

Carl Duffy<sup>1</sup>

Sang-Hoon Kim<sup>2</sup>

Jin-Soo Kim<sup>1</sup>

<sup>1</sup>Systems Software &  
Architecture Lab,

Seoul National University

<sup>2</sup>Systems Software Lab,  
Ajou University

# The Key-Value SSD as a First-Class Citizen in the Operating System



# Key-Value SSDs vs Block SSDs

- Key-value stores running on block SSDs require several lookups from key to physical address

User Program



"userkey001"

KV Store



00000001.sst

Lookup #1  
(key to file)

File System



00001024

Lookup #2  
(file to LBA)

SSD



00065536

Lookup #3  
(LBA to PPA)

# Key-Value SSDs vs Block SSDs

- Compaction is later required to remove stale data and reclaim logical space

aaaa bcde  
cdcb cdef  
ergf lfre

~~aaaa~~ ~~cdef~~  
aaab erfg  
cdcb ~~aaaa~~  
aaaa bcde  
lfre cdef



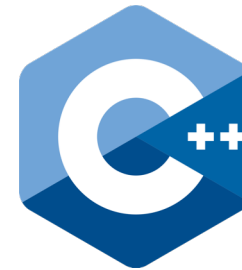
00000005.sst  
00000006.sst

00000001.sst  
00000002.sst  
00000003.sst  
00000004.sst

# Key-Value SSDs vs Block SSDs

- Key-value SSDs use *keys* to access files, not *LBAs*
- Data management operations are handled inside the device
- Less translation overheads, no compaction (or similar data management)

User Program



"userkey001"

SSD



00065536

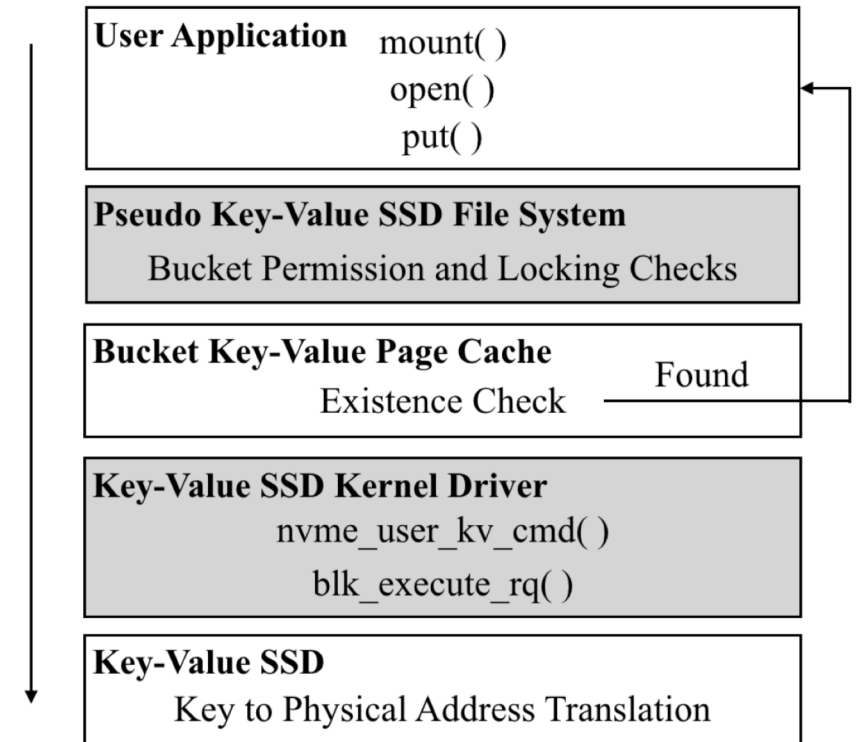
Lookup #1  
(key to PPA)

# Missing KVSSD Support at the OS Level

- Key-value interface affords a leaner I/O stack
- However, bypasses important OS layers
  - Cannot safely use KVSSDs in cloud and multi-user environments
  - No OS level page caching
- Current system calls unsuitable for KVSSD I/O

# Kernel Support for KVSSDs File System and Page Cache

- First proposal : a thin pseudo file system designed for KVSSDs
  - Provide a special file for each unique key space (bucket)
  - open() these files for bucket level permissions and locking
  - Key-value tailored data cache inside the file system
  - Call familiar functions on buckets (ls, cat)



# Kernel Support for KVSSDs

## System Calls

- Second proposal : system calls for KVSSD I/O
  - Current Linux system calls unsuitable for key-value I/O
  - Perform I/O at the bucket level
  - put(), get(), delete(), batch(), iterate()
  - Larger value size support than allowed by device

```
int fd, ret, keylen = 8, vallen = 128;

void* key      = "ABCDEFGH";
void* putval   = (char*)malloc(vallen);
void* getval   = (char*)malloc(vallen);

fd = open("/mnt/kvssd/my_bucket", O_CREATE);

fill_buf(write_val);
ret = kv_put(fd, key, keylen, putval,
             vallen, NULL);
ret = kv_get(fd, key, keylen, getval,
             vallen, NULL);
```

[cduffy@snu.ac.kr](mailto:cduffy@snu.ac.kr)

[sanghoonkim@ajou.ac.kr](mailto:sanghoonkim@ajou.ac.kr)

[jinsoo.kim@snu.ac.kr](mailto:jinsoo.kim@snu.ac.kr)

[csl.snu.ac.kr](http://csl.snu.ac.kr)

[sslaboratory.ajou.ac.kr](http://sslaboratory.ajou.ac.kr)

# Thank You

