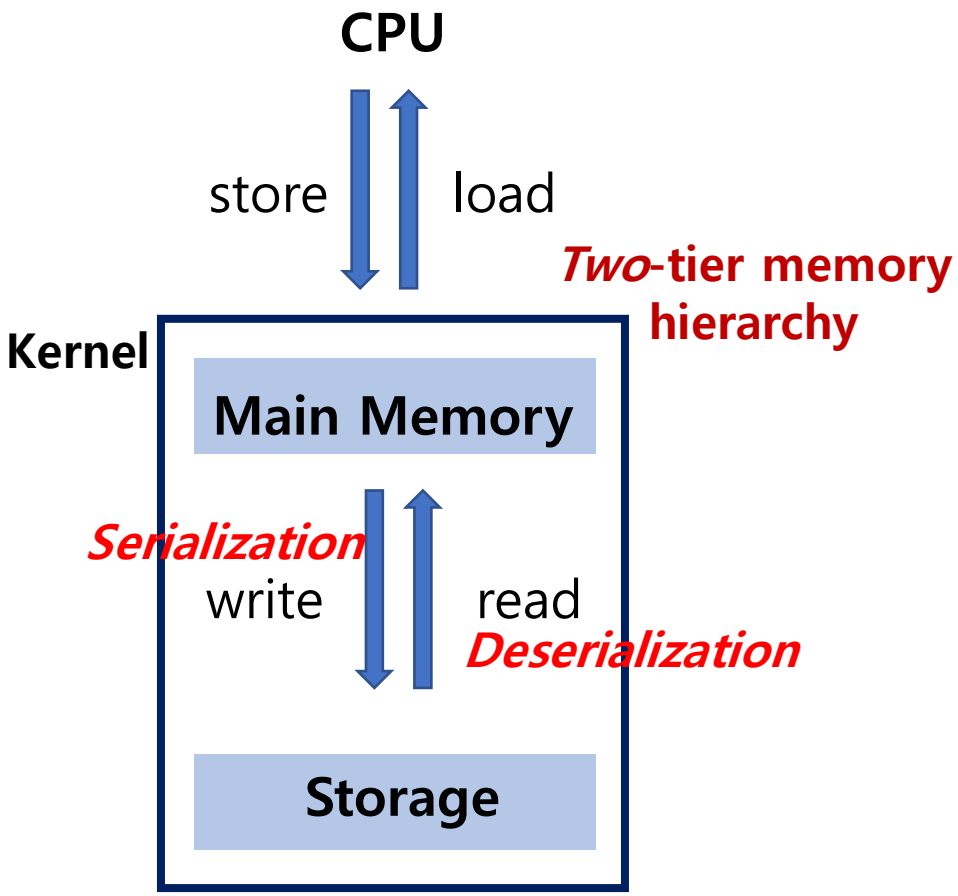
COMPARISON OF HIGH-DENSITY MEMORY TECHNOLOGIES

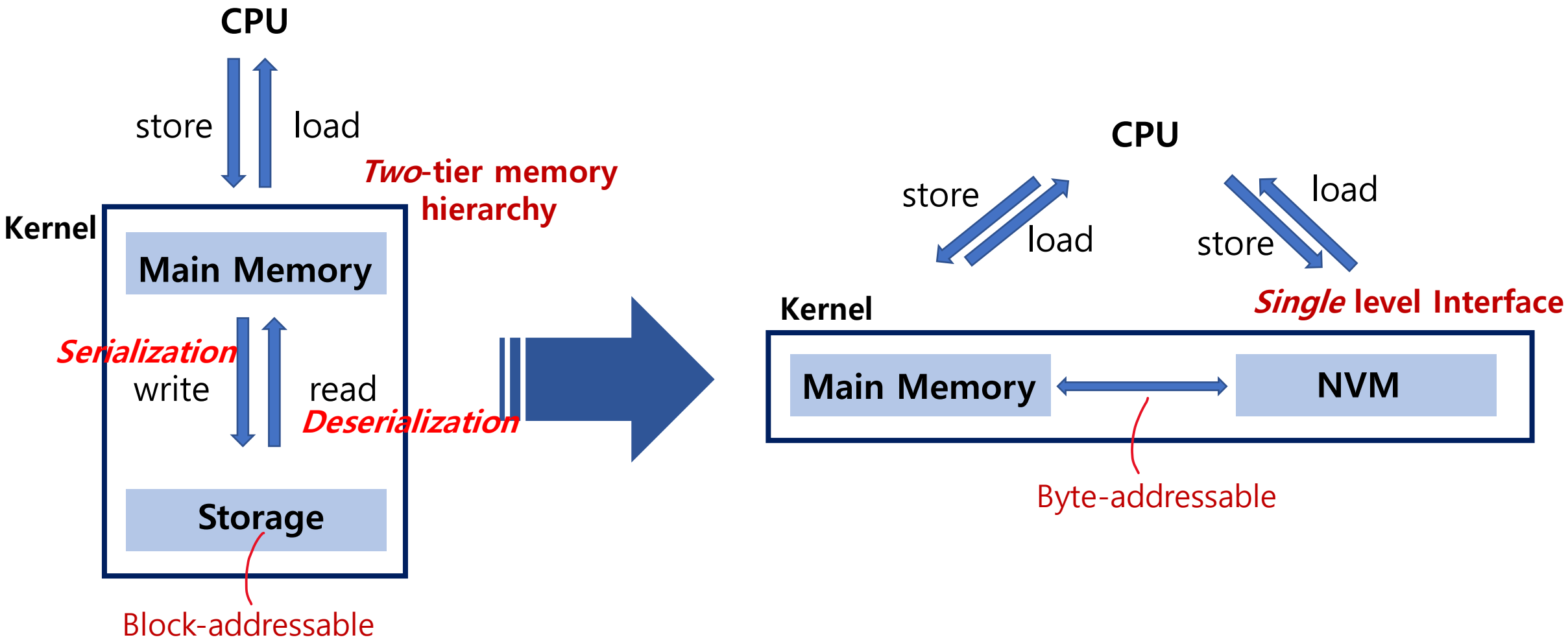
Attributes	DRAM	PCM	NAND
Non-Volatile	No	Yes	Yes
Erase Required	Bit	Bit	Block
Software	Simple	Simple	Complex
Power	~W/GB	100→500mW/die	~100mW/die
Write Bandwidth	~GB/s	1→100+ MB/s/die	10→100 MB/s/die
Write Latency	~20-50ns	~1µs	~100µs
Write Energy	~0.1nJ/b	<1nJ/b	0.1-1nJ/b
Read Latency	50ns	50 - 100 ns	10-25 µs
Read Energy	~0.1nJ/b	<<1nJ/b	<<1nJ/b
Byte-addressable	Yes	Yes	No

1.5-8x

NVM->Storage

출처: https://research.cs.wisc.edu/sonar/wiki/pmwiki.php?n=Mnemosyne.PCMTechnology


CPU
store
load
Two-tier memory hierarchy
Kernel
Main Memory
Serialization
write
read
Deserialization
Storage



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

Requirements

- OS must evolve to support future trends in memory hierarchy organization.
- Twizzler is an OS designed around NVM for direct access to memory.
- Low overhead by removing the kernel from persistent data access path

Library OS

- ✓ Design
  - Object Management
  - Address space Management
  - Persistent Pointers
- ✓ Open-Source

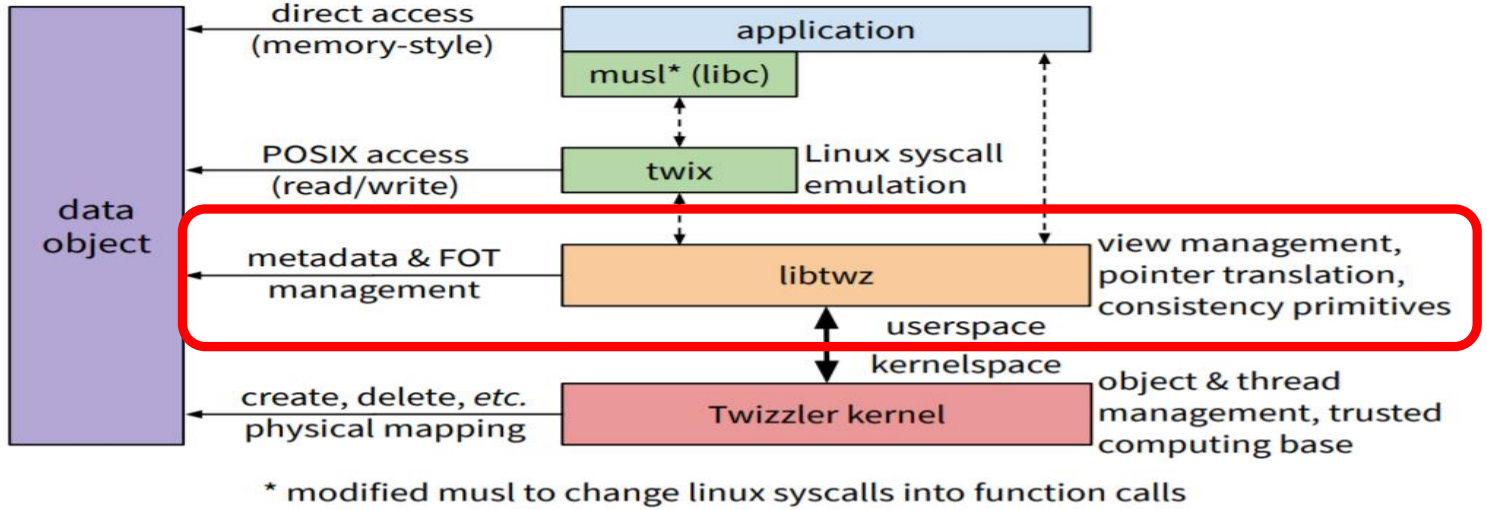
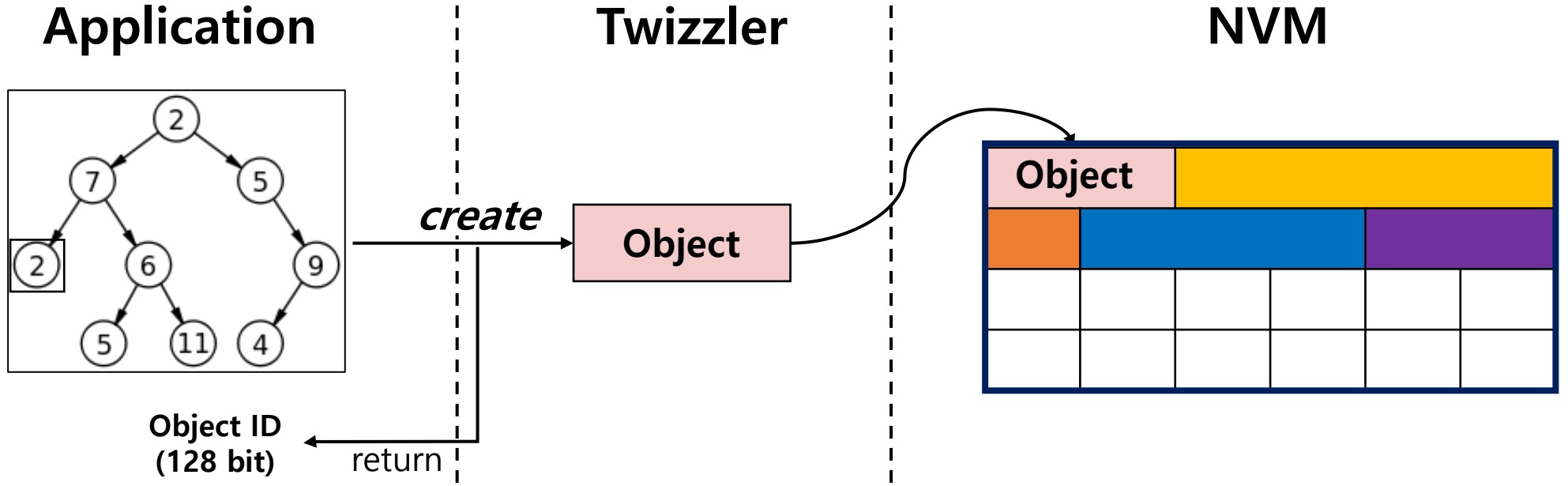


Figure 1: Twizzler system overview. Applications link to `musl` (a C library), `twix` (our Linux syscall emulation library), and `libtwz` (our standard library).

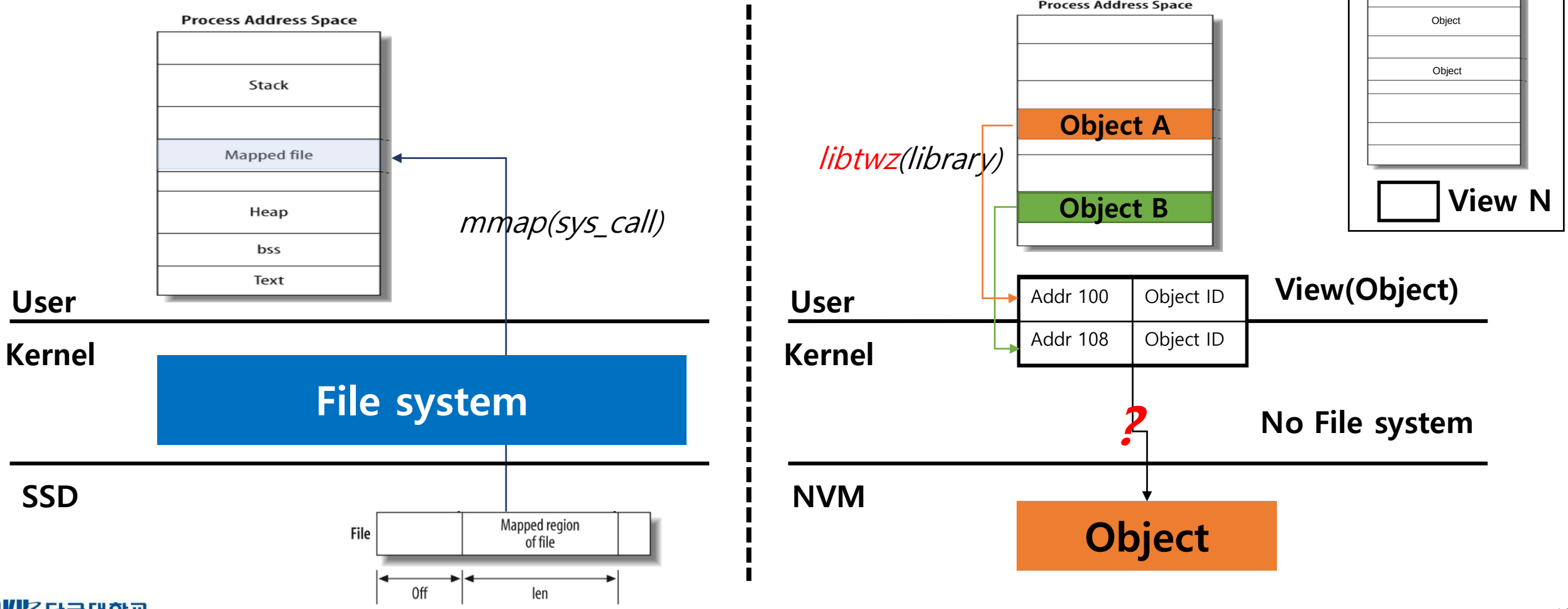
Object Management

- ✓ Twizzler organizes data into *objects*
- ✓ Twizzler uses objects as the unit of access control
  - Read/Write/Execute Permissions (MMU)
- ✓ An object is flexible in its contents (ex. A single node, the entire tree)
- ✓ Objects are created/deleted by the *create/delete* system call



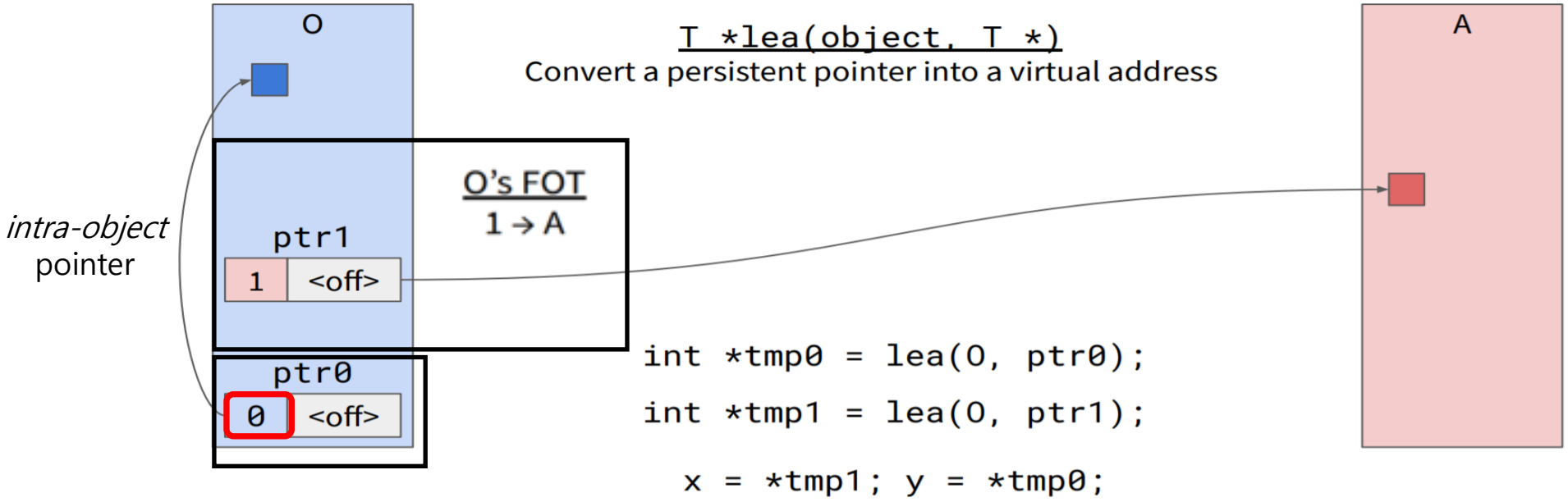
Address Space Management

- ✓ Twizzler maps persistent objects into the VA space via *libtwz*
- ✓ Kernel and user-space share a view that defines an AS layout



Persistent Pointers

- ✓ Twizzler provides cross-object persistent pointers
  - A cross-object pointer is stored as a 64 bit `FOT_idx:offset` value
- ✓ Twizzler use indirection through a per-object *foreign object table* (FOT)
  - The FOT is an array of entries that each stores an object ID and flags



Persistent Pointers

- ✓ Twizzler provides cross-object persistent pointers
  - A cross-object pointer is stored as a 64 bit `FOT_idx:offset` value
- ✓ Twizzler use indirection through a per-object *foreign object table* (FOT)
  - The FOT is an array of entries that each stores an object ID and flags

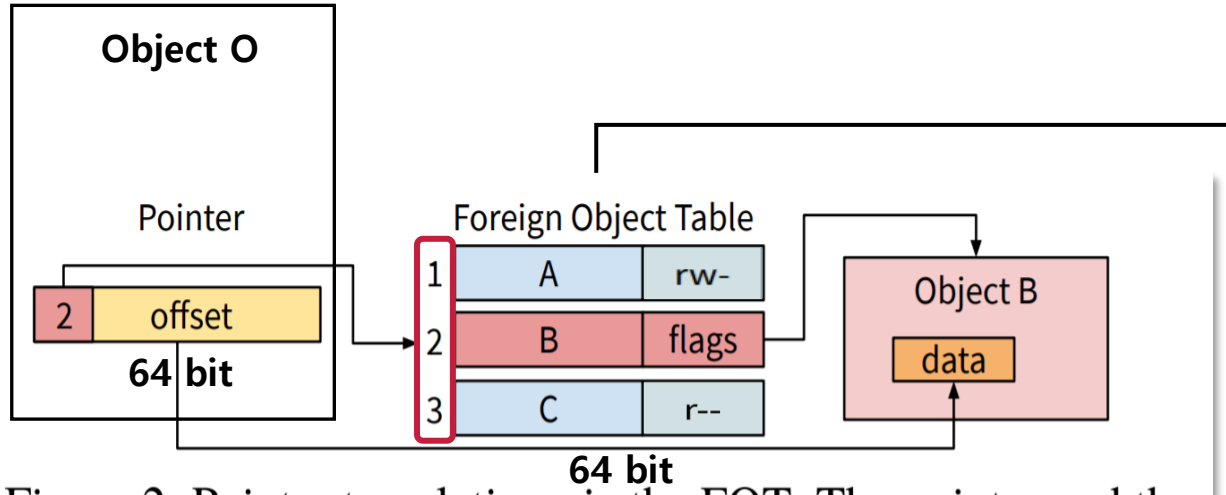
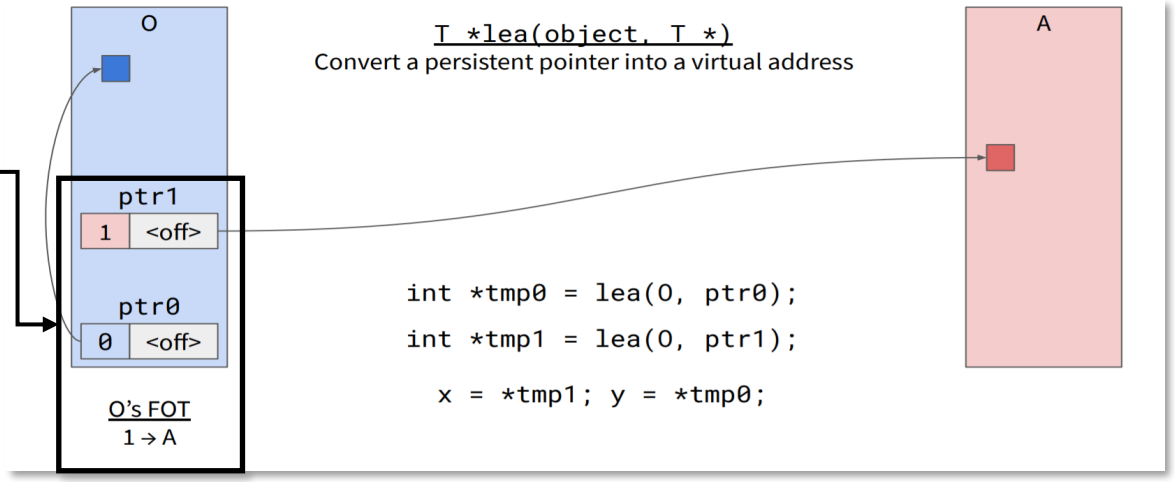


Figure 2: Pointer translation via the FOT. The pointer and the FOT are both contained in the same object (not shown).



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

128bit Direct pointer→ 64bit pointer → Cache Hit


숟가락업기
히트다히트

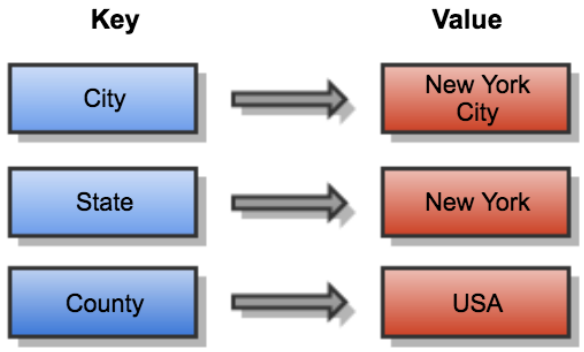
Micro benchmarks

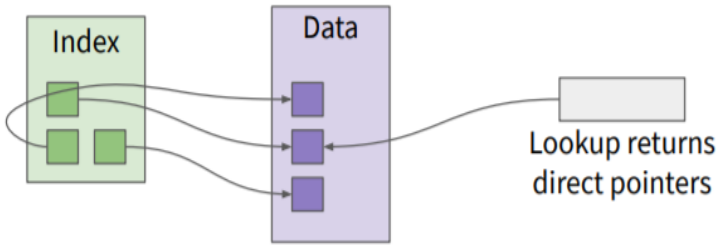
Table 1: Latency of common Twizzler operations.

Pointer Resolution Action	Average Latency (ns)
Uncached FOT translation	27.9 ± 0.1
Cached FOT translation	3.2 ± 0.1
Intra-object translation	0.4 ± 0.1
Mapping object overhead	49.4 ± 0.2

Case Study

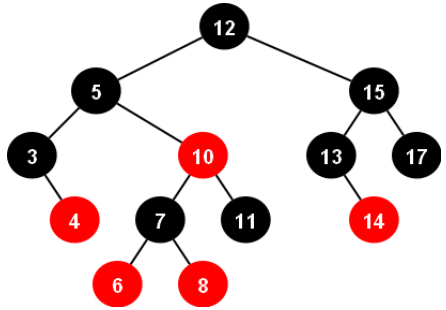
1) Key-Value Store





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

2) Red-Black Tree



ramrbt

DRAM

NVM

unixrbt vs. *twzrbt*

Simpler

Data sharing

Performance

1) Key-Value Store

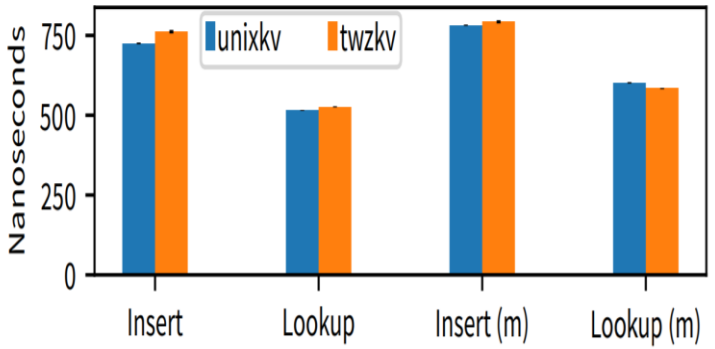


Figure 6: Latency of insert and lookup in `twzkv` and `unixkv`. An "(m)" indicates support for multiple data objects.

2) Red-Black Tree

We measured the latency of insert and lookup of 1 million 32-bit integers on both `unixrbt` and `twzrbt`. The insert and lookup latency of `twzrbt` was 528 ± 3 ns and 251.8 ± 0.5 ns, while insert and lookup latency of `unixrbt` was 515 ± 2 ns and 213 ± 1 ns. The modest overhead comes with significantly improved flexibility, as `unixrbt` does not support cross-object trees, and less support code (`unixrbt` manually implements mapping and pointer translations). Note that even though there is lookup overhead in `twzrbt`, this overhead did not predict the results of a larger program—the SQL-Twizzler port used this red-black tree, and saw performance benefits over block-based implementations.

3)SQLite

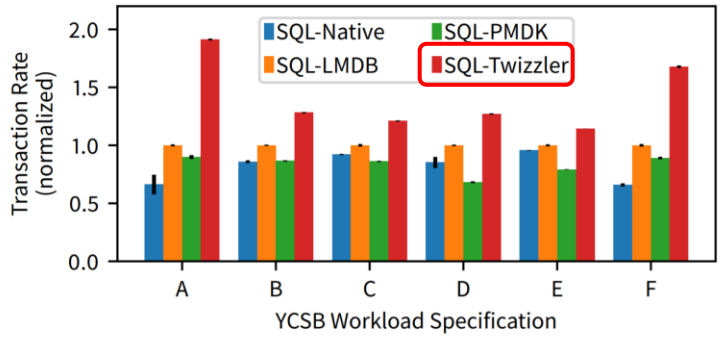


Figure 4: YCSB throughput, normalized (higher is better).

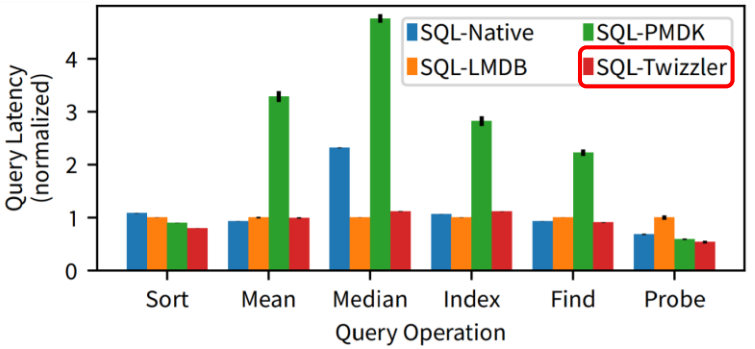


Figure 5: Query latency, normalized (lower is better).

- **Future trends in memory hierarchy**
  - Byte-addressable, non-volatile memory NVM
- **Low latency and the direct-access nature of NVM**
  - Remove the kernel from I/O path, need not to serialize
  - Object/Address space management, persistent pointers
- **Evaluation**
  - Simple implementation, high performance

Q&A

2020. 08. 10

Presentation by Han, Yejin

32164881@dankook.ac.kr