ScalaCache:
Scalable User-Space Page Cache Management with Software-Hardware Coordination

Peng, L., An, Y., Zhou, Y., Wang, C., Li, Q., Cheng, C., & Zhang, J.

In 2024 USENIX Annual Technical Conference (USENIX ATC 24)

2024.10.02

Presentation by Choi, Gunhee

choi_gunhee@dankook.ac.kr

단국대학교 DANKOOK UNIVERSITY

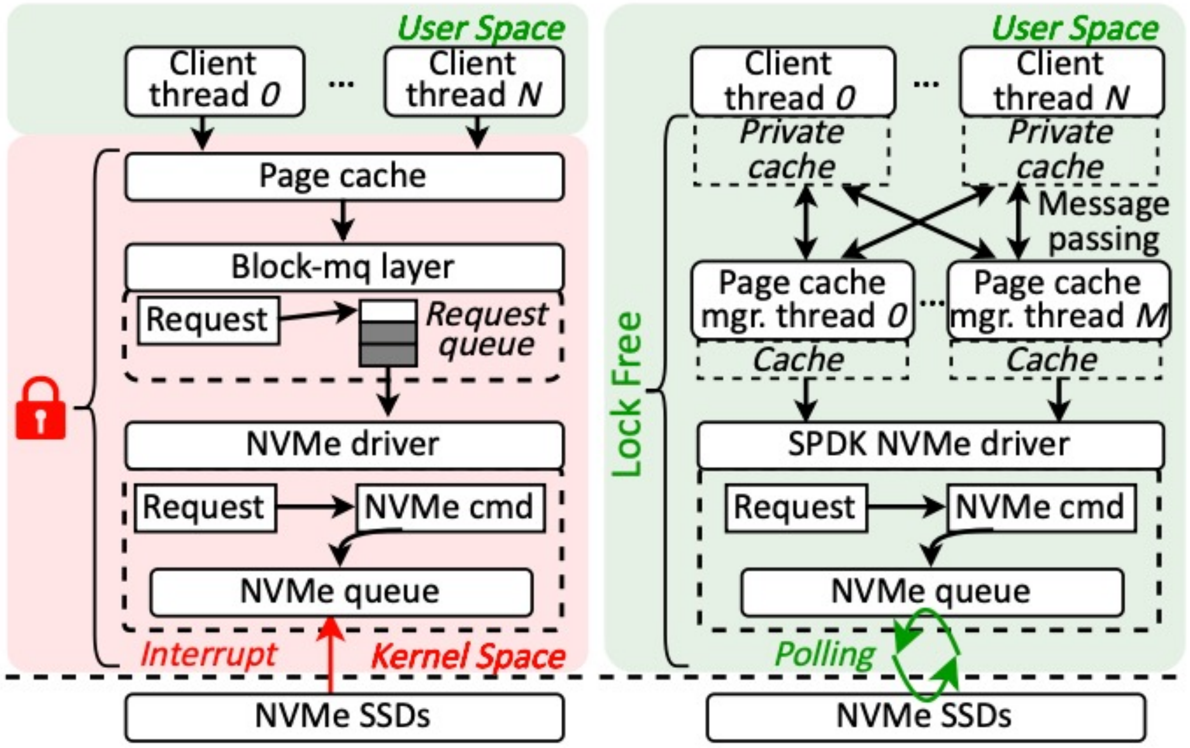
Dankook University
System Software Laboratory

Contents

The importance of cache

You know it well

2. Problem



(a) Kernel storage software stack. (b) User-space storage software stack.

Figure 3: Typical kernel and user-space software stacks.

Kernel software stack

- Kernel space implementation
- Global locking
- Interrupt

- Fails to follow up on SSD performance

TriCache

- Efficient user-space SPDK I/O engine
- Multiple threads manage cache without lock
- Message passing

- Communication Overhead
- Small, non-contiguous IO requests

2. Problem

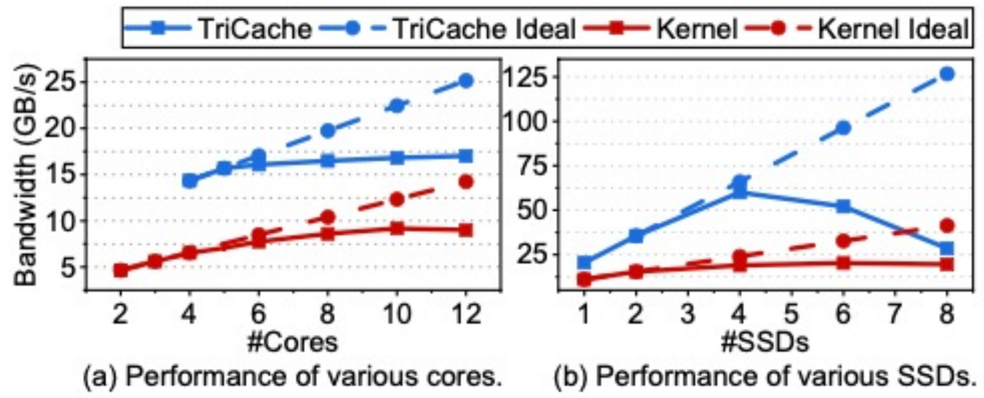


Figure 4: Performance analysis.

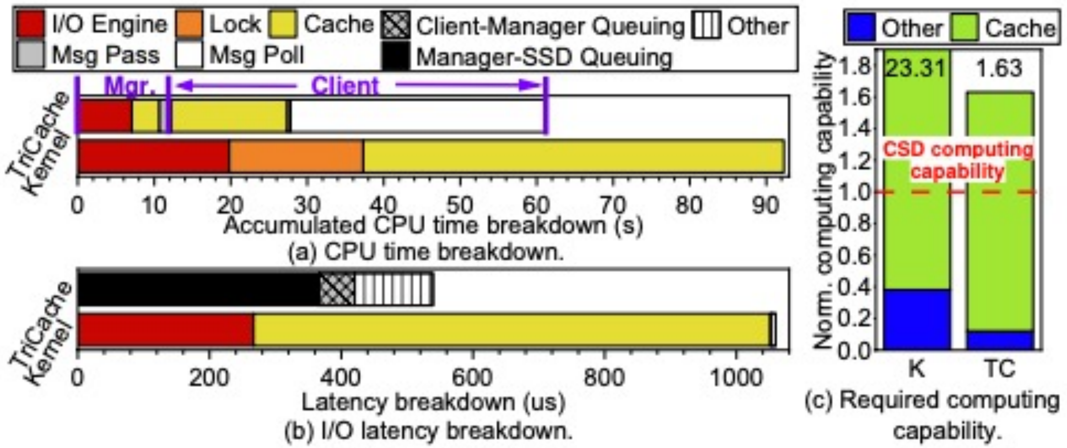


Figure 5: Breakdowns and required computing capability.

Kernel software stack

- CPU time breakdown
  - IO engine : 21.45%
  - Lock : 18.96%
- I/O latency
  - I/O engine : 25.22%
  - Cache : 74.07%

TriCache

- CPU time breakdown
  - Msg pass : 10.08%
  - Manager thread : 30.5%
  - 6.72x more NVMe cmd due to fragmentation
  - Msg poll : 68.14%
- I/O latency
  - C-M queuing : 10.02%
  - M-S queuing : 67.72%

2. Problem

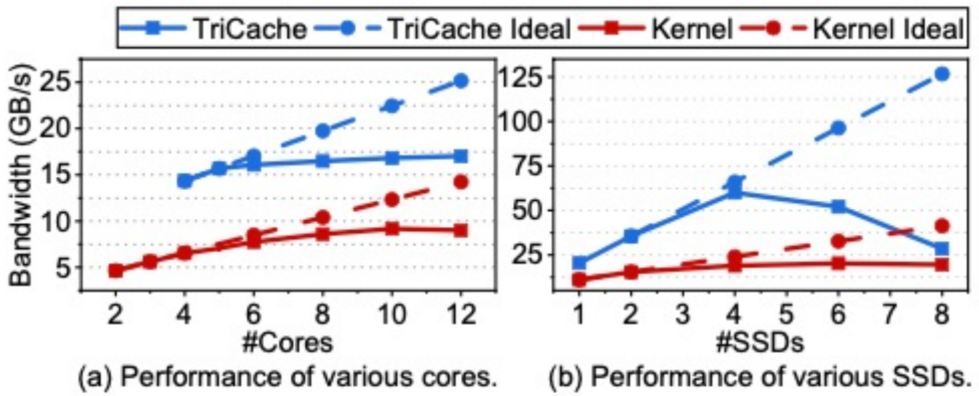


Figure 4: Performance analysis.

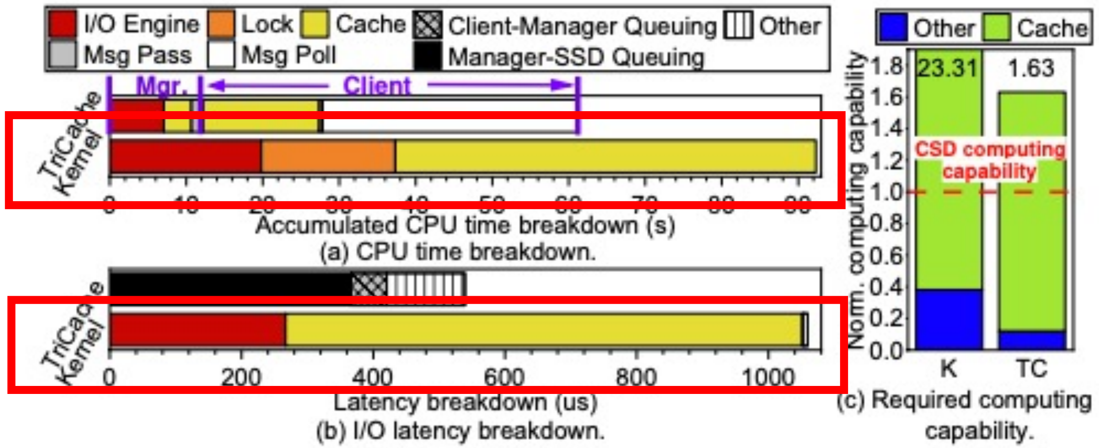


Figure 5: Breakdowns and required computing capability.

Kernel software stack

- CPU time breakdown
  - IO engine : 21.45%
  - Lock : 18.96%
- I/O latency
  - I/O engine : 25.22%
  - Cache : 74.07%

TriCache

- CPU time breakdown
  - Msg pass : 10.08%
  - Manager thread : 30.5%
  - 6.72x more NVMe cmd due to fragmentation
  - Msg poll : 68.14%
- I/O latency
  - C-M queuing : 10.02%
  - M-S queuing : 67.72%

2. Problem

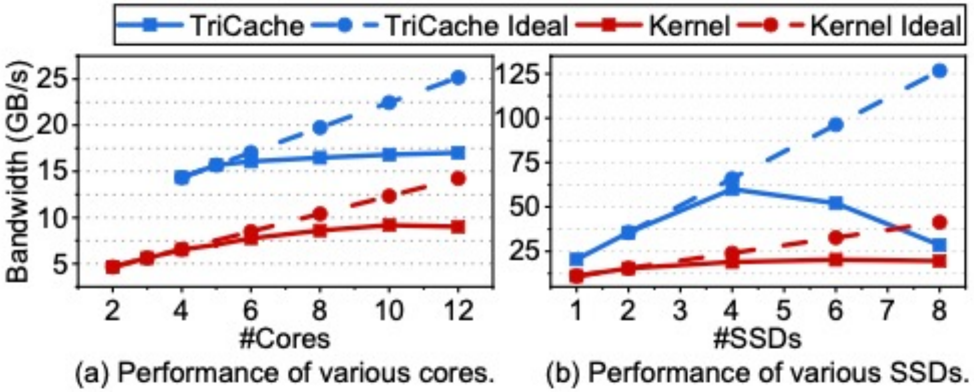


Figure 4: Performance analysis.

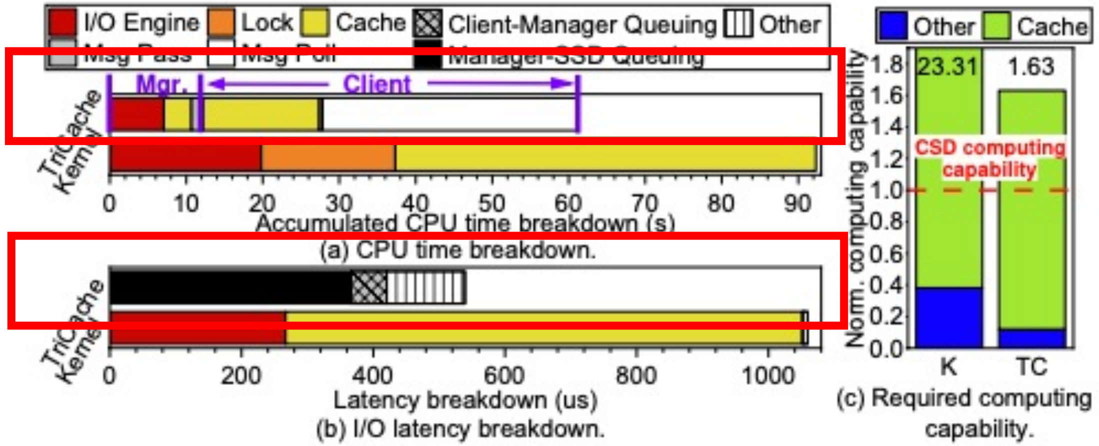


Figure 5: Breakdowns and required computing capability.

Kernel software stack

- CPU time breakdown
  - IO engine : 21.45%
  - Lock : 18.96%
- I/O latency
  - I/O engine : 25.22%
  - Cache : 74.07%

TriCache

- CPU time breakdown
  - Msg pass : 10.08%
  - Manager thread : 30.5%
  - 6.72x more NVMe cmd due to fragmentation
  - Msg poll : 68.14%
- I/O latency
  - C-M queuing : 10.02%
  - M-S queuing : 67.72%

3. Challenges

CPU consumption

- Both kernel and TriCache consume excessive computing resources for cache operations

- The CPU dependency of the host-centric design worsens as the number of SSDs increases, causing scalability problems

Communication cost

- Heavy kernel IO engine prevents efficient communication between kernel page cache and SSD

- TriCache requires frequent communication between client, cache manager thread, and page cache manager thread

GC interference

- GC activity on SSDs inadvertently blocks cache management

- Host-centric design is difficult to mitigate because it separates this layer from the SSDs

4. ScalaCache

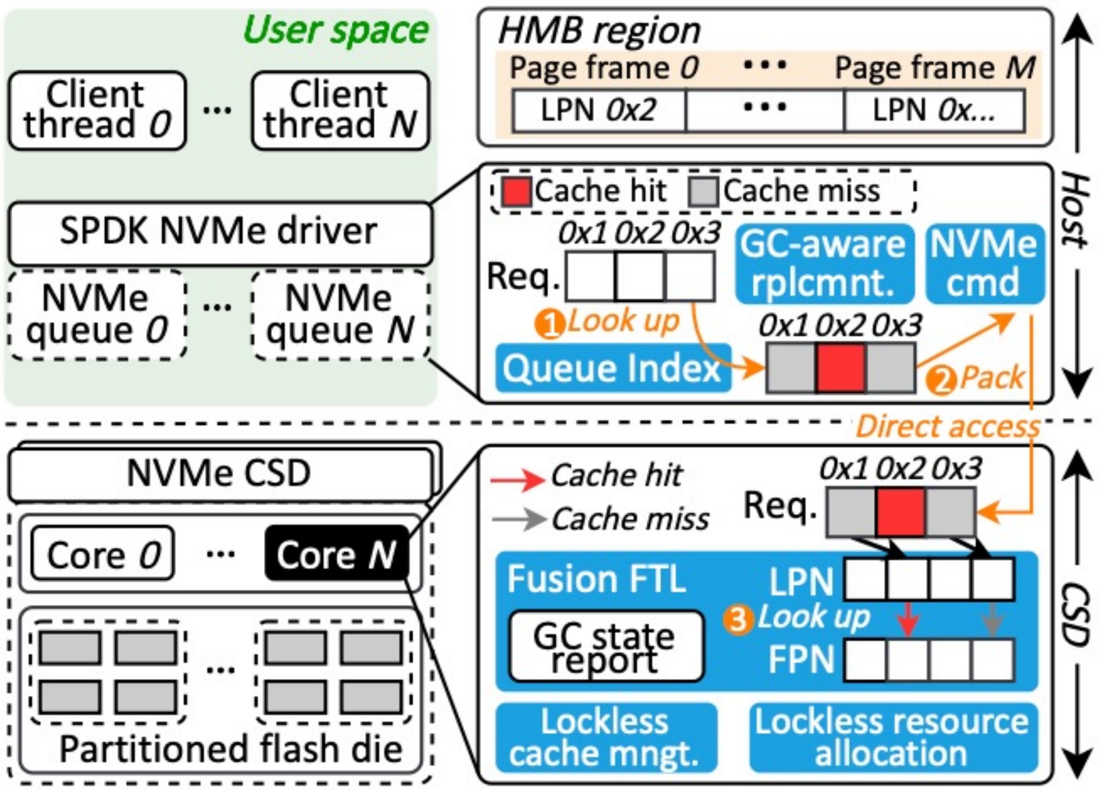


Figure 6: Overview of ScalaCache.

- **Offload the page cache manager to CSD (Computational Storage drives)**
- **FusionFTL**
- **Queue index**
- **Partitioning for concurrent access**

4. ScalaCache

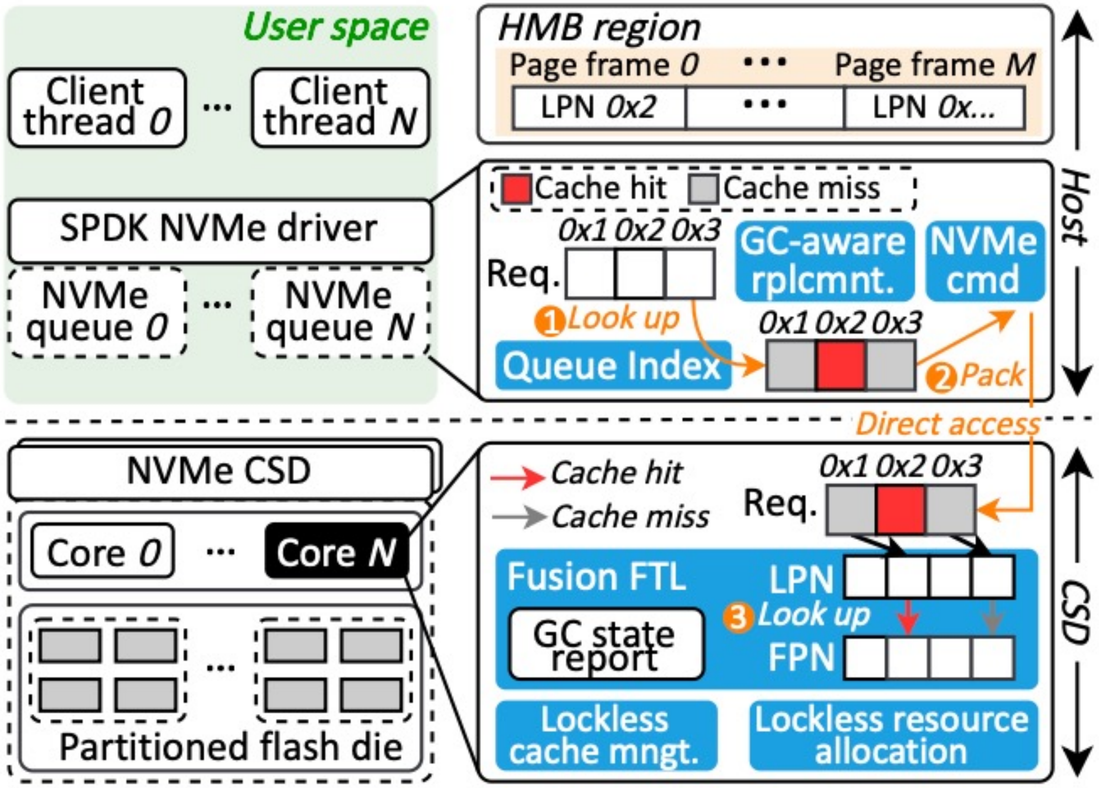


Figure 6: Overview of ScalaCache.

FusionFTL

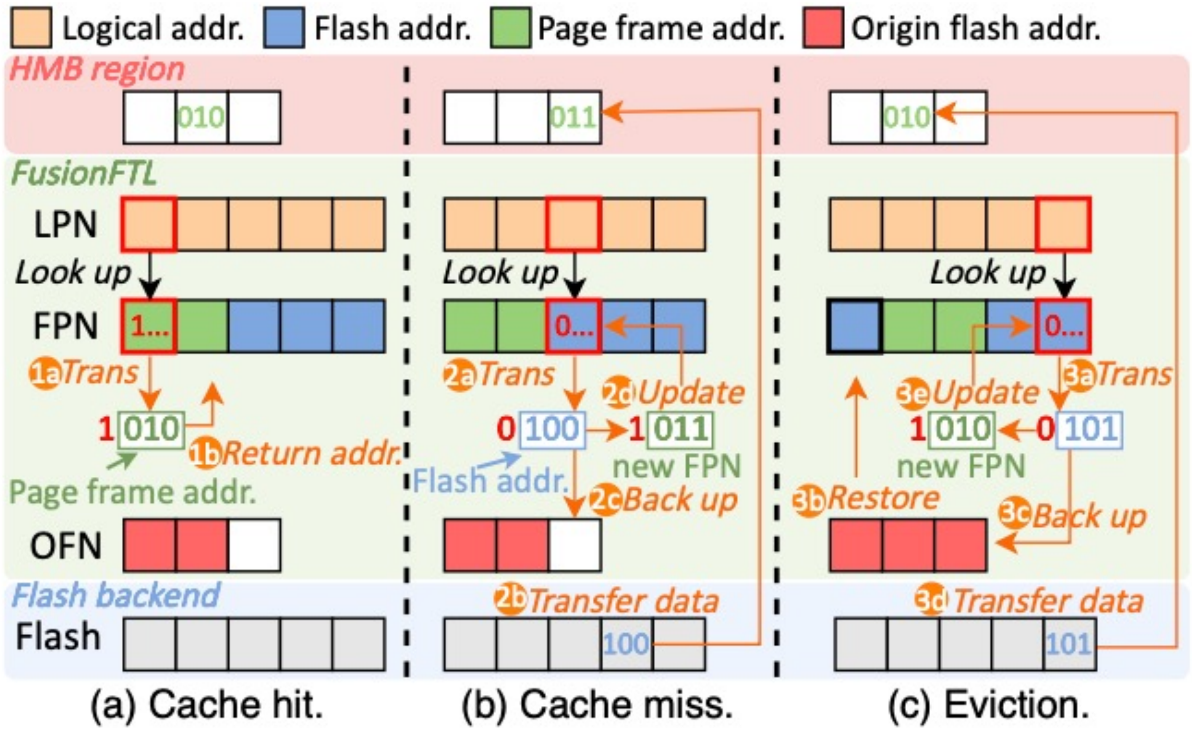


Figure 7: Details of FusionFTL.

4. ScalaCache

Queue index

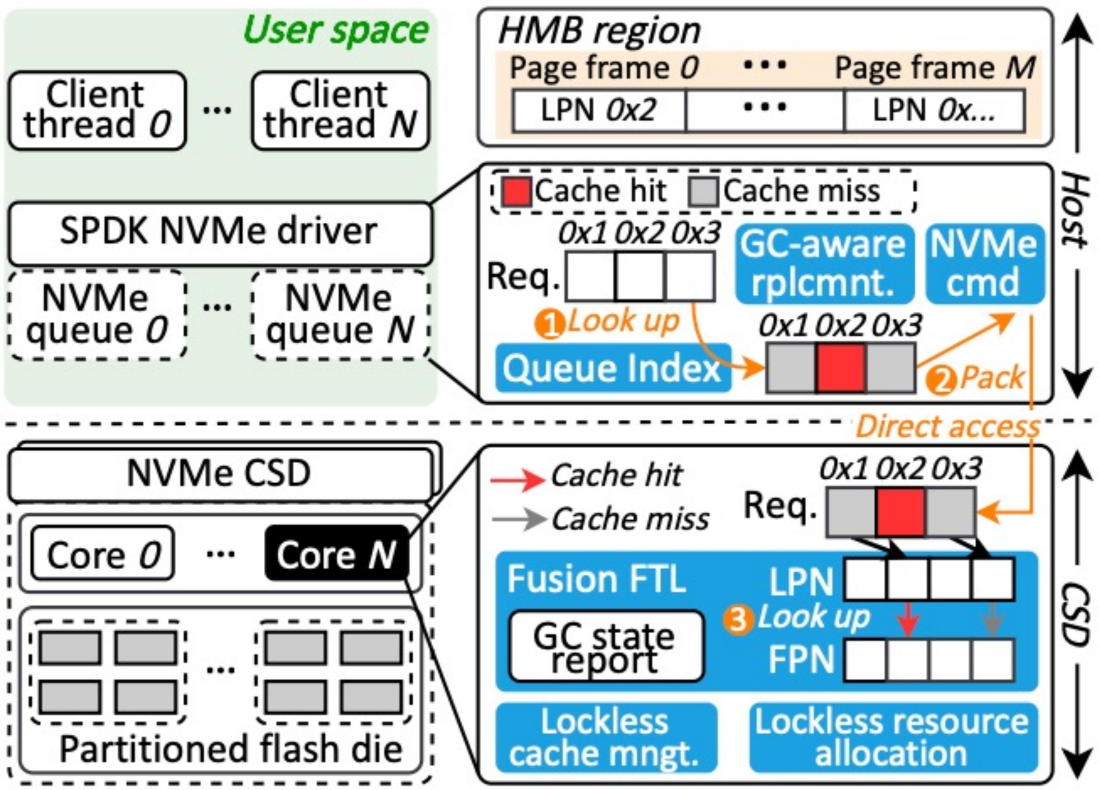


Figure 6: Overview of ScalaCache.

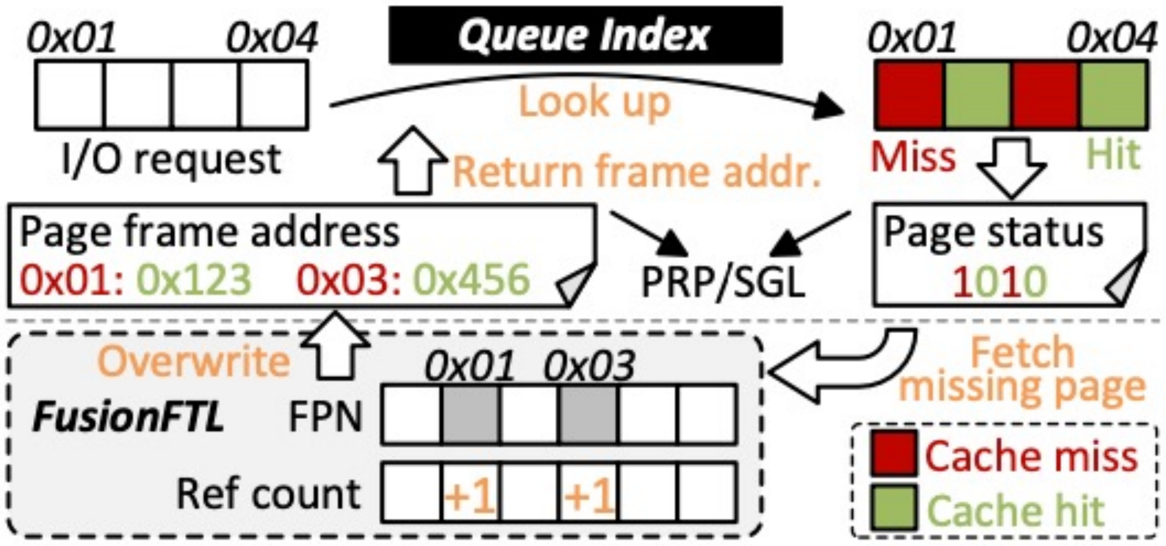


Figure 9: Coordination between Queue Index and FusionFTL.

4. ScalaCache

- **Partitioning for concurrent access**

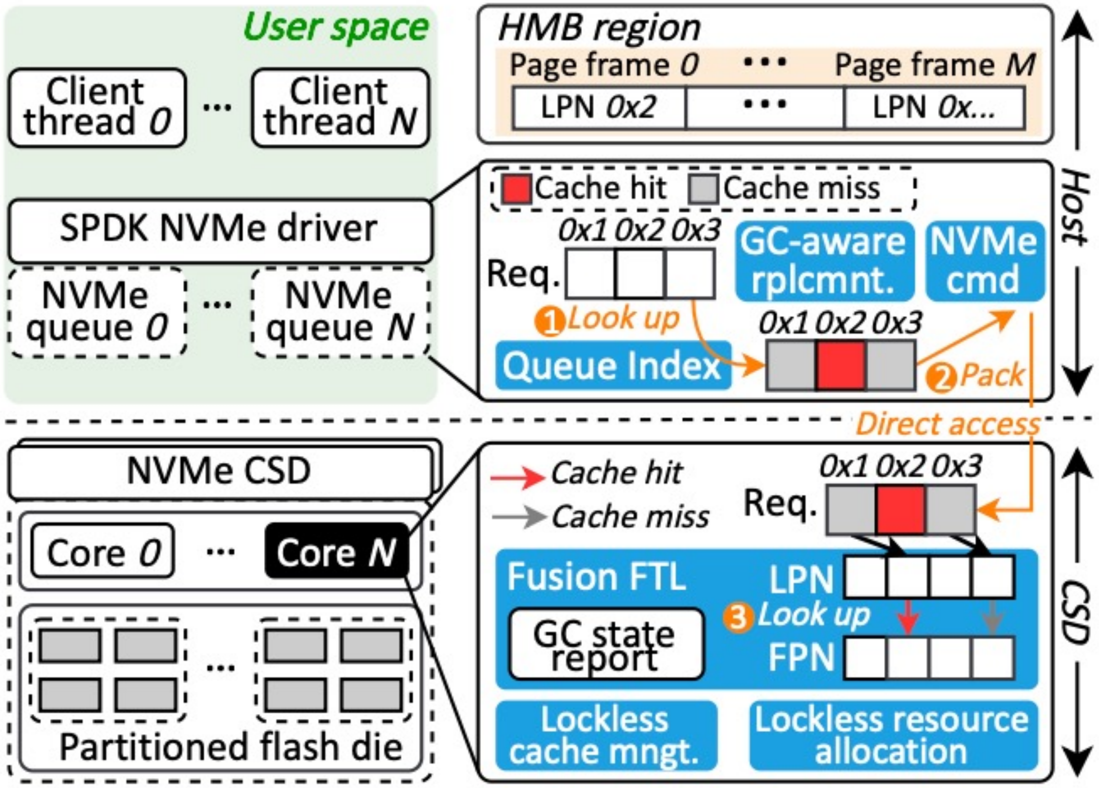


Figure 6: Overview of ScalaCache.

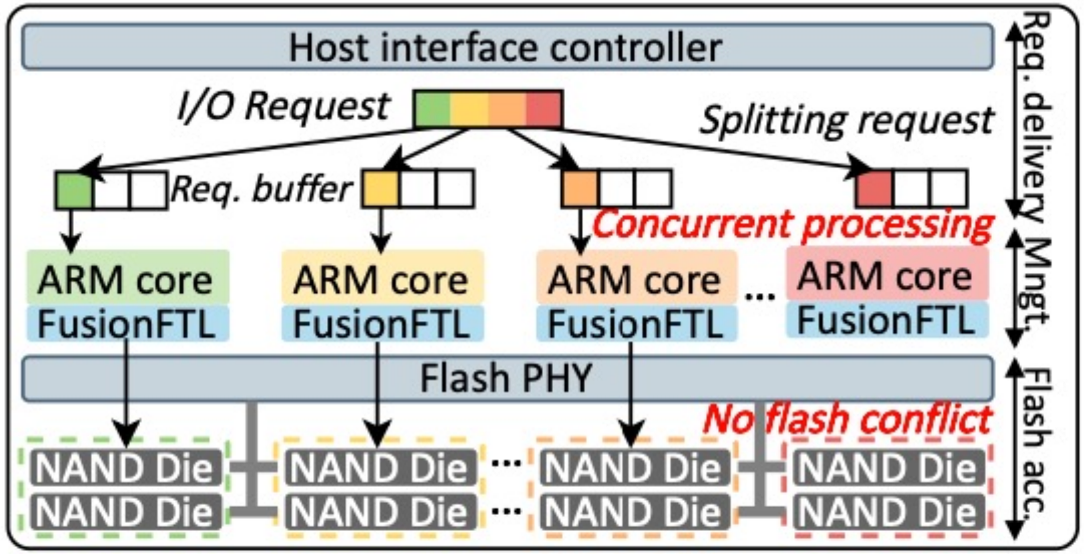


Figure 8: Concurrent I/O processing within CSD.

5. Evaluation

Set up

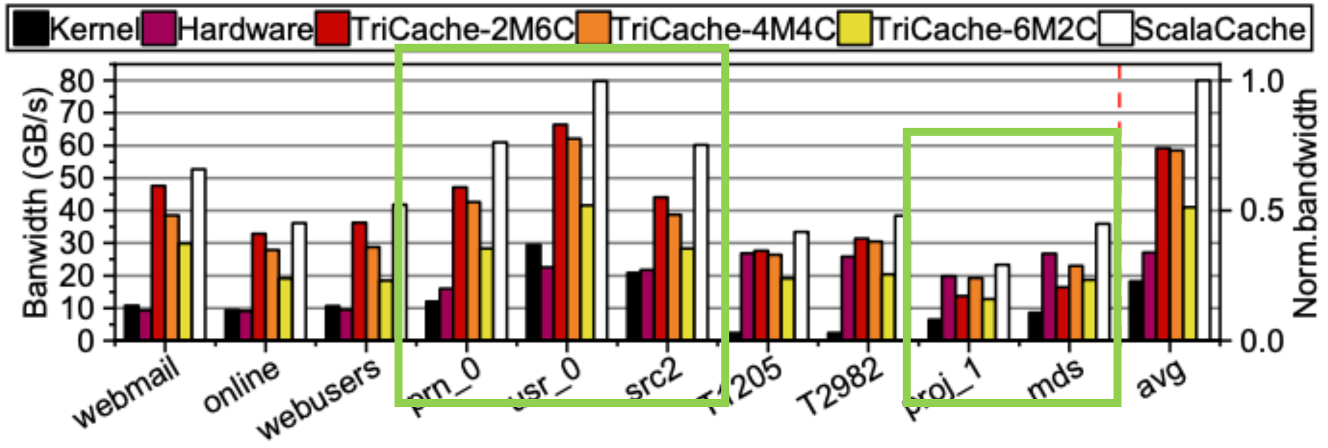
Host system		FEMU		Software	
CPU	AMD EPYC 9654	VM	24 Core / 128 GB DRAM	Linux kernel	6.8
	96 Core / 2.4 GHz	Flash	Rd./Prog.: 18/35 us		
Mem.	768 GB DRAM		8 Channel / 4 Die / 1024 Block / 512 Page / 4 KB	SPDK	22.01.2
				Cache size	18.75%

Table 1: System configurations.

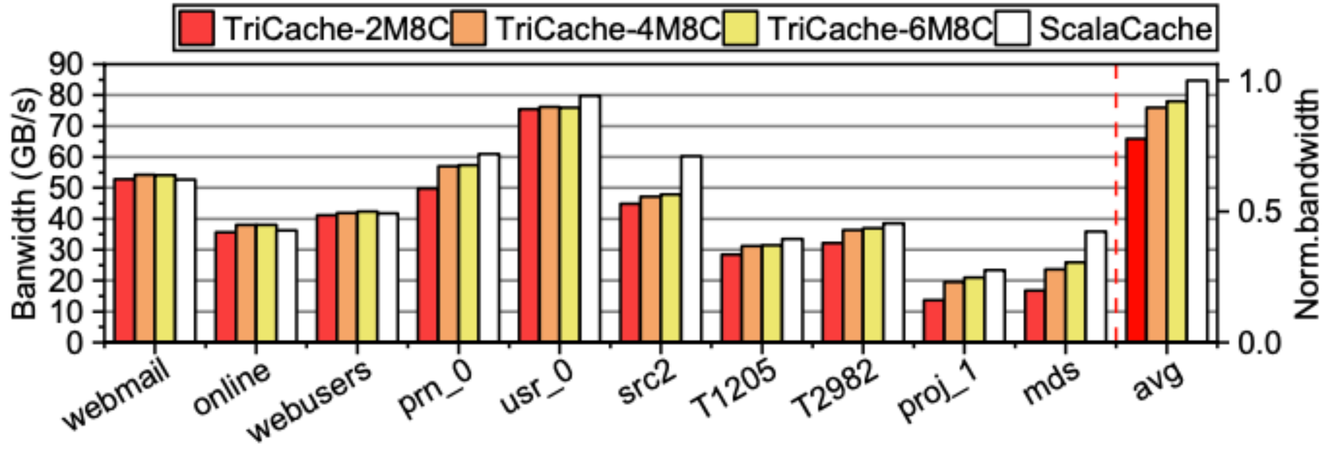
Trace	Req cnt. (Mops)	Avg req size (KB)	Data size (GB)	Hit ratio	Randomness	Hotness [19] (reuse dis. (GB))
webmail	7.80	4	29.74	0.96	0.22	0.21
online	5.70	4	21.80	0.94	0.26	0.62
webusers	5.70	4.22	22.90	0.71	0.30	0.40
prn_0	5.59	11.09	59.09	0.89	0.77	0.81
usr_0	2.24	22.66	48.37	0.96	0.89	1.05
src2	3.37	34.19	109.97	0.90	0.95	0.47
T1205	0.33	160.10	50.47	0.61	0.89	1.45
T2982	1.06	65.55	66.02	0.67	0.97	2.59
proj_1	23.64	34.42	775.93	0.78	0.87	1.02
mds	2.85	36.56	99.33	0.76	0.91	0.50

Table 2: The characteristics of examined workloads.

Performance



(a) Bandwidth comparison with fixed (8) host CPU cores.

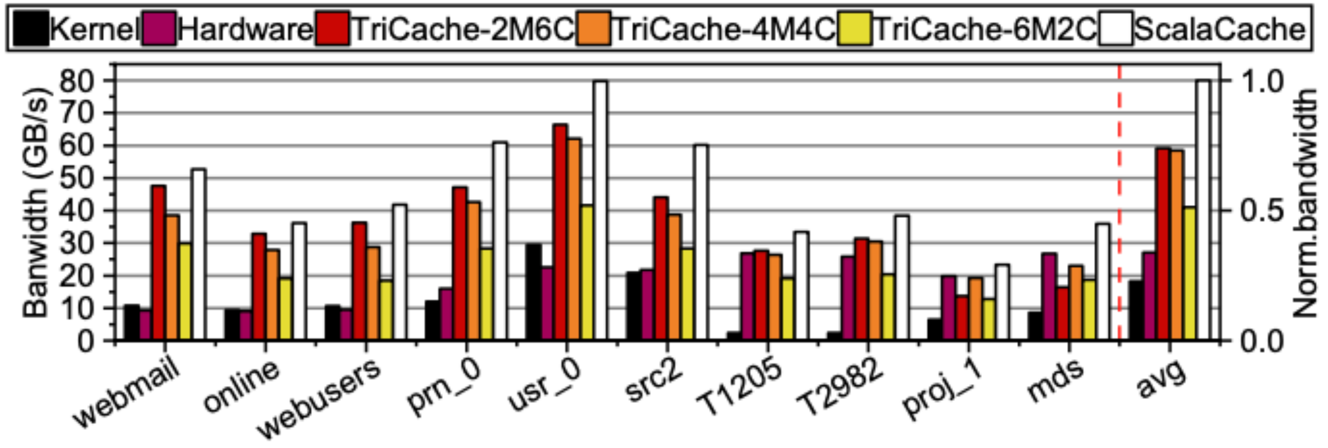


(b) Bandwidth comparison with fixed (8) client threads.

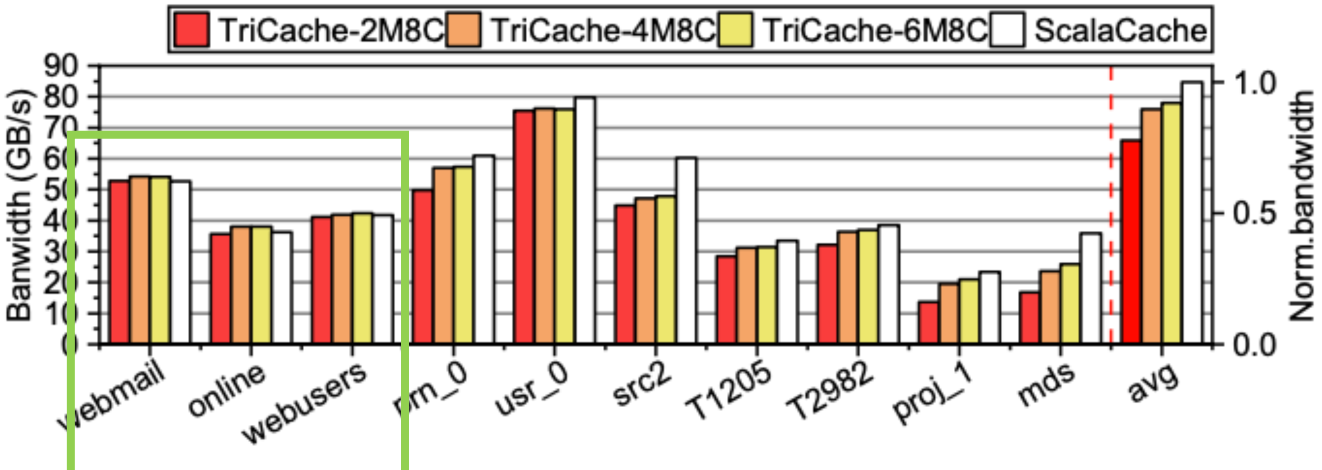
- 8 Host CPU
- 5.12×and 1.95×bandwidth improvement compared to Kernel and Hardware
- 35.30% and 94.78% bandwidth improvement compared to TriCache

5. Evaluation

Performance



(a) Bandwidth comparison with fixed (8) host CPU cores.



(b) Bandwidth comparison with fixed (8) client threads.

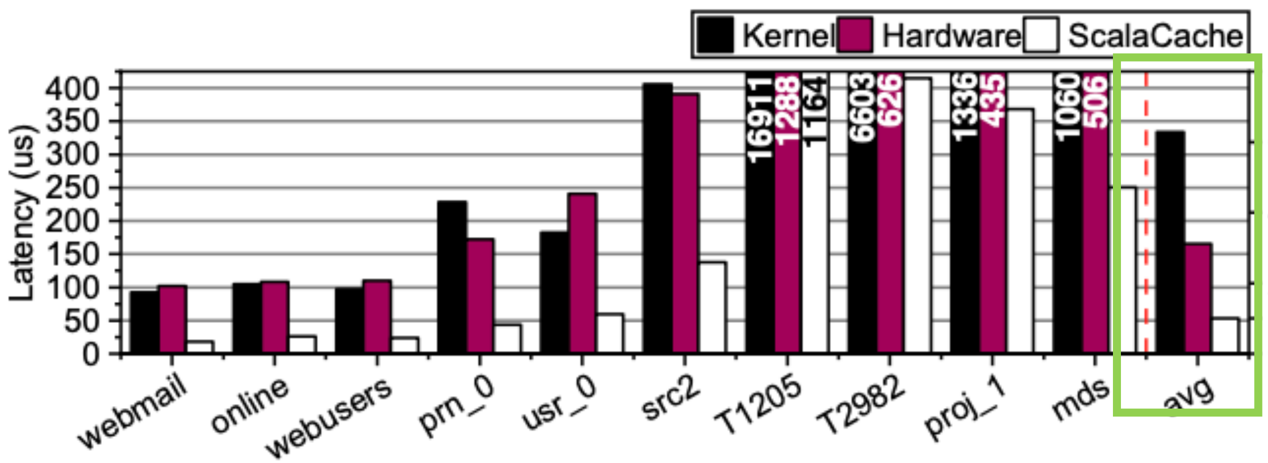
- Fixed 8 core for client threads
- outperforms 2M8C by 29%

Trace	Req cnt. (Mops)	Avg req size (KB)	Data size (GB)	Hit ratio	Randomness	Hotness [19] (reuse dis. (GB))
webmail	7.80	4	29.74	0.96	0.22	0.21
online	5.70	4	21.80	0.94	0.26	0.62
webusers	5.70	4.22	22.90	0.71	0.30	0.40
prn_0	5.59	11.09	59.09	0.89	0.77	0.81
usr_0	2.24	22.66	48.37	0.96	0.89	1.05
src2	3.37	34.19	109.97	0.90	0.95	0.47
T1205	0.33	160.10	50.47	0.61	0.89	1.45
T2982	1.06	65.55	66.02	0.67	0.97	2.59
proj_1	23.64	34.42	775.93	0.78	0.87	1.02
mds	2.85	36.56	99.33	0.76	0.91	0.50

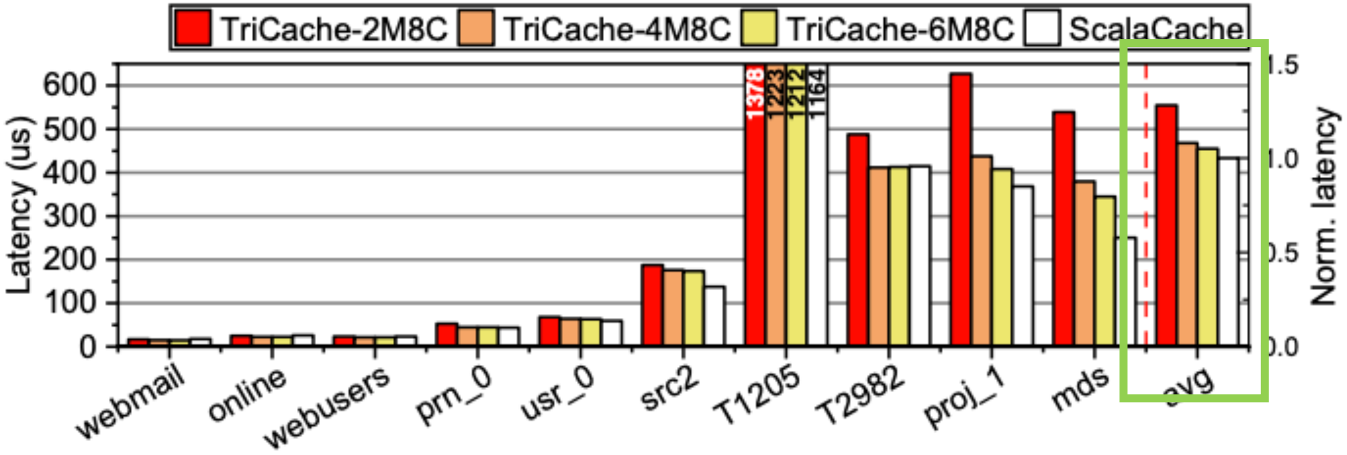
Table 2: The characteristics of examined workloads.

5. Evaluation

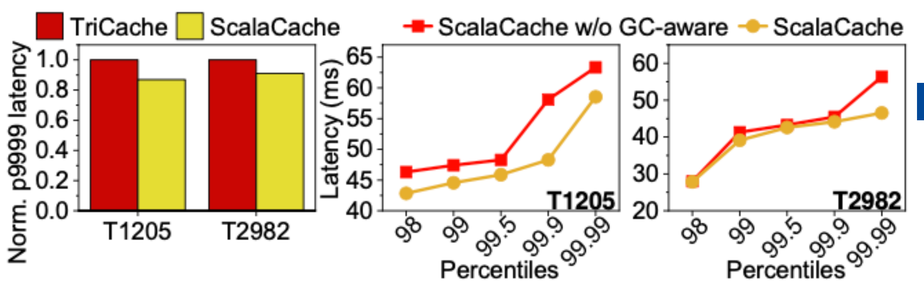
Latency



(a) Latency comparison with fixed (8) host CPU cores.



(b) Latency comparison with fixed (8) client threads.



(a) Tail latency comparison. (b) Improvement of GC-aware replacement policy.

Figure 13: Tail latency comparison.

- 78.13% lower latency
  Compare to Kernel
- 56.07% lower latency
  Compare to Hardware
- 53.50%, 33.97%, and 27.33% lower latency
  Compare to TriCache
- 11% 99.99th latency reduction compared to TriCache

5. Evaluation

Breakdown

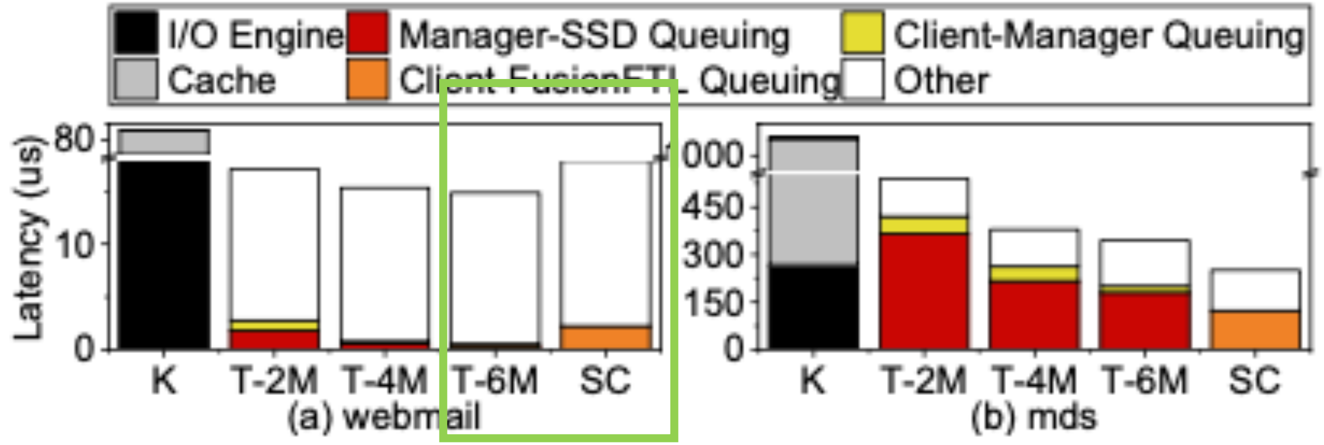


Figure 14: I/O latency breakdown.

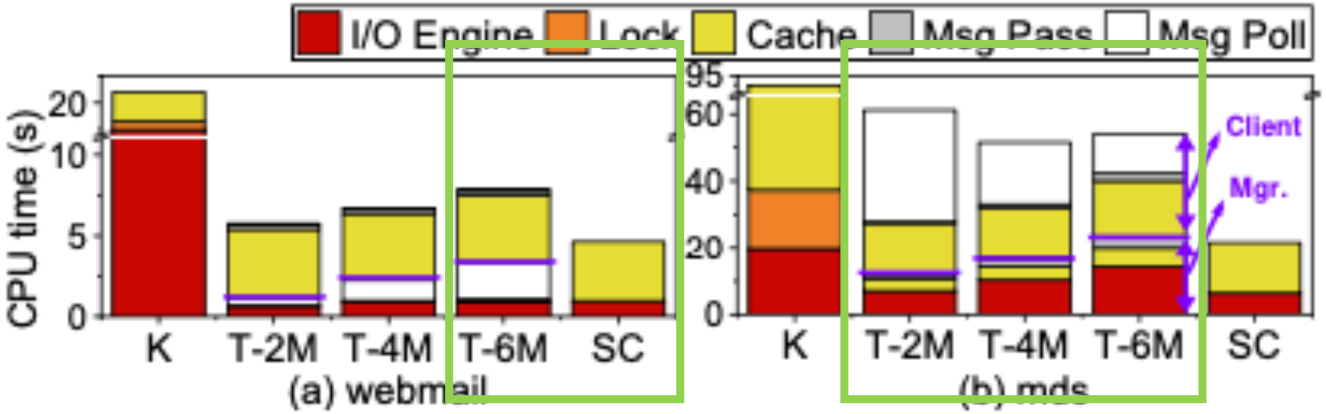


Figure 15: CPU time breakdown.

- Other is an action unrelated to the cache
- Reduced 47.09%
- But,
  Slow operation of CSD affects IO performance
- SC does not have msg poll
- TriCache has
  request fragmentation

5. Evaluation

Scalability

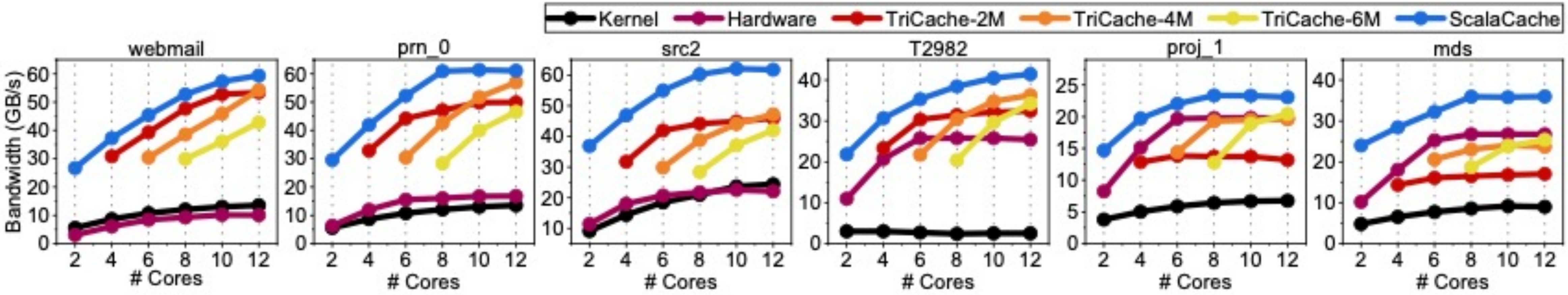
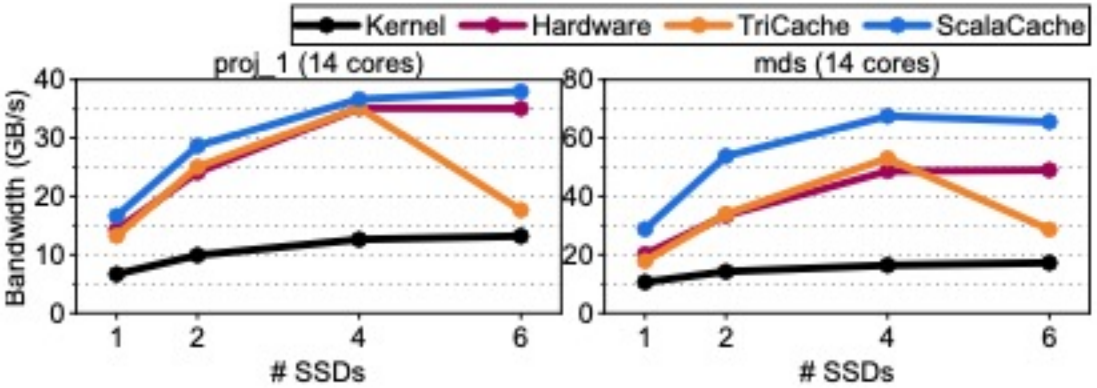
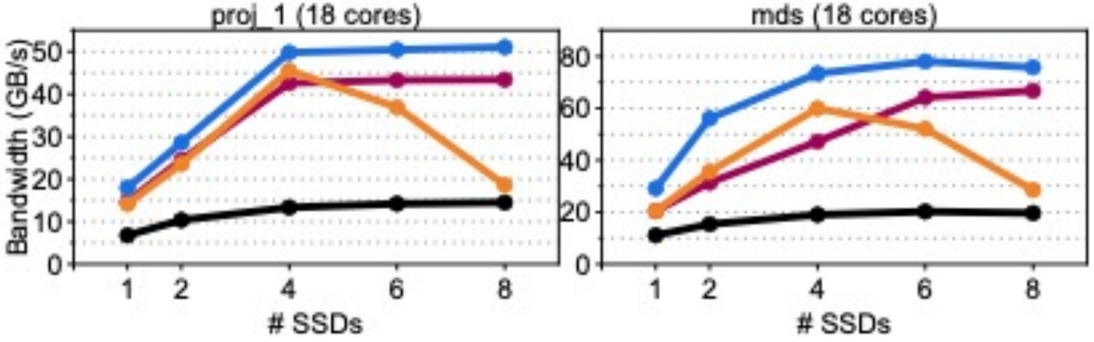


Figure 17: Scalability with host CPU cores.



(a) Performance with 14 host CPU cores.



(b) Performance with 18 host CPU cores.

Figure 18: Scalability with varying CPU cores and SSDs.

6. Conclude

- **User-space cache with software-hardware collaboration**:
  Take advantage of both user-space design and software-hardware collaboration.

- **Lightweight cache management in CSD**:
  They propose a lightweight index structure called FusionFTL to address the difficulty of delegating cache management to CSD.

- **Enabling concurrent I/O processing for CSD**:
  They build a lock-free resource allocation framework within CSD to enable multiple CSD cores to access resources without locks.

ScalaCache:
Scalable User-Space Page Cache Management with Software-Hardware Coordination

Peng, L., An, Y., Zhou, Y., Wang, C., Li, Q., Cheng, C., & Zhang, J.

In 2024 USENIX Annual Technical Conference (USENIX ATC 24)

Thank you!
Q & A ?

2024.10.02

Presentation by Choi, Gunhee
choi_gunhee@dankook.ac.kr