

uCache

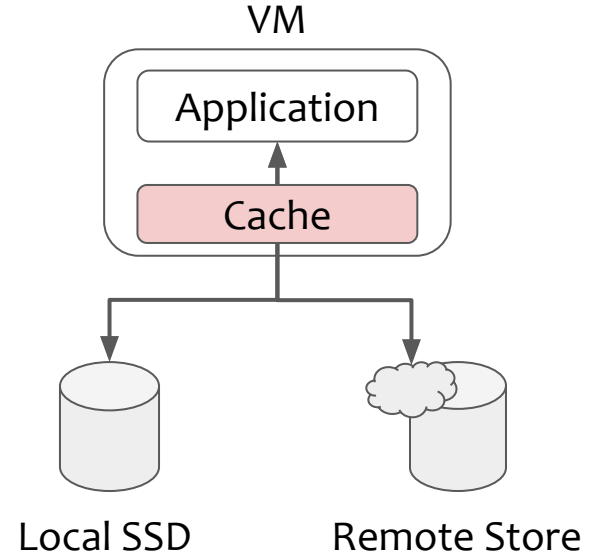
A Customizable Unikernel-based IO Cache

Ilya Meignan--Masson, Masanori Misono, Viktor Leis,
Pramod Bhatotia



IO caching in the cloud

- Data-intensive cloud applications
 - Database systems
 - ML systems
- High-performance IO devices
 - NVMe SSDs, NICs
 - PCI 5.0: up to 10s of GiB/s

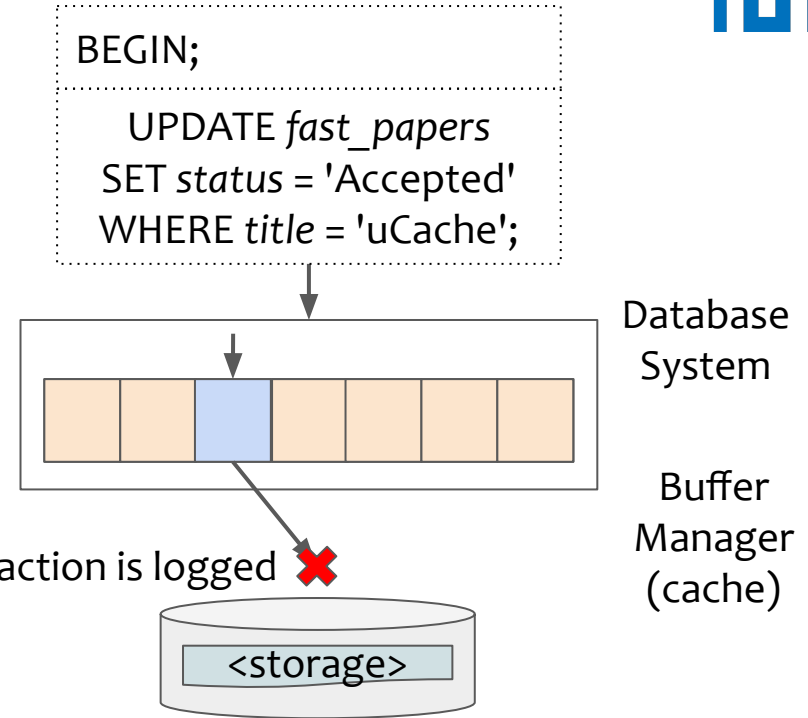


What are the applications requirements for the cache?

What do applications want?

1. Non-hardware supported page sizes
 2. Custom cache replacement policies
- > Cheaply prevent the eviction of a page
3. Access local drives and remote stores

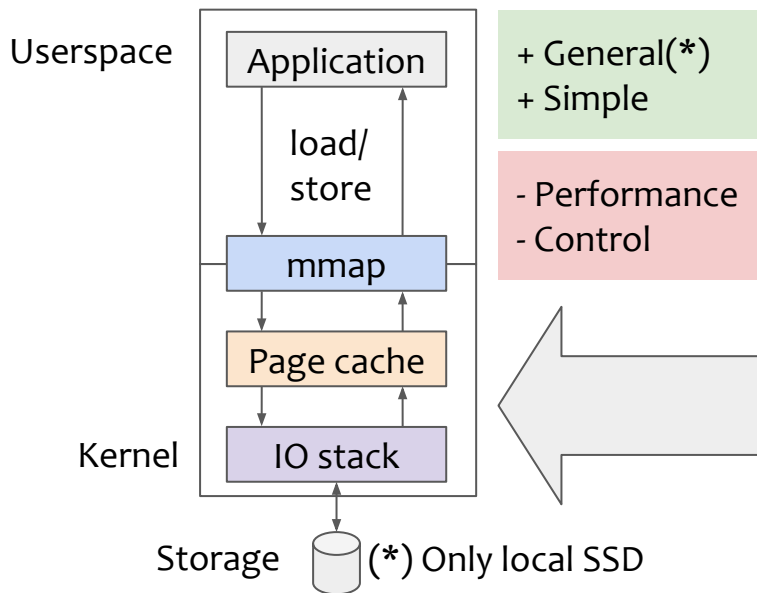
Not until transaction is logged ❌



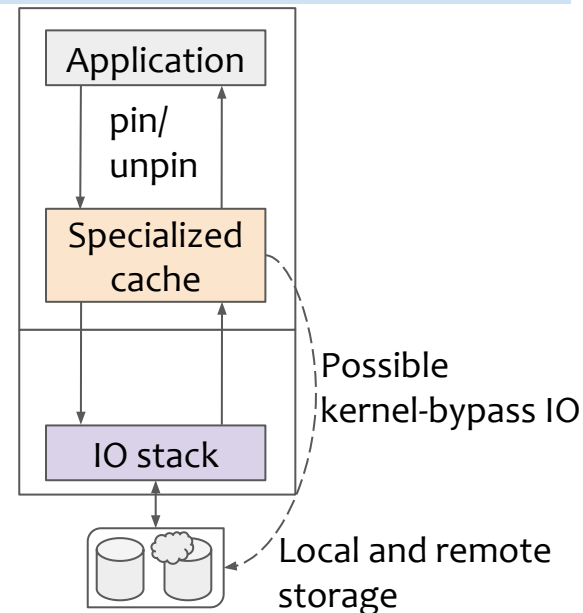
What are the options for caching?

Cache design approaches

Traditional: OS-level caching



State-of-the-art: Explicit userspace caching



Can we overcome the limitations of each approach?

Can the OS provide a simple, flexible and performant IO cache?

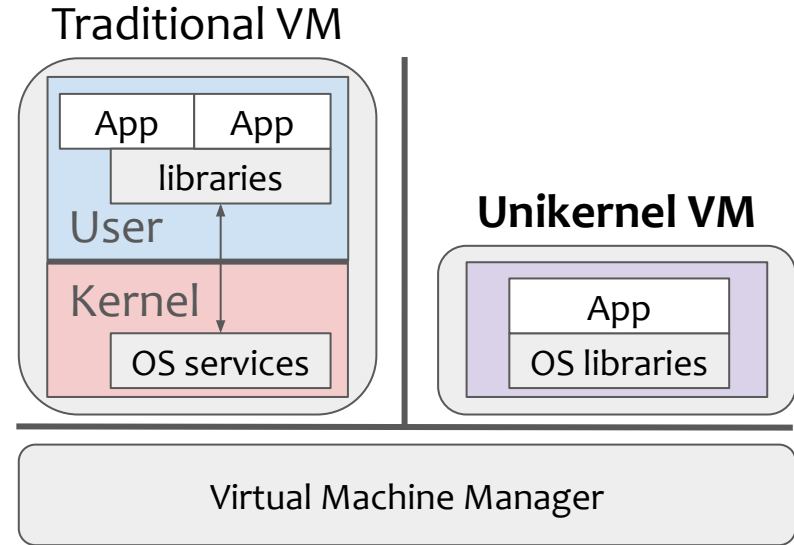
A customizable memory-mapped IO caching framework

Design Goals:

- Flexibility, adapt to the multiple workloads and storage solutions
- Simplicity, the application decide the level of control
- Performance, enable applications to leverage high-performance IO devices

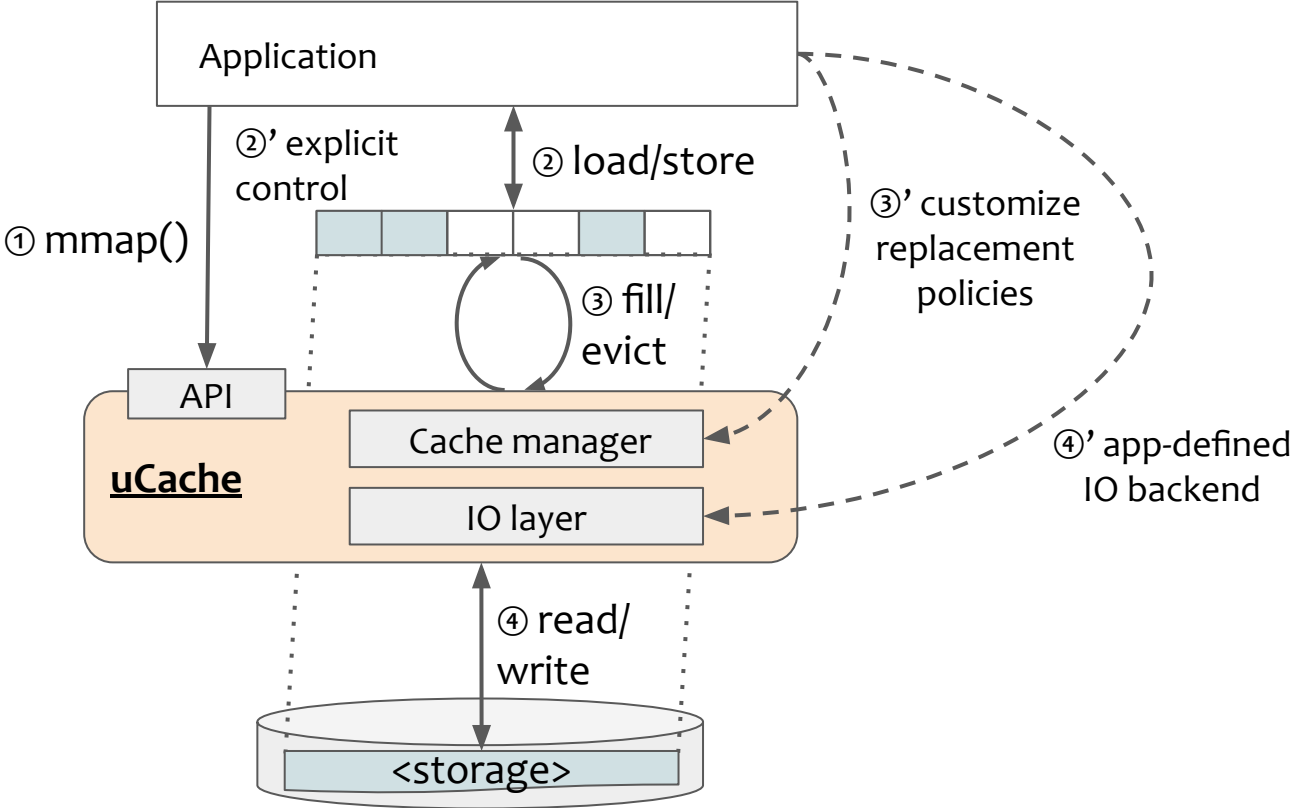
Key idea: A unikernel architecture

1. **Single** address-space
2. **Library** OS
3. Built into a **VM image**



The unikernel architecture enables OS/App co-design

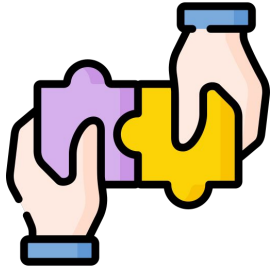
Overview



Outline



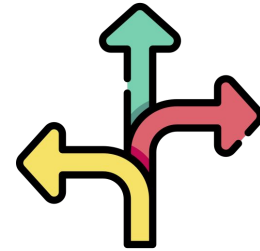
- Overview
- Design
- Evaluation



Application/OS
coordination



Scalable cache
operations



Flexible IO
abstraction

Challenge #1: Application/OS coordination

**Completely
transparent**

*e.g., mmap +
load/store*

**Customizable
policies**

*e.g., eviction,
prefetching, ...*

**Explicit
control**

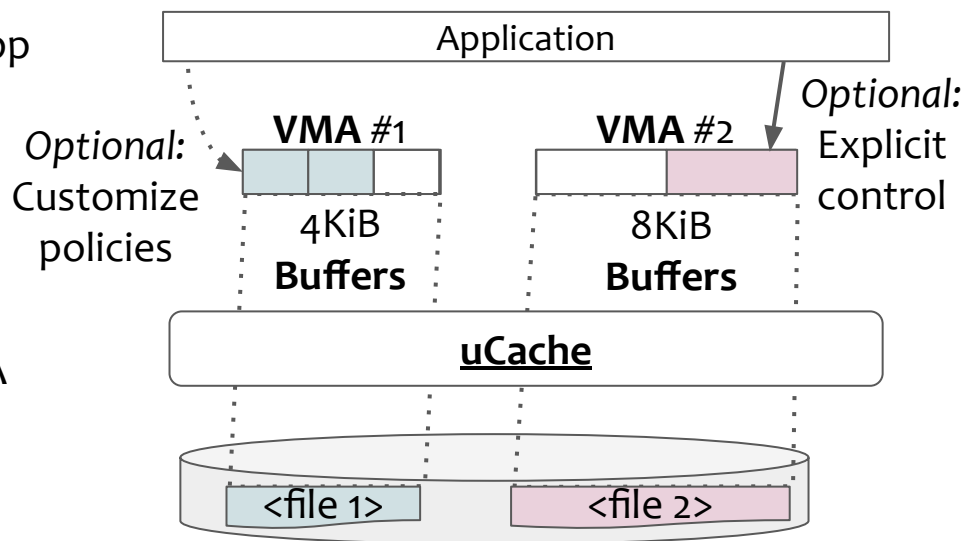
*e.g., pin/unpin, evict,
writeback, ...*



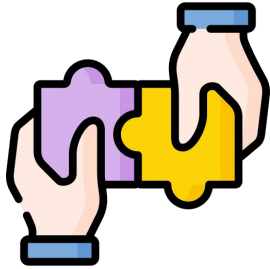
How to enable a **spectrum** of interactions models?

Solution #1: uCache memory abstractions

- Virtual memory area (**VMA**)
 - Shared between the OS and the app
- **Buffer**
 - Granularity for primitives
- **Customizable** policies:
 - Attach function callbacks to a VMA



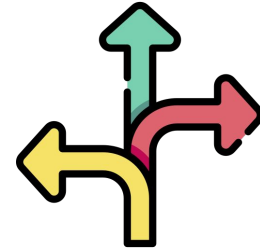
uCache enables OS/App coordination through shared VMA and Buffer abstractions



Application/OS
coordination

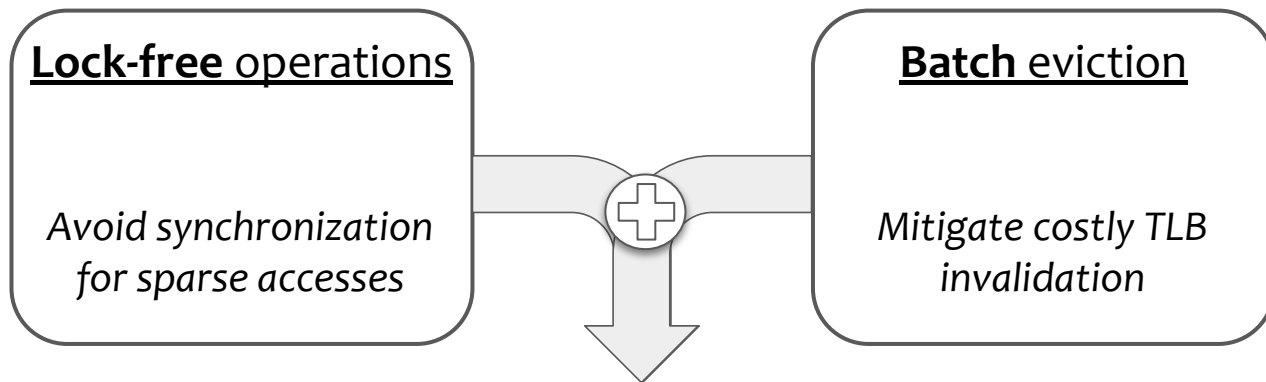


Scalable cache
operations



Flexible IO
abstraction

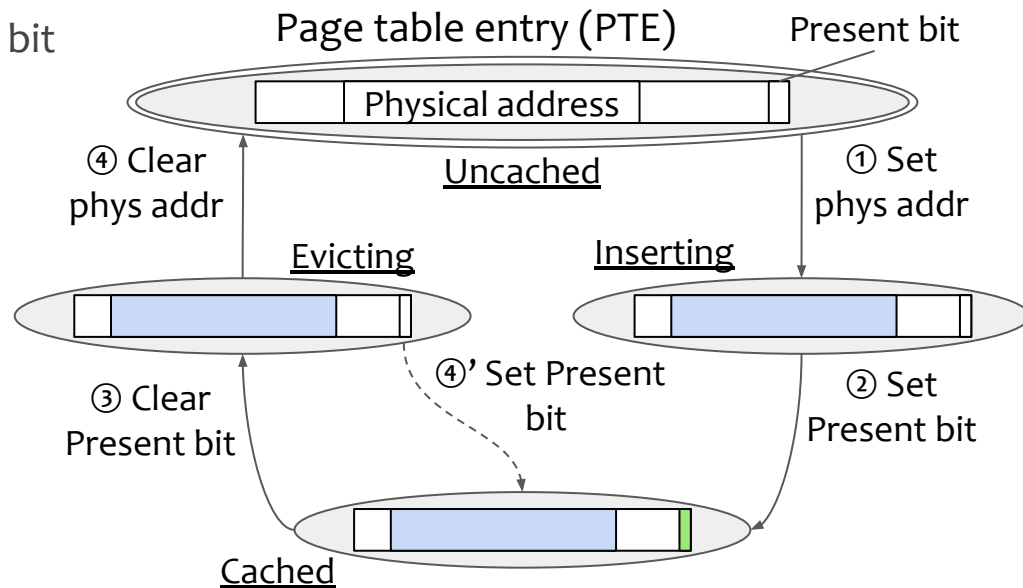
Challenge #2: Scalable cache operations



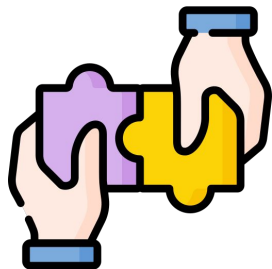
How to modify the page table in a **lock-free** manner with **batching**?

Solution #2: Optimistic cache operations

- **Two-step** CAS operations
 - Separate phys addr and present bit
- During eviction:
 - Accumulate candidates
 - Accesses **cancel** eviction
- Works also for “**big pages**”
 - More details in the paper



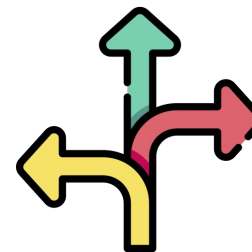
uCache uses **optimistic operations** to limit synchronization even when batching



Application/OS
coordination

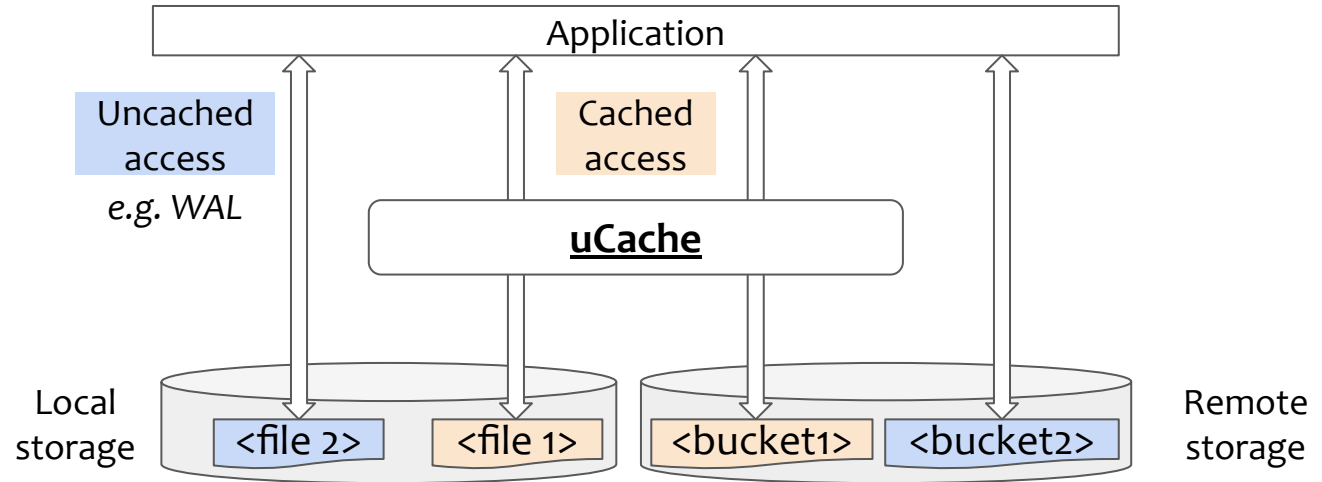


Scalable cache
operations



Flexible IO
abstraction

Challenge #3: Flexible IO abstraction

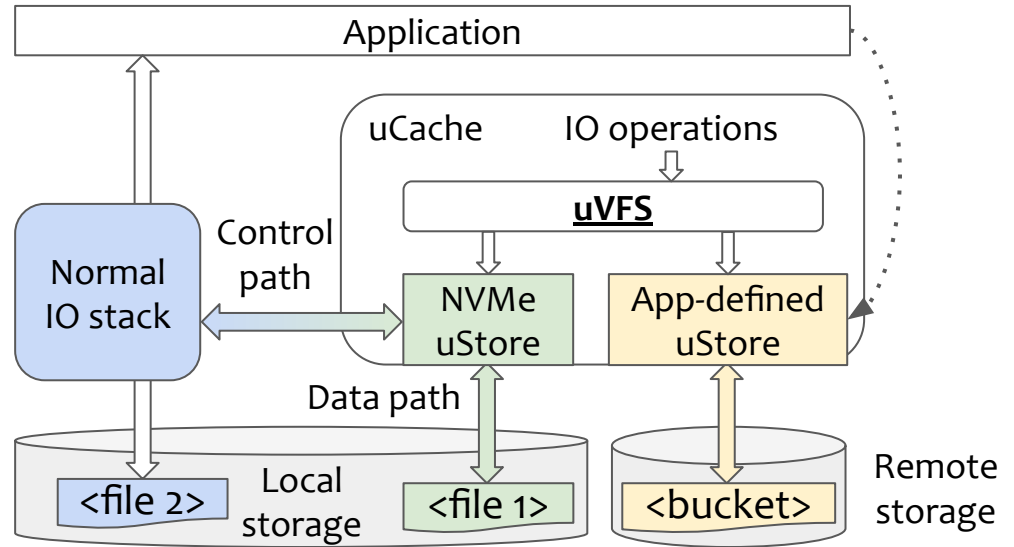


How to enable flexible storage access to applications?

Solution #3: uCache IO abstractions

- uVFS: Abstracts IO stack
 - dispatches to uStores
 - App-defined uStores

- NVMe uStore
 - performant data path
 - compatible control path



uCache enables performant and customizable IO without hindering compatibility

Implementation

- Built on the **OSv** unikernel
 - Cache manager
 - Default: CLOCK replacement
 - Physical memory allocation: LLFree[ATC'23]
- IO layer
 - ext4-compatible NVMe uStore



Outline



- Overview
- Design
- Evaluation

Evaluation

1. How performant are cache management operations?
- ~~2. How much footprint does uCache introduce?~~
- ~~3. What is the performance of the IO layer?~~
4. Use cases
 - a. Is uCache a compelling replacement to mmap?
 - b. Can uCache be used for the buffer manager of database systems?
 - ~~c. Can uCache be used to improve the performance of accessing data files?~~

More evaluation in the paper

Experimental setup

Setup:

- AMD EPYC 9654 (96C), 768GiB RAM
- Kioxia CM-7 NVMe SSD (PCI 5.0)

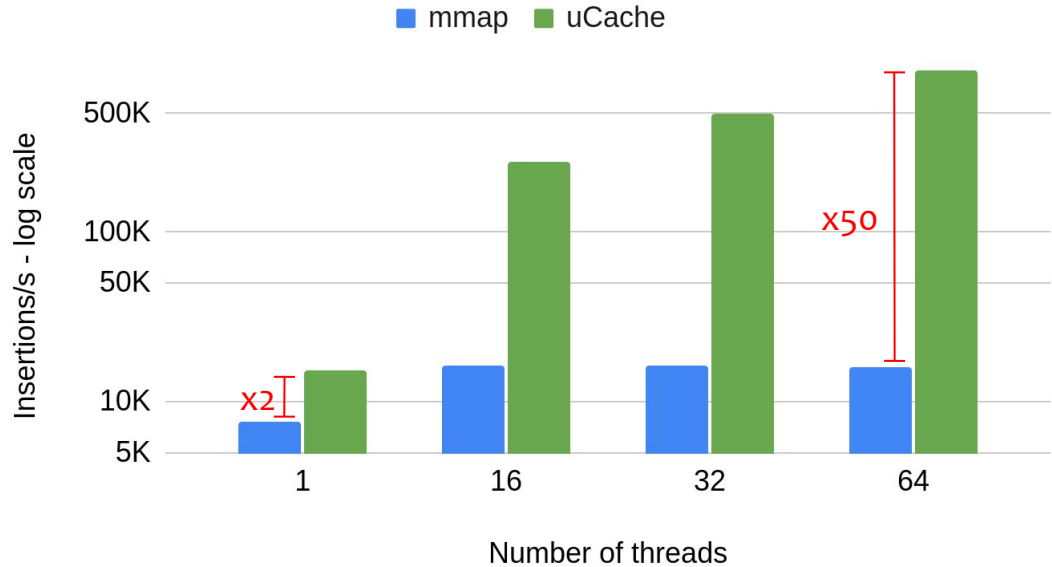
Variants:

- mmap in Linux VM
- Specialized userspace cache
- uCache

Higher is better ↑

Insertions per second

- focus on out-of-memory case
- includes evictions (no writes)



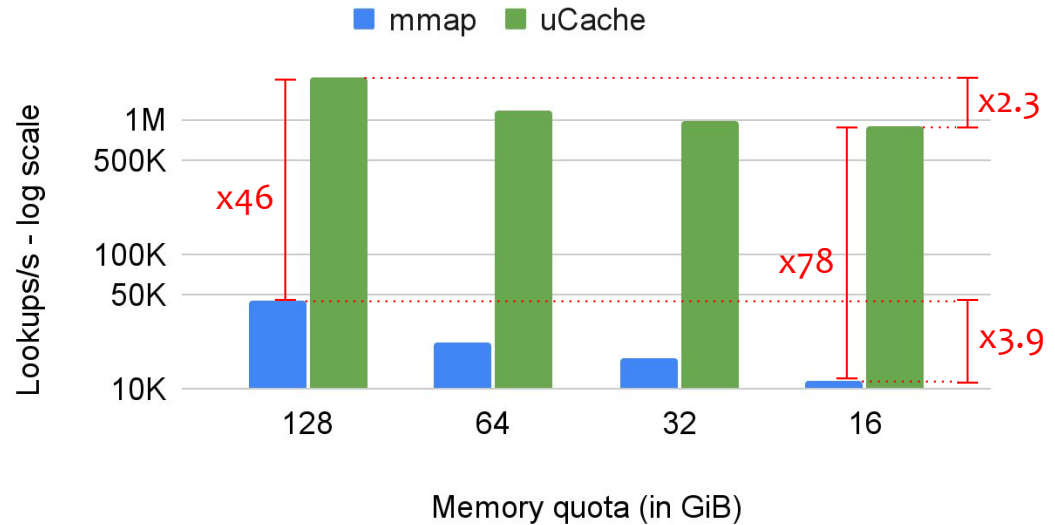
uCache scales linearly with the number of threads

End-to-end performance

Lookups per second:

- random access pattern
- 200GiB data size
- 64 threads

Higher is better ↑



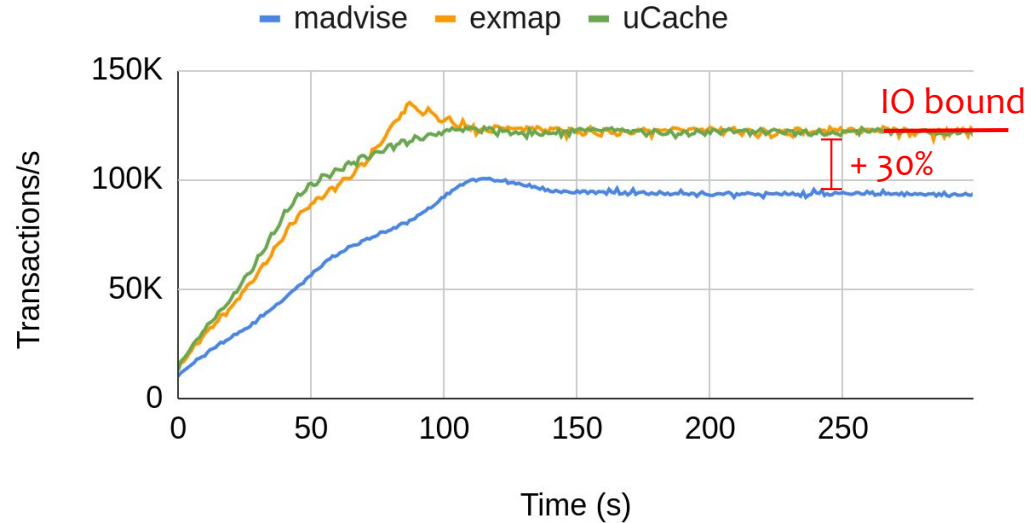
uCache is a compelling replacement to mmap performance-wise

Use case: Database systems buffer management

Buffer cache: vmcache[SIGMOD'23]

- with POSIX (madvise)
- with kernel module (exmap)
- ported to uCache

Higher is better ↑



TPC-C:

- size 5000 (~1TiB)
- cache 128GiB
- 64 threads

uCache can integrate complex application semantics without incurring overheads

Conclusion

ARTIFACT EVALUATED	ARTIFACT EVALUATED	ARTIFACT EVALUATED
 USENIX	 USENIX	 USENIX
AVAILABLE	FUNCTIONAL	REPRODUCED



Can the OS provide simple, flexible and performant memory-mapped IO caching?

uCache

- Aims to be convenient like OS-level caches while performant and flexible
- Customizable and scalable caching framework through the unikernel architecture
- Outperforms mmap and customizable like specialized caches

Try it out !



[TUM-DSE/uCache](https://github.com/TUM-DSE/uCache)