



KVFTL - Optimization of Storage Space Utilization for Key-Value-Specific Flash Storage Devices

ASP-DAC 2017

Yen-Ting Chen, Ming-Chang Yang, Yuan-Hao Chang, Tseng-Yi Chen,
Hsin-Wen Wei, and Wei-Kuan Shih

National Tsing Hua University, Department of Computer Science, Taiwan
(R.O.C)

Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

Introduction

- KV-store has gained popularity in **large-scale and data-driven storage applications**.
 - E.g., Google's BigTable and Amazon's Dynamo
- KV-specific storage devices have been commercialized.

- *2 Ethernet ports*
- *Kinetic key/value API*
 - *Value = get (key)*
 - *Put(key, value)*
 - *Delete(key)*
 - *Key :: 1 byte to 4 KiB*
 - *Value :: 0 bytes to 1 MiB*

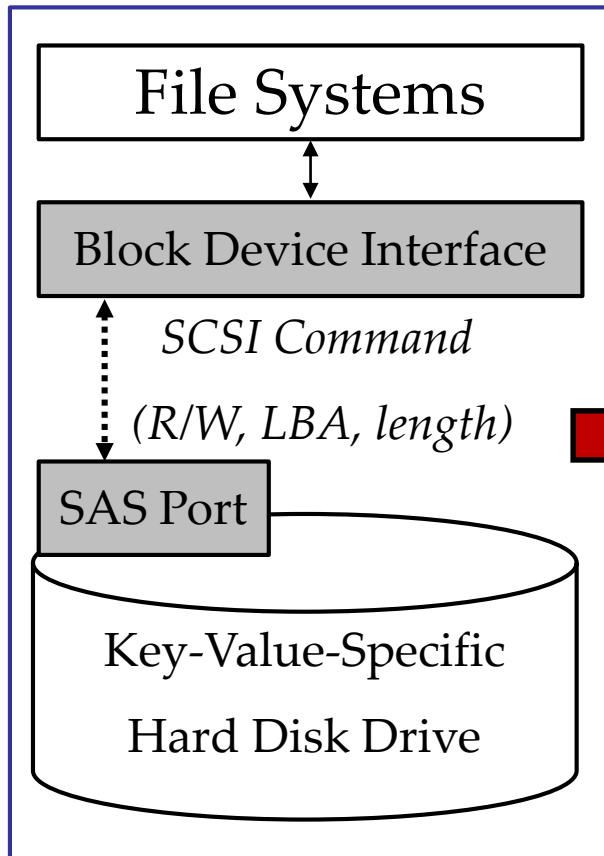
Seagate

Kinetic Open Storage



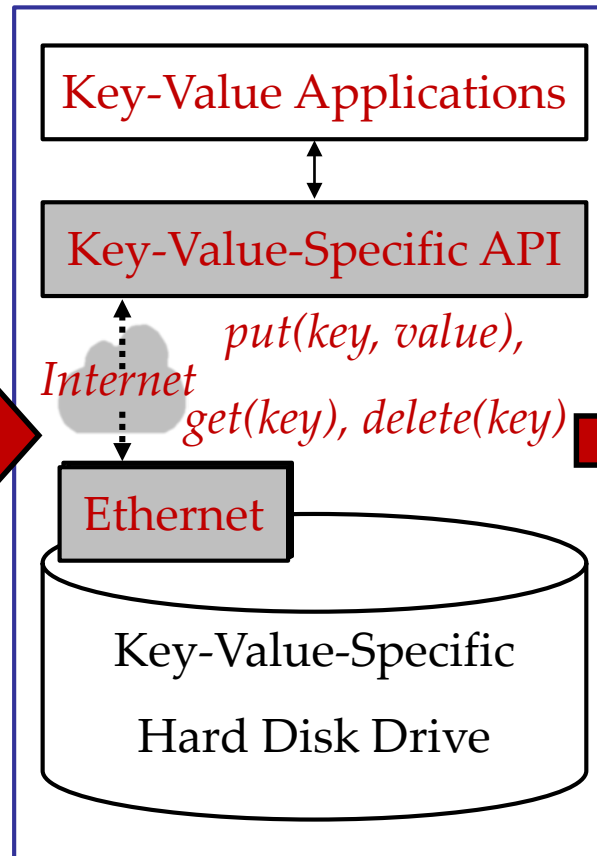
Evolution of Storage Devices

HDD



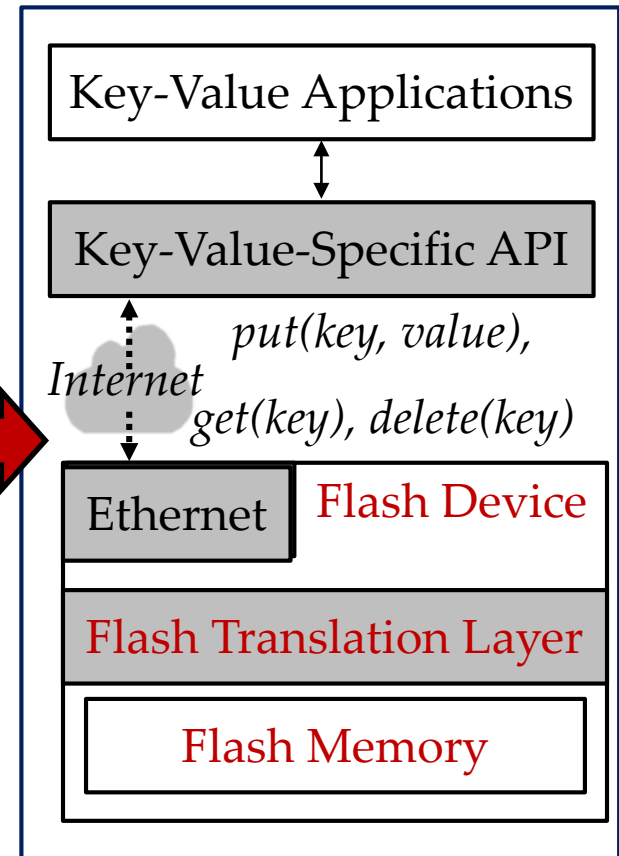
Seagate Drive

KVHDD



Future Work

KVSSD

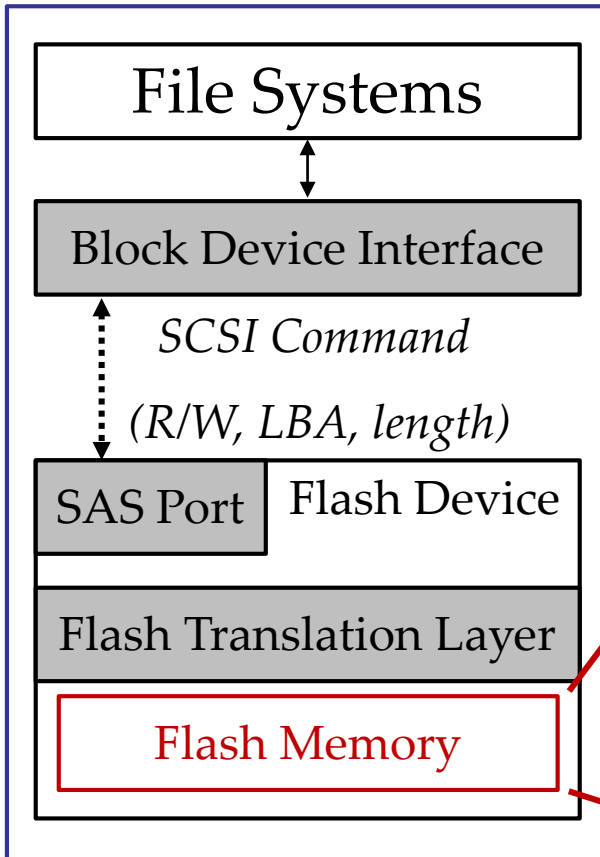


Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

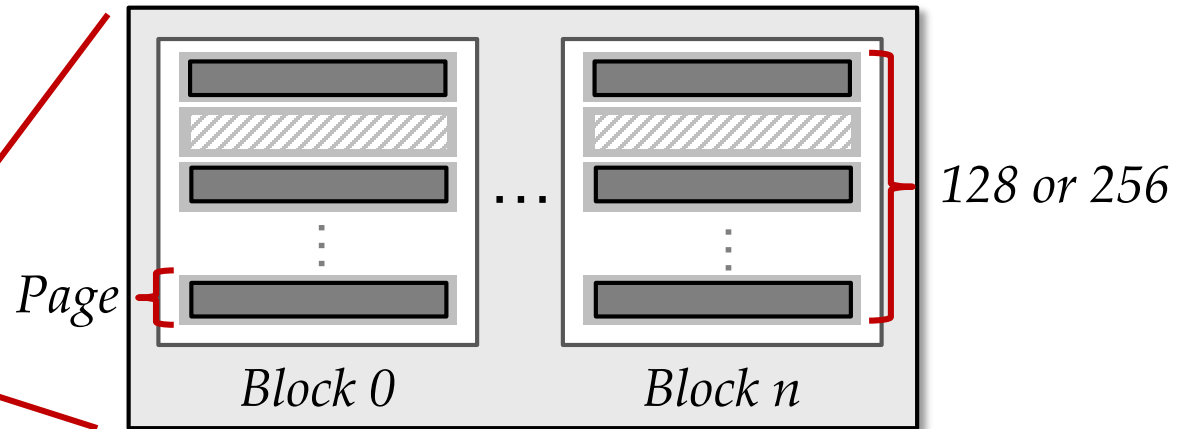
Basic of Raw Flash Memory

SSD



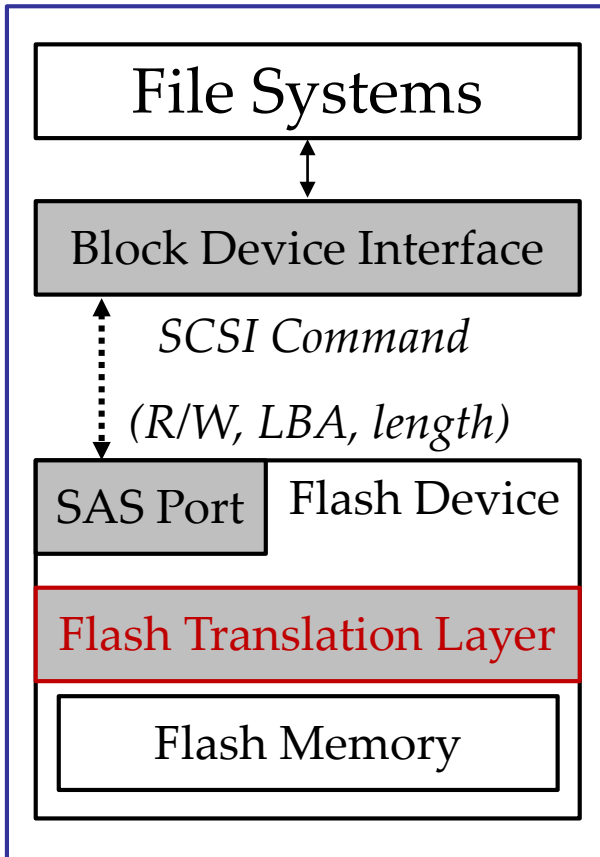
- **Page**
 - Basic **R/W** operation unit
- **Block**
 - Basic **Erase** operation unit

Physical Flash Memory



Basic of Flash Translation Layer

SSD

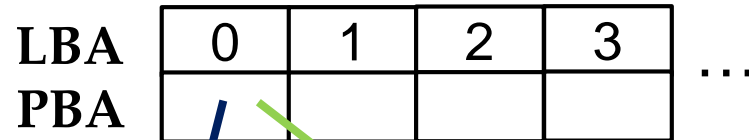


Address Translation

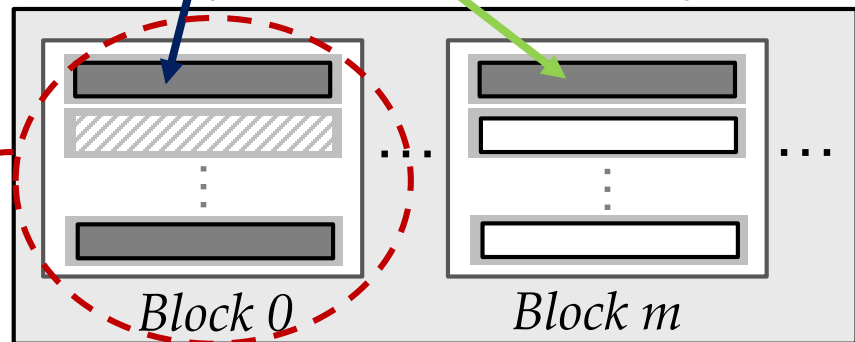
- Write-once property
- Out-place updates

Garbage Collection

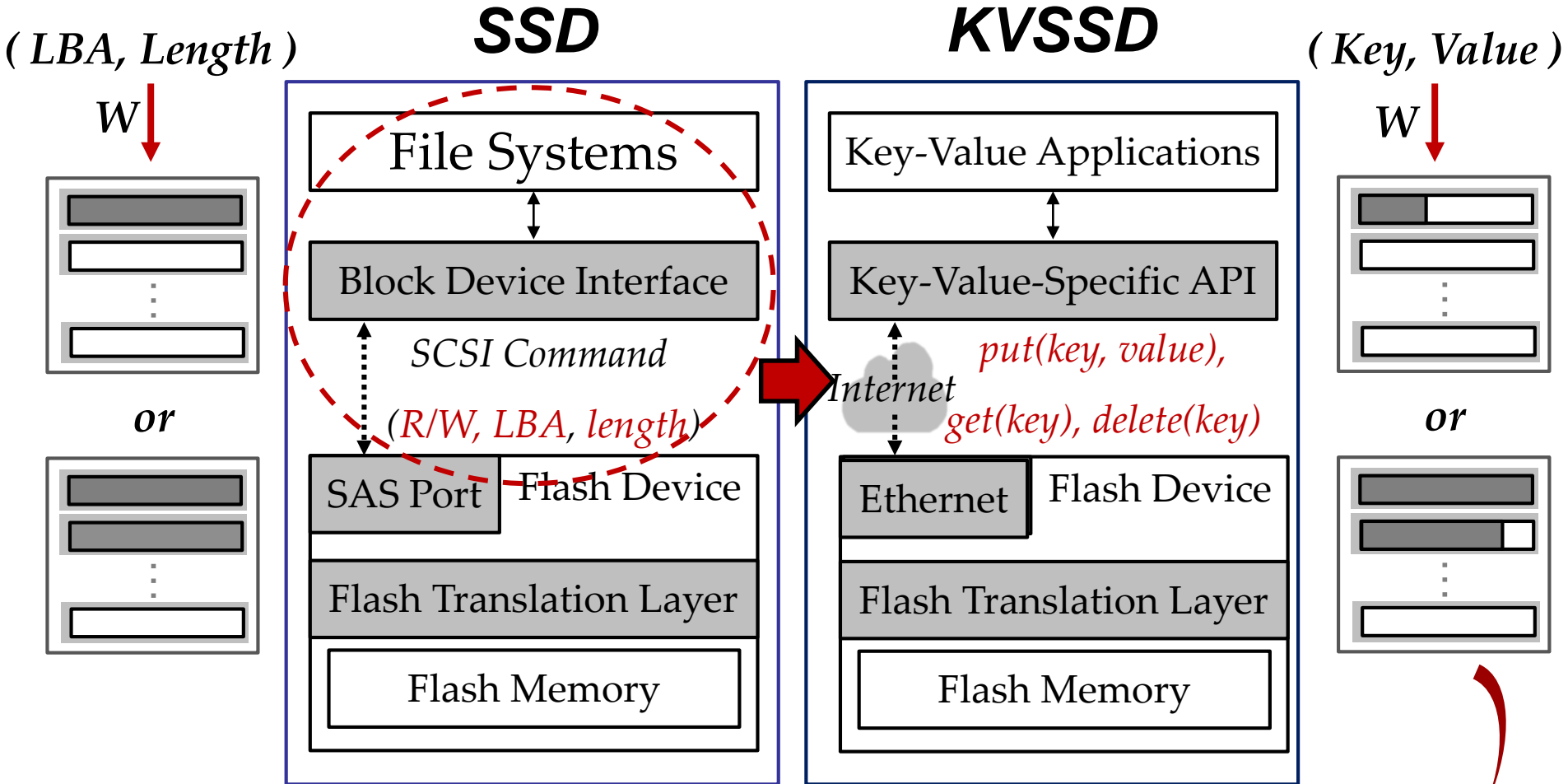
- Bulk Erase
- Live-page copyings



Physical Flash Memory



Motivation



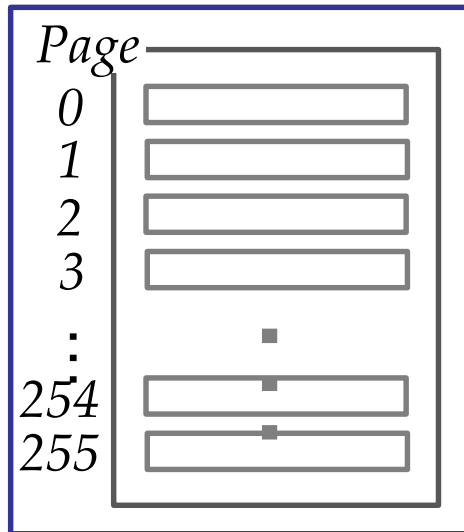
Space Utilization Issue!

Outline

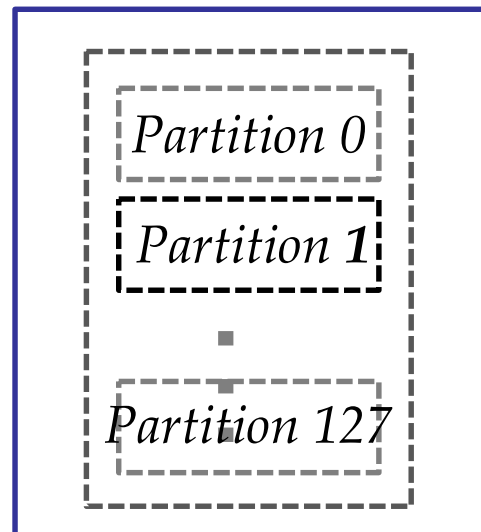
- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

New Storage Units: Partition

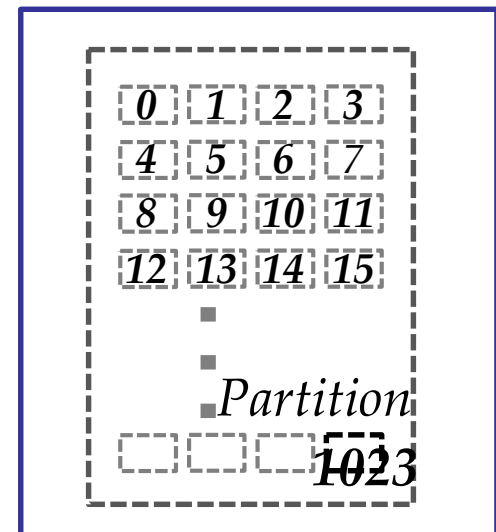
- To avoid wasting storage space on **storing variable-sized values**, we propose to create **a range of different sizes of partitions**.



Flash Block
Page Size: $2^{14} B$



Partitioned Block
Page Size: $2^{15} B$



Partitioned Block
Page Size: $2^{12} B$

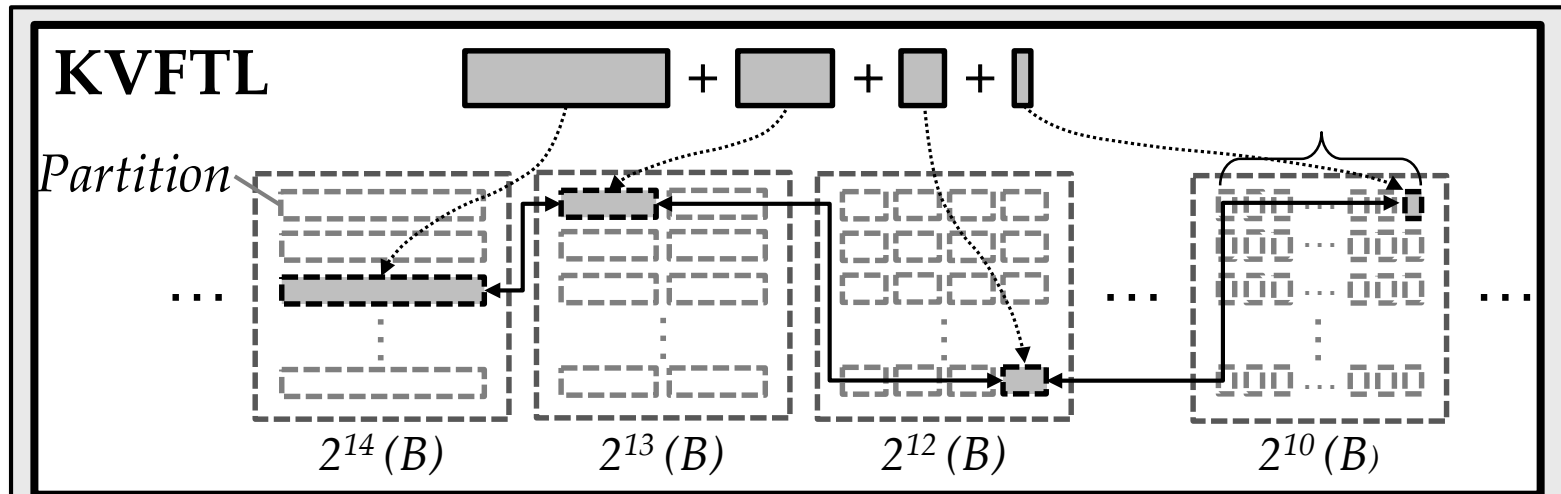
Overview of KVFTL

- Break the common rule to only store/access data over flash memory in the unit of fixed-sized flash pages.

Key-Value Application
(Key, Value)

KV Flash Device

↕ *get(key) / put(key, value) / delete(key)*



↕ *read(page) / write(page) / erase(block)*

Physical Flash Memory

Outline

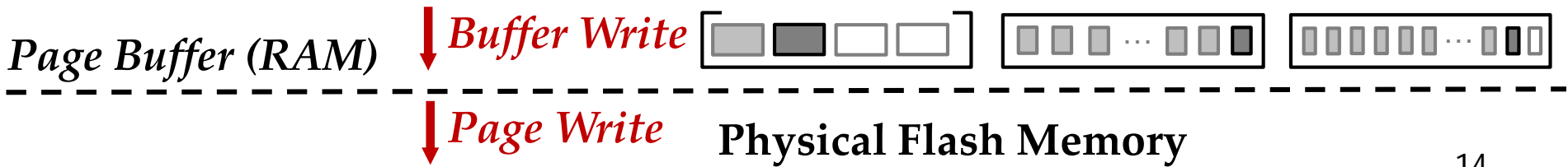
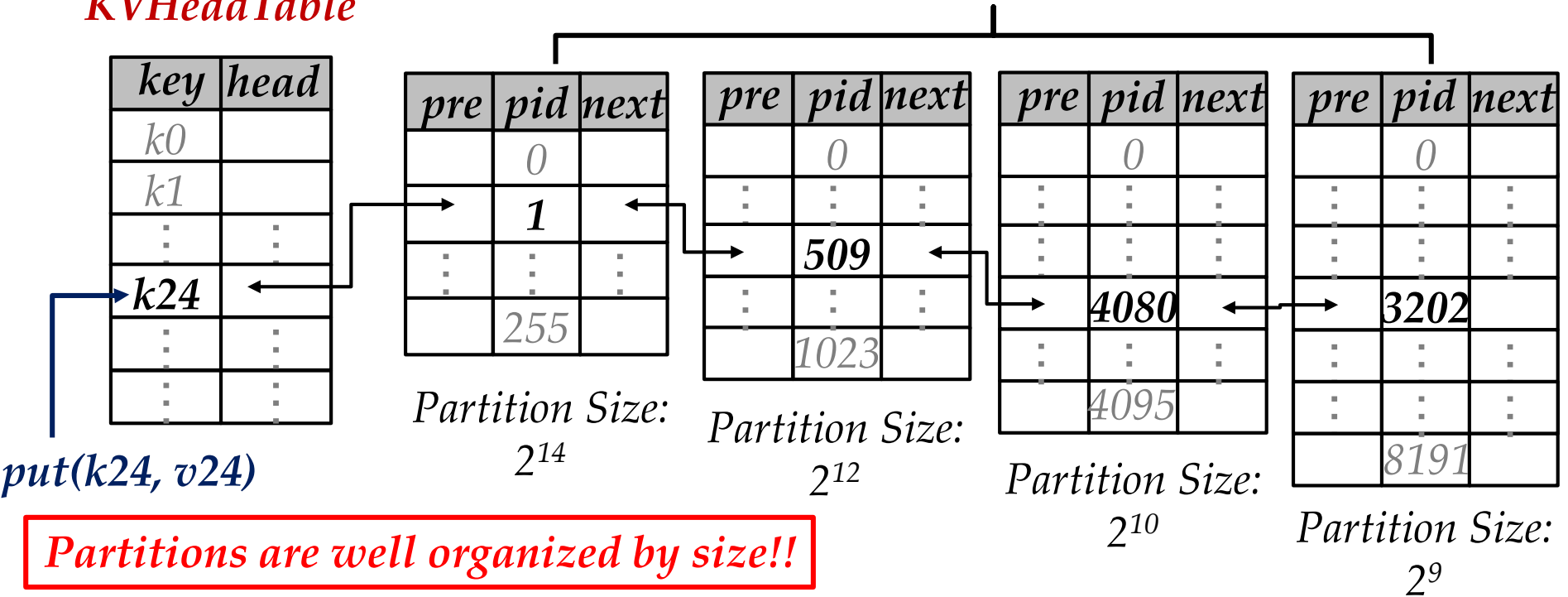
- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - **Key Operations**
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

Put (Key, Value)

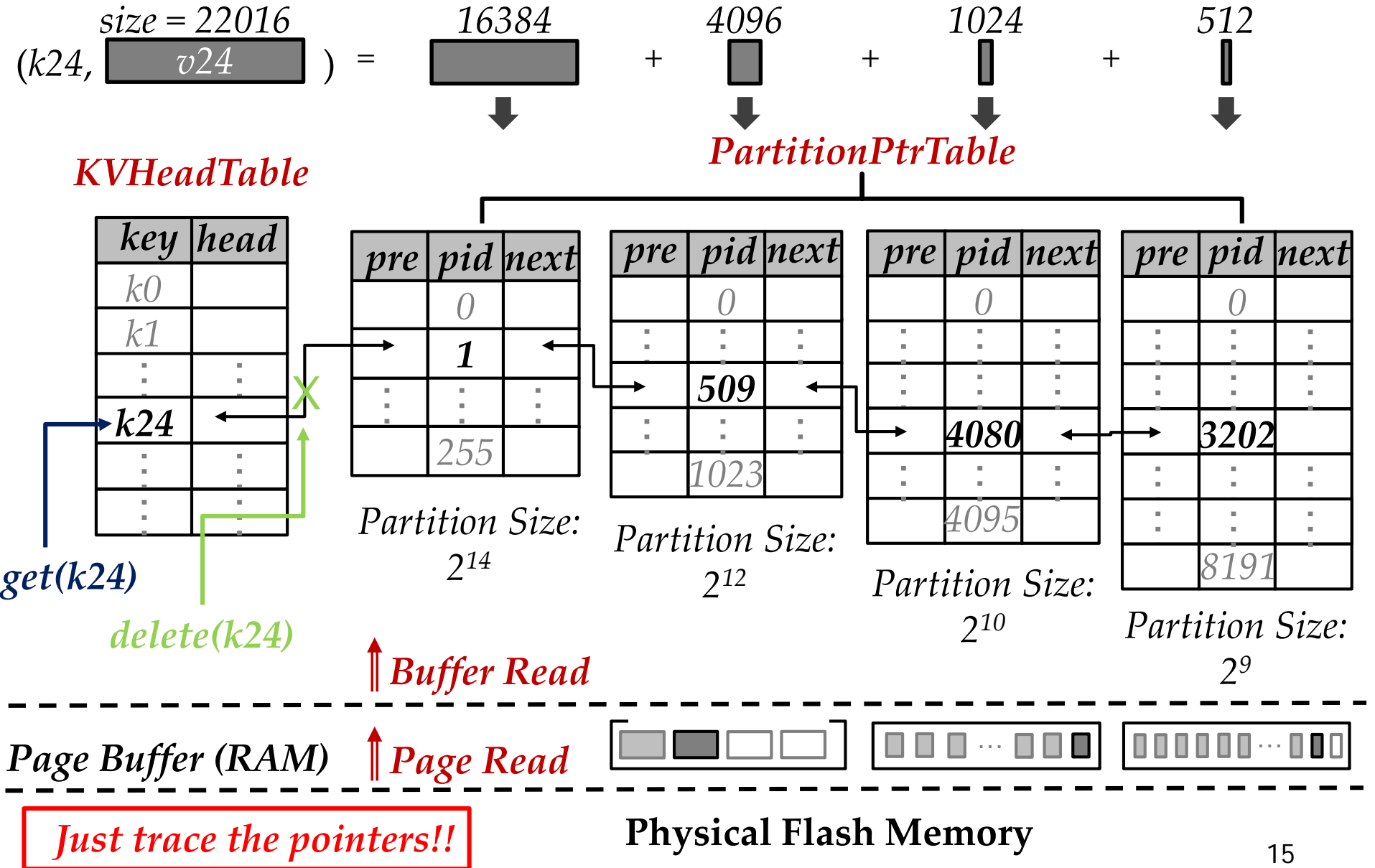
$$(k_{24}, \overset{\text{size} = 22016}{\boxed{v_{24}}}) = \boxed{16384} + \boxed{4096} + \boxed{1024} + \boxed{512}$$

KVHeadTable

PartitionPtrTable



Get (Key, Value) & Delete (Key, Value)



Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

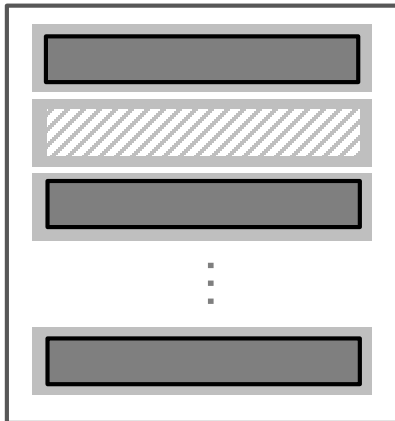
Efficiency of Recycling a Block

$$W_i = \frac{\#invalid\ partitions * partition\ size\ of\ block_i}{\#required\ live\ page\ copyings} = Efficiency$$

FTL

More invalid **pages**

➔ Higher efficiency ✓

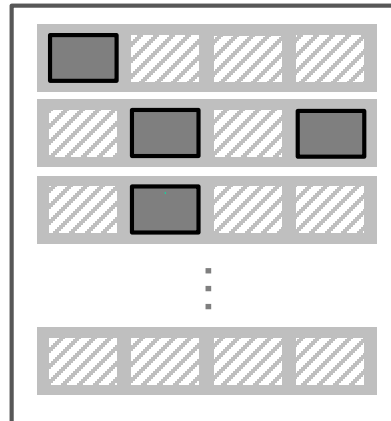


Flash Block

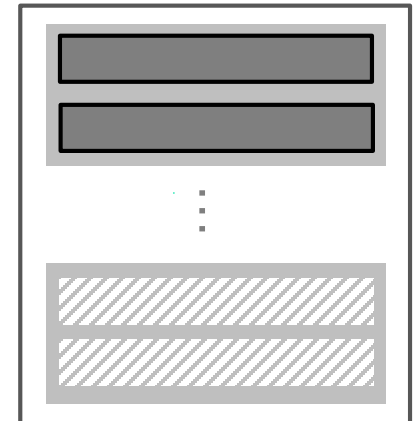
KVFTL

More invalid **partitions**

➔ Higher efficiency ??



Partitioned Block (2¹²)



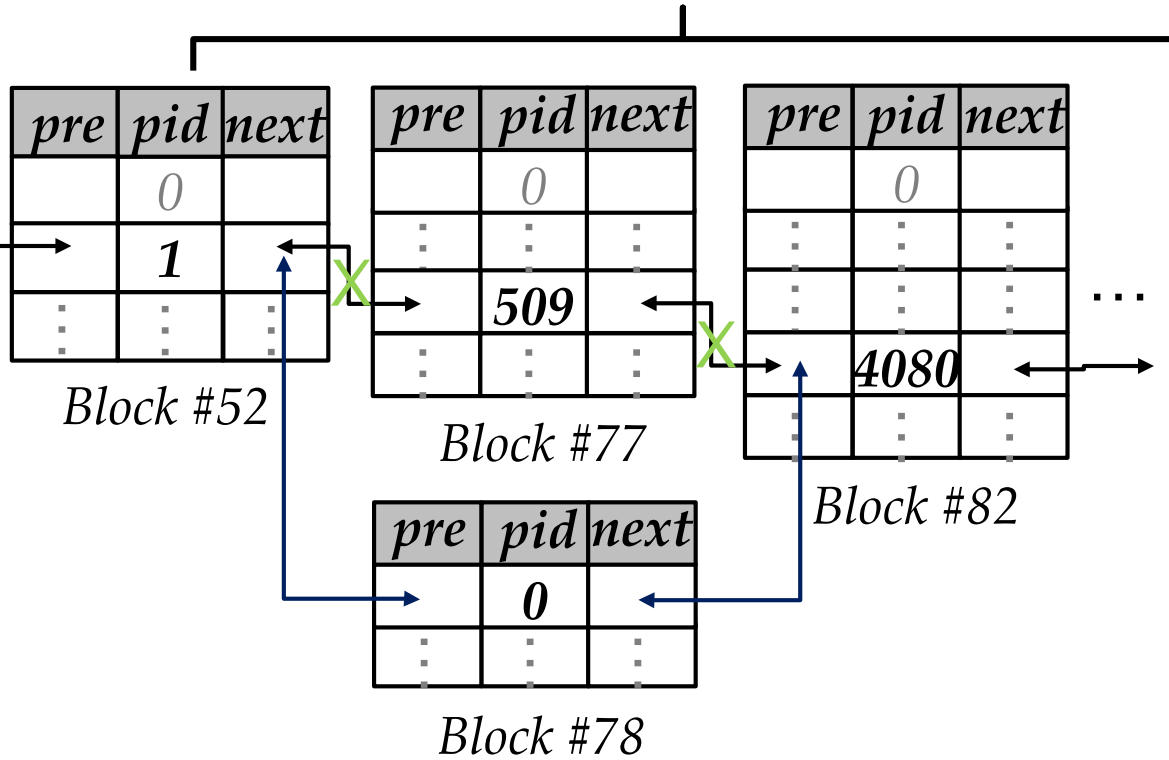
Partitioned Block (2⁹)

Example of Garbage Collection

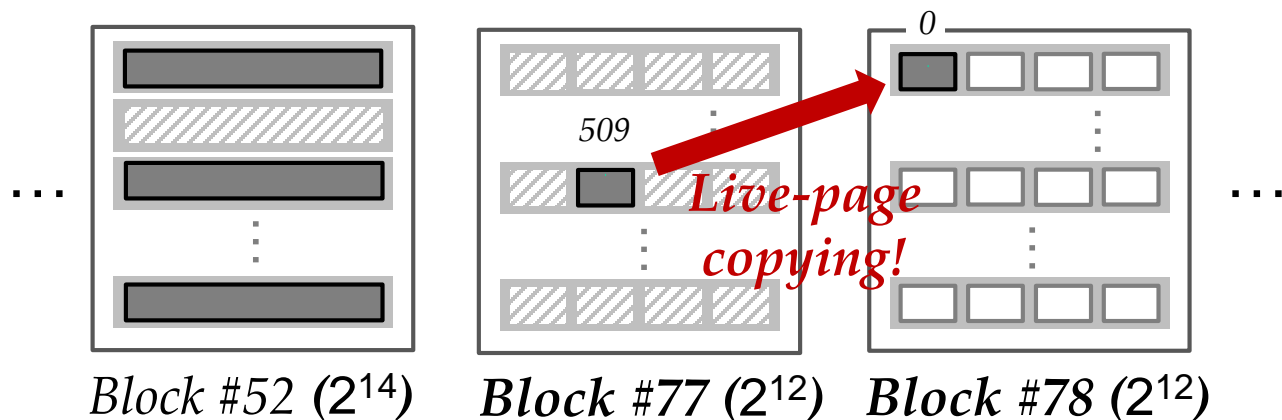
KVHeadTable

| key | head |
|-----|------|
| k0 | |
| k1 | |
| ⋮ | ⋮ |
| k24 | |
| ⋮ | ⋮ |
| ⋮ | ⋮ |

PartitionPtrTable



Physical
Flash
Memory



Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

Experimental Setup

- Simulated key-value-specific flash device:

| | | | |
|---------------|-----------------|---------------|--------------|
| Page size | 16KB | Read latency | 115 μ s |
| Block size | 4MB (256 pages) | Write latency | 1600 μ s |
| Serial access | 180 μ s | Erase latency | 3ms |

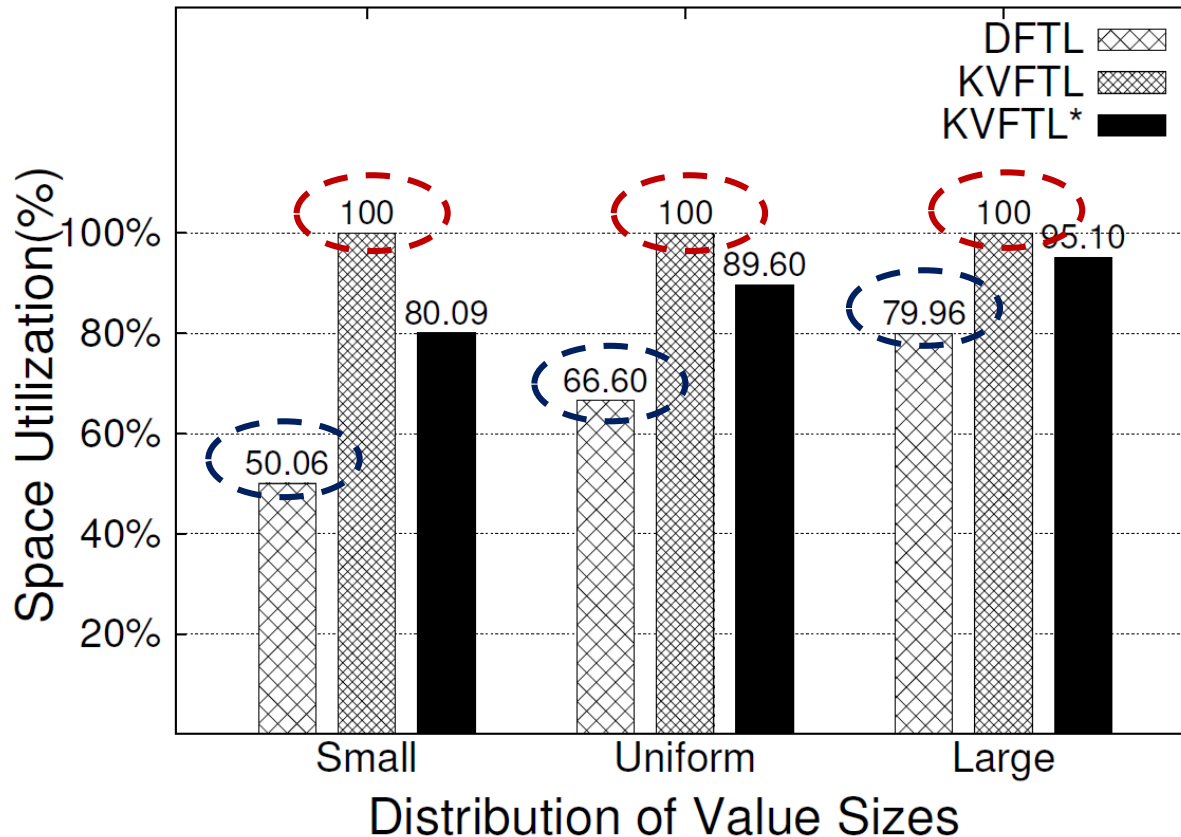
- Two subjects for comparison:
 - *DFTL*: Page-based FTL
 - *KVFTL** (*KVFTL-12*): Variation of *KVFTL*
- Key-Value workloads: YCSB on Hbase
- Three representative distributions:
 - Small
 - Large
 - Uniform

Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - **Experimental Results**
- Conclusion

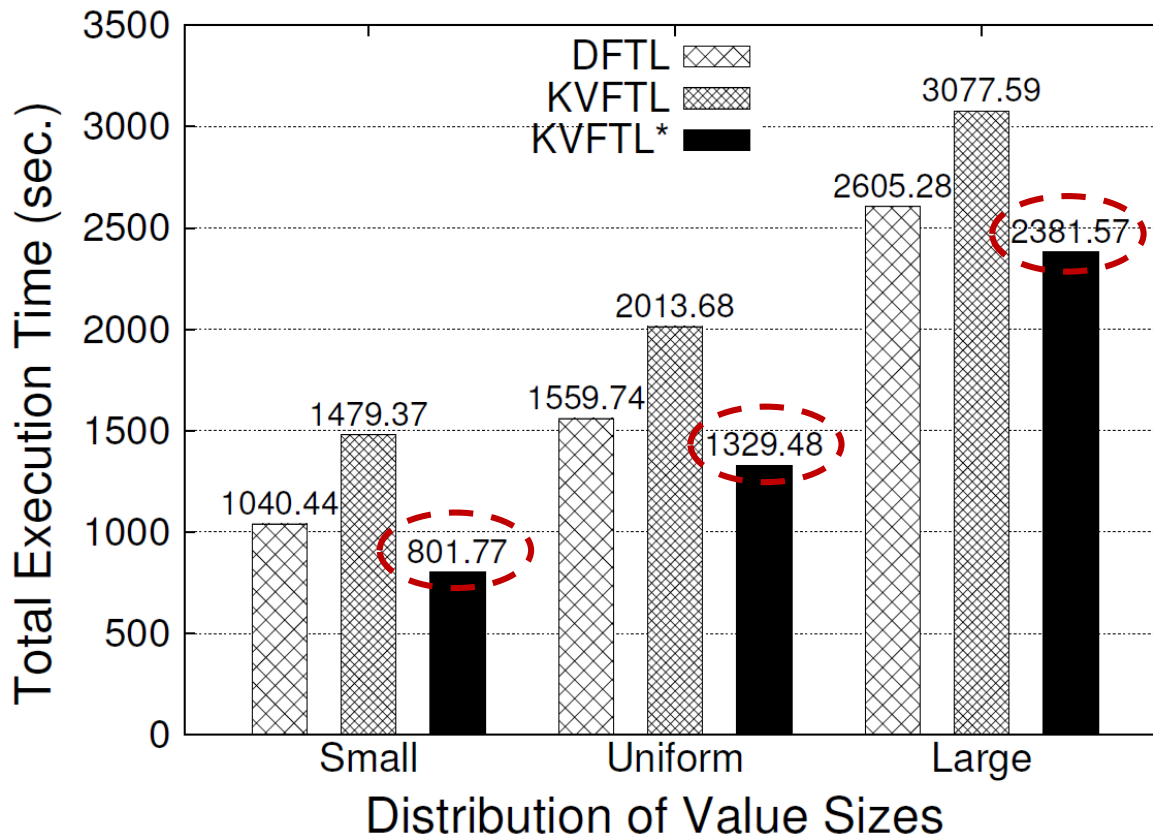
Results of Storage Space Utilization

- *KVFTL*: Optimized space utilization
- *DFTL*: Low space utilization



Results of Total Execution Time

- *KVFTL** could at most reduce 23% and 46% total execution time than that of *DFTL* and *KVFTL*.



Outline

- Introduction
- Background and Motivation
- KVFTL
 - Basic Concepts
 - Key Operations
 - Garbage Collection
- Evaluation
 - Experimental Setup
 - Experimental Results
- Conclusion

Conclusion

- A new key-value flash translation layer for key-value-specific flash storage devices.
- Optimal storage space utilization.
- Great flexibility to trade little space utilization for better performance.

Thanks for Listening!
Q & A

Yen-Ting Chen

Appendix

New Data Access Model

- The size of the read/written data of a block device could be only multiple times of the size of a logical block.
 - E.g., 512 B
- The size of the accessed value from/to a key-value-specific device can be variable length.
 - Varied between 0 bytes and 1 MiB.

Trend of Future Key-Value-Specific Storage Device

- In order to achieve a higher degree of performance and satisfy the demands for some high performance applications, the **specific flash-based solid state drives (SSD) for key-value store applications** would be commercialized in a foreseeable future.

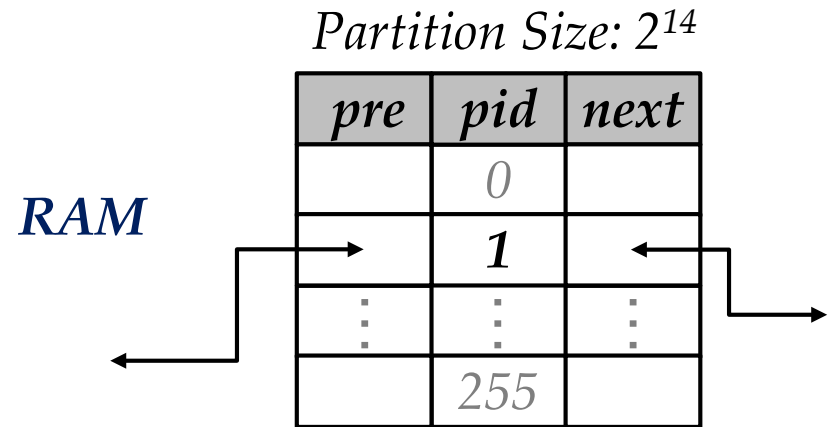
Motivation

- The variable-length data access model of key-value stores might result in **low storage space utilization** in the flash-based storage devices.
- **Flash Translation Layer (FTL) must be re-designed** to meet this new data access model, so as to resolve the low storage space utilization issue, namely the ***KVFTL***.

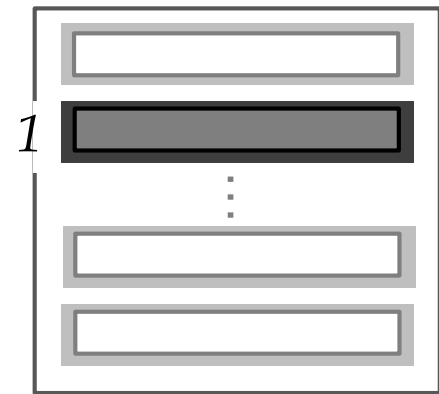
PartitionPtrTable

- To keep track of each separated partitions of a value, *KVFTL* associates these partitions with **double link pointer**.

PartitionPtrTable



*Physical
Flash
Memory*



Block #8

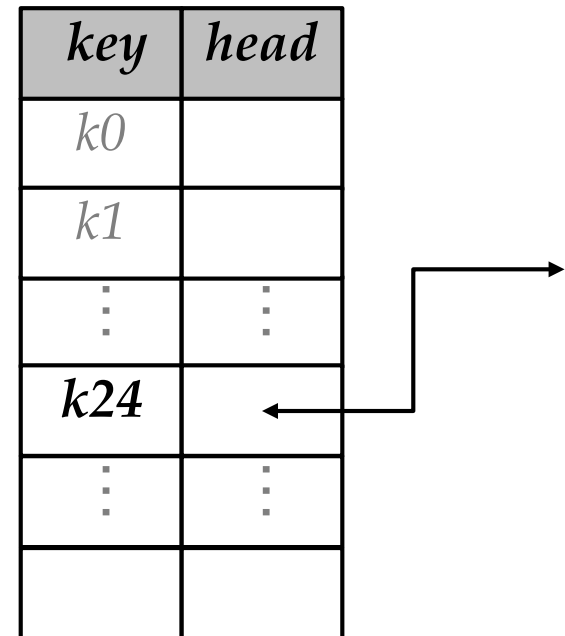
KVHeadTable

- To keep track of the first partition of each value, *KVFTL* maintained a *KVHeadTable* for all keys.

RAM

KVHeadTable

| <i>key</i> | <i>head</i> |
|------------|-------------|
| <i>k0</i> | |
| <i>k1</i> | |
| ⋮ | ⋮ |
| <i>k24</i> | |
| ⋮ | ⋮ |
| | |



The diagram shows a table with two columns: 'key' and 'head'. The 'key' column contains 'k0', 'k1', '⋮', 'k24', '⋮', and an empty cell. The 'head' column contains empty cells, '⋮', an empty cell, '⋮', and an empty cell. An arrow points from the 'head' cell corresponding to 'k24' to the right, then up, then right, and finally down to a point in RAM.

Garbage Collection

- The partitioned-based garbage collection design could be invoked only when **the number of the remaining free blocks** in the device is **less than a predefined value**.
- When the garbage collection is invoked, the first step is to find **a worthwhile flash block** (also referred to as victim block) to recycle/reclaim.

Example of Garbage Collection

$$(k_{24}, \boxed{v_{24}}) = \boxed{16384} + \boxed{4096} + \boxed{1024} + \boxed{512}$$

KVHeadTable

PartitionPtrTable

| key | head |
|------------|------|
| <i>k0</i> | |
| <i>k1</i> | |
| ⋮ | ⋮ |
| <i>k24</i> | |
| ⋮ | ⋮ |
| ⋮ | ⋮ |

| pre | pid | next |
|-----|-----|------|
| | 0 | |
| | 1 | |
| ⋮ | ⋮ | ⋮ |
| | 255 | |

Partition Size:
 2^{14}

| pre | pid | next |
|-----|------|------|
| | 0 | |
| ⋮ | ⋮ | ⋮ |
| | 509 | |
| ⋮ | ⋮ | ⋮ |
| | 1023 | |

Partition Size:
 2^{12}

| pre | pid | next |
|-----|------|------|
| | 0 | |
| ⋮ | ⋮ | ⋮ |
| | 4080 | |
| ⋮ | ⋮ | ⋮ |
| | 4095 | |

Partition Size:
 2^{10}

| pre | pid | next |
|-----|------|------|
| | 0 | |
| ⋮ | ⋮ | ⋮ |
| | 3202 | |
| ⋮ | ⋮ | ⋮ |
| | 8191 | |

Partition Size:
 2^9

| pre | pid | next |
|-----|-----|------|
| | 0 | |
| | 1 | |
| ⋮ | ⋮ | ⋮ |
| | 255 | |

Experimental Setup

- The experiment was **conducted in a customized simulator** and a key-value-specific flash device of flash pages of 16 KB and blocks of 256 pages was under investigation.

| | | | |
|---------------|-----------------|---------------|--------------|
| Page size | 16KB | Read latency | 115 μ s |
| Block size | 4MB (256 pages) | Write latency | 1600 μ s |
| Serial access | 180 μ s | Erase latency | 3ms |

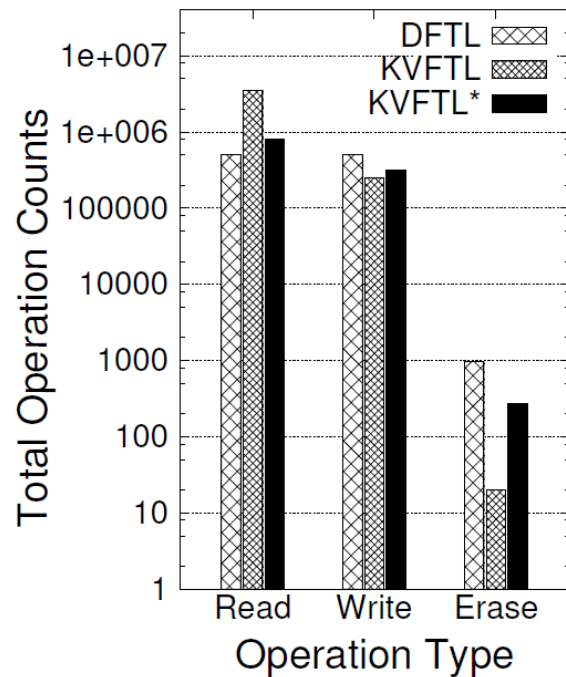
- We compare the proposed **KVFTL** with:
 - **TRDFTL**: Traditional page-based flash translation layer design
 - **KVFTL***: Variations of the original **KVFTL** design

Experimental Setup

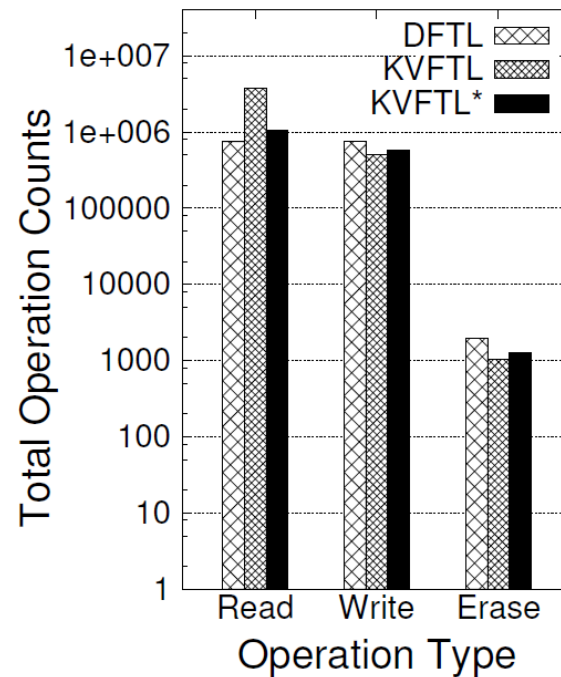
- We used the well-known **Yahoo Cloud Serving Benchmark** to generate the **key-value workloads** based on the HBase database.
- Three representative distributions of value sizes were adopted:
 - **Small**: Value sizes are mainly smaller than the flash page.
 - **Large**: Value sizes are mainly larger than the flash page.
 - **Uniform**: Value sizes uniformly varies from the smallest value size (i.e., 1 B) to the largest value size (i.e., 1 MB).

Results of Total Operation Counts

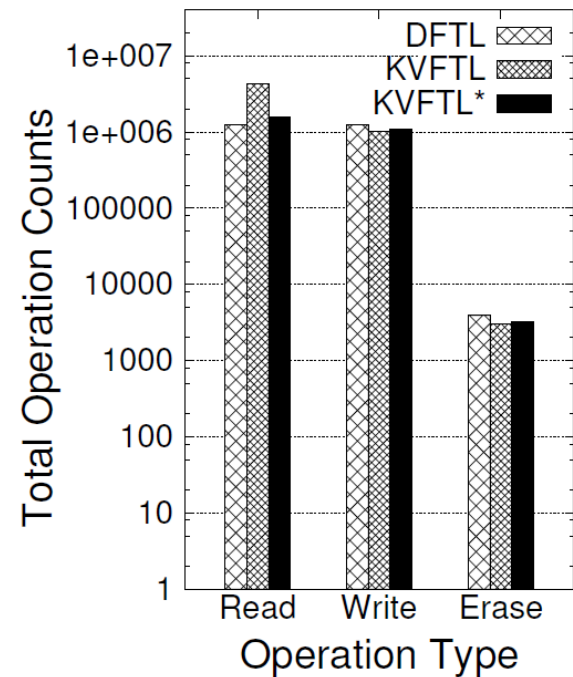
- *KVFTL** could achieve a great balance among the three types of operations.



Small



Uniform



Large