KV Userspace Driver
(OpenMPDK-uNVMe)

Kyungsan Kim / SW Dev Team / Memory Business Unit

Today's topic

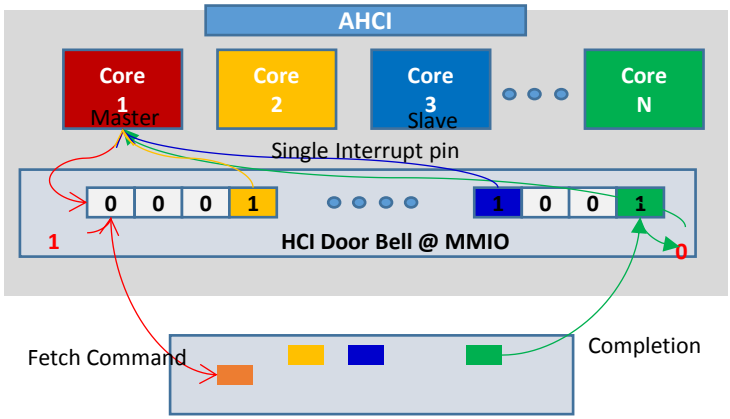
KeyValue NVMe?

Userspace Driver?

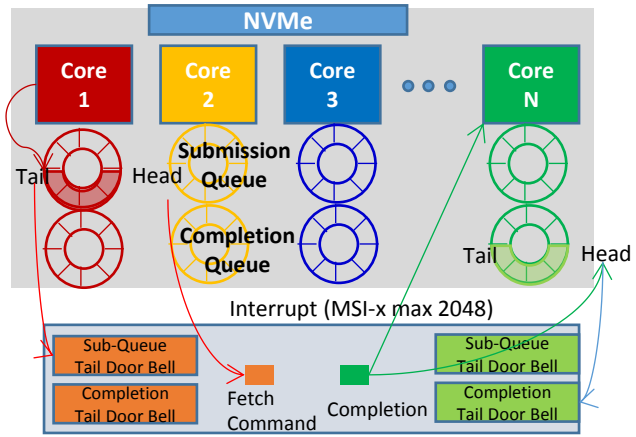
NVMe – High performance storage bus interface

- Scalable storage bus interface over PCI-e controller, emerged in the era of multi core and higher performance SSD

- **SATA(ACHI): Master/Slave architecture**
  - ✓ IO Submission : performance limitation when slave core issue commands
  - ✓ IO Completion : performance limitation due to Broadcasting interrupt from master to slave core



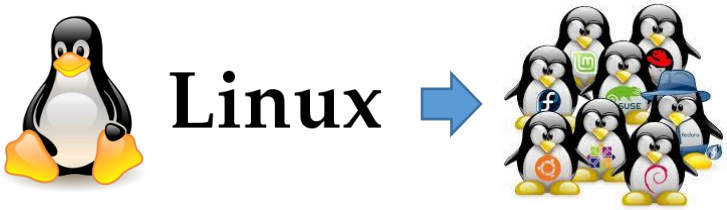
- **NVMe: Parallelism**
  - ✓ Per CPU Queuing : minimum lock contention / maximum cache hit ratio
  - ✓ Paired Queuing : IO Submission and Completion Q is separated
  - ✓ Interrupt balancing with MSI-X : maximum 2048 interrupts can be balanced on all cores



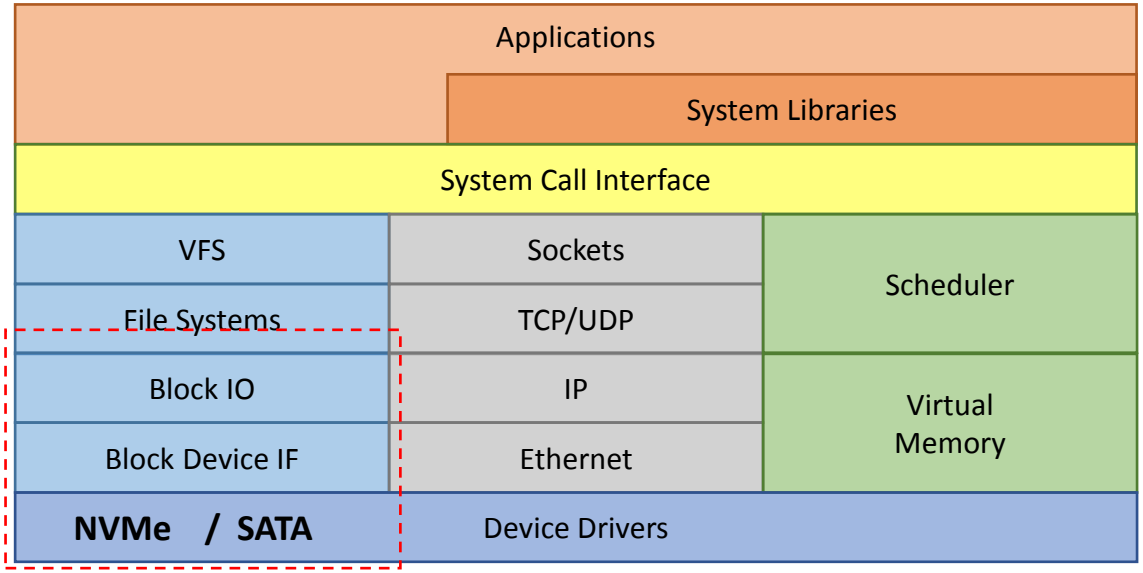
	UFS (AHCI)	SATA (AHCI)	NVMe
Queue : CPU mapping	1:N	1:N	1:1 (per cpu)
Maximum Queue Count	1	1	64K
Maximum Queue Depth	32	32	Max 64K (host/ssd negotiation)
Maximum Interrupt handler	1	1	2048(MSI-X)

Kernel and Userspace Driver

- POSIX OS consists of kernel sub components such as VM, scheduler, network, file system, and block layer
- Kernel device driver for NVMe storage located beneath block layer



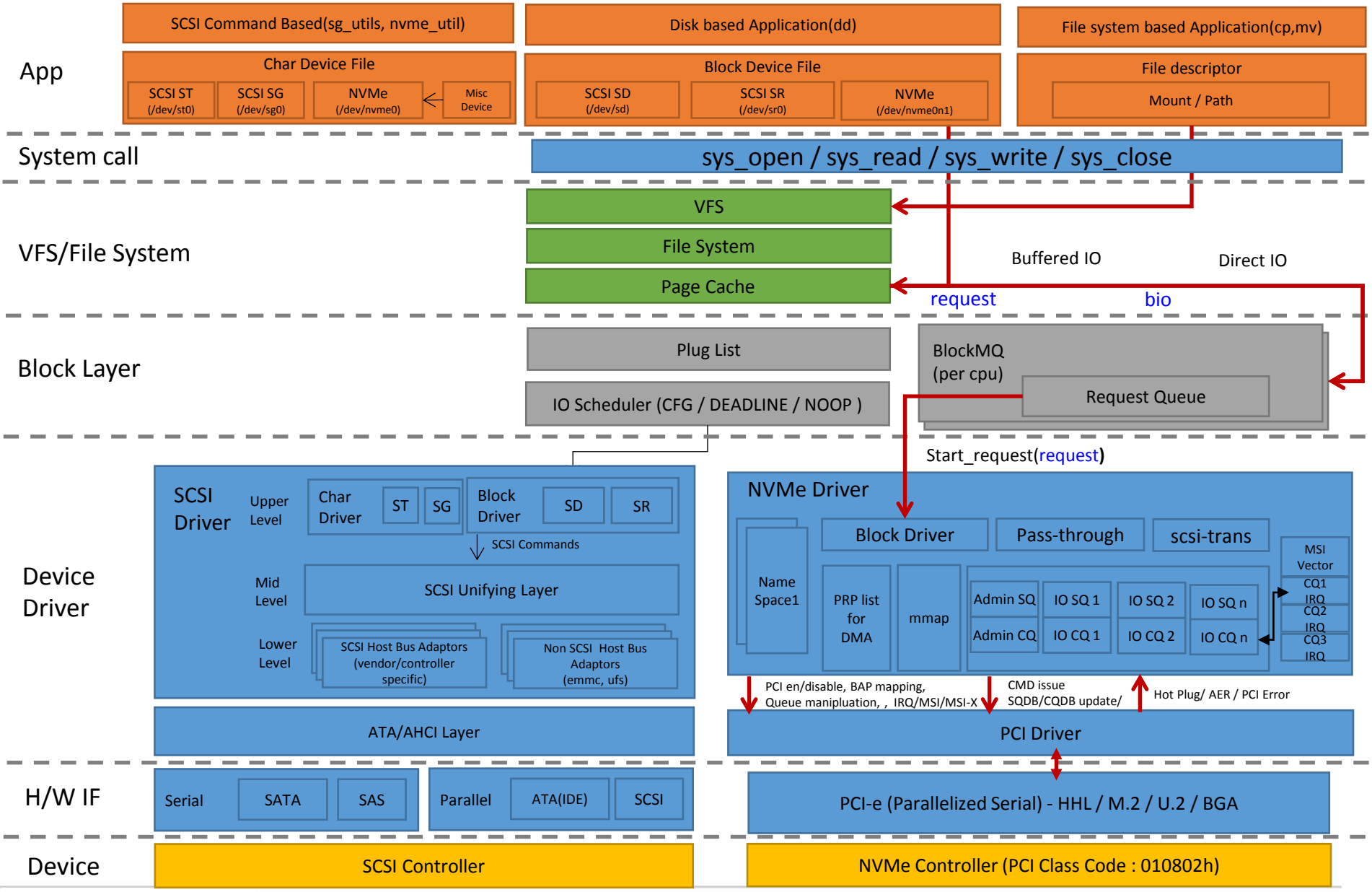
various distros…
redhat, ubuntu, debian, centos, fedora, suse, and so on



NVMe SSD

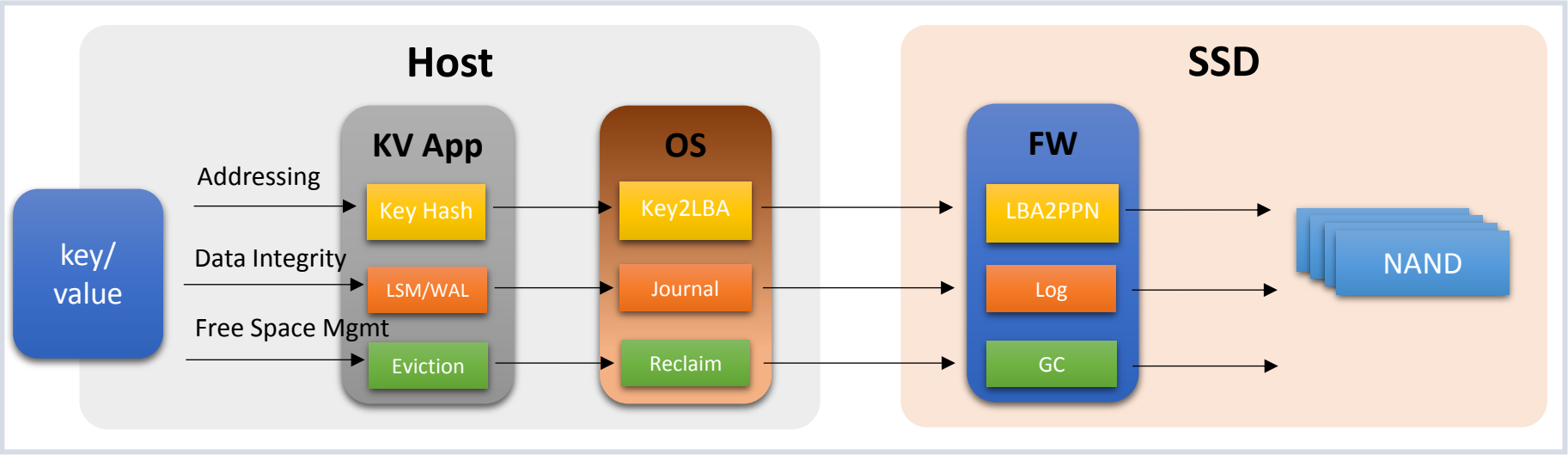
SATA SSD

Kernel Driver - NVMe IO Path



KV User Driver Background - KV SSD Premise

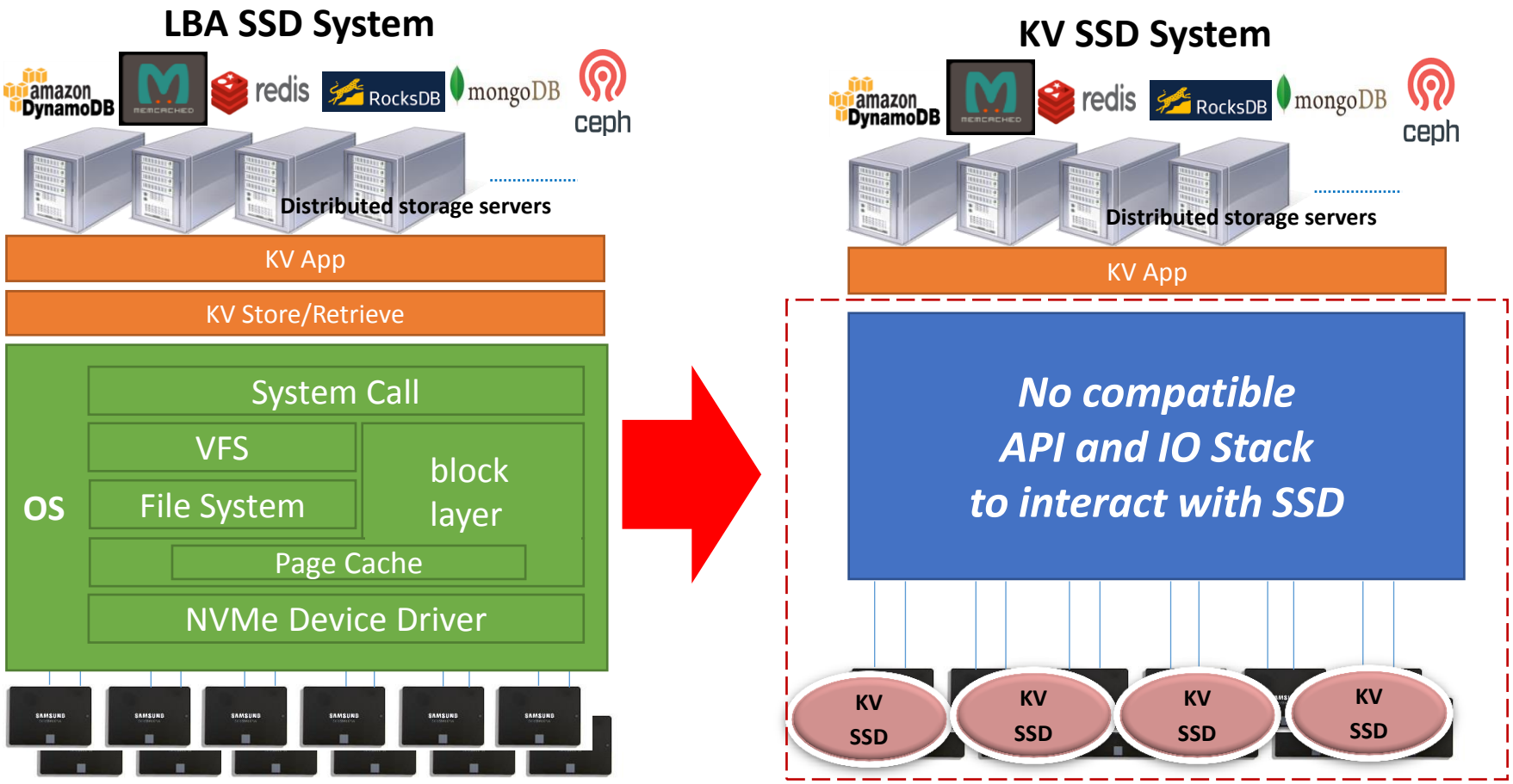
- KV SSD benefits are originated from eliminating functional **redundency** ranging from host application to SSD internal
  - **Addressing , Data Integrity, and Free Space Management**
- Especially, data integrity redundency causes increased WAF/RAF, which makes SSD longevity worse and app IO latency longer, consuming more host cpu/memory resources



- KV SSD introduces new NVMe commands such as iterate and interfacing protocol to deal with dynamic size key/value pair via host driver

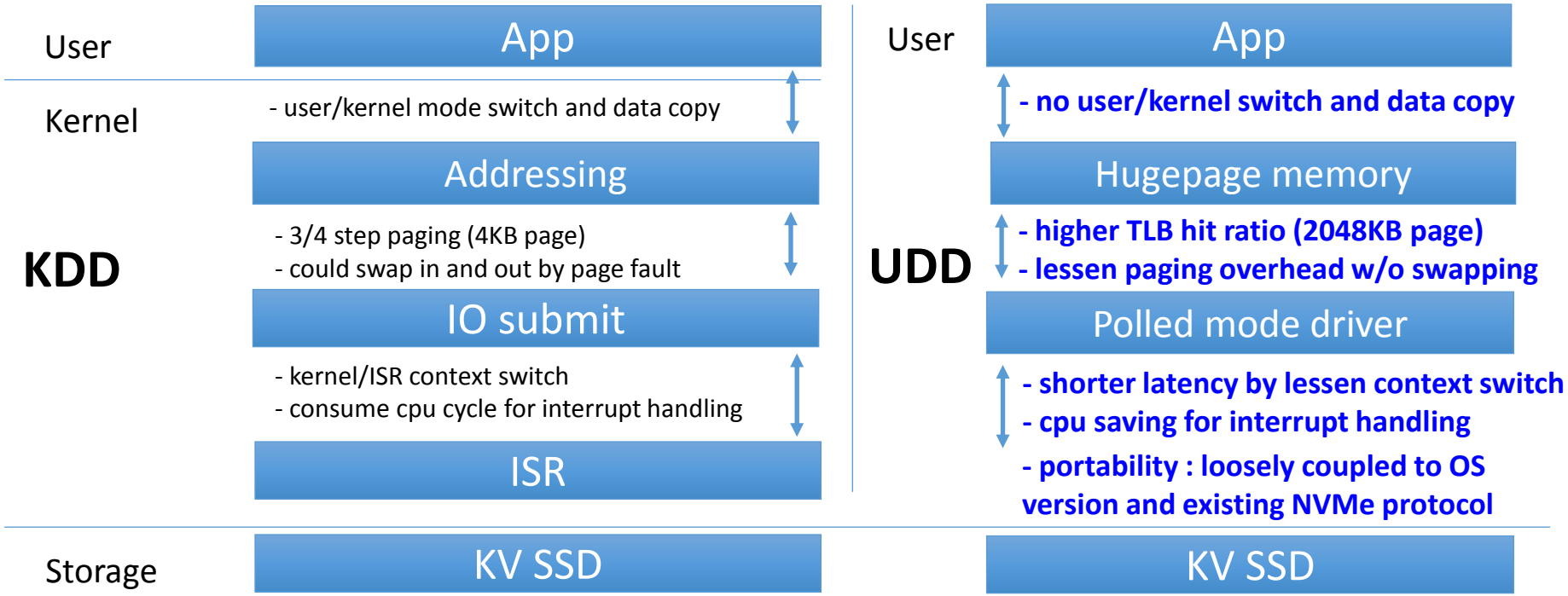
KV User Driver Background - KV SSD System

- Accordingly, KV SSD system is designed to bypass typical operating system storage stack as many as possible. Also, the modified storage protocol requires changes in device driver



KV Userspace Driver: OpenMPDK-uNVMe

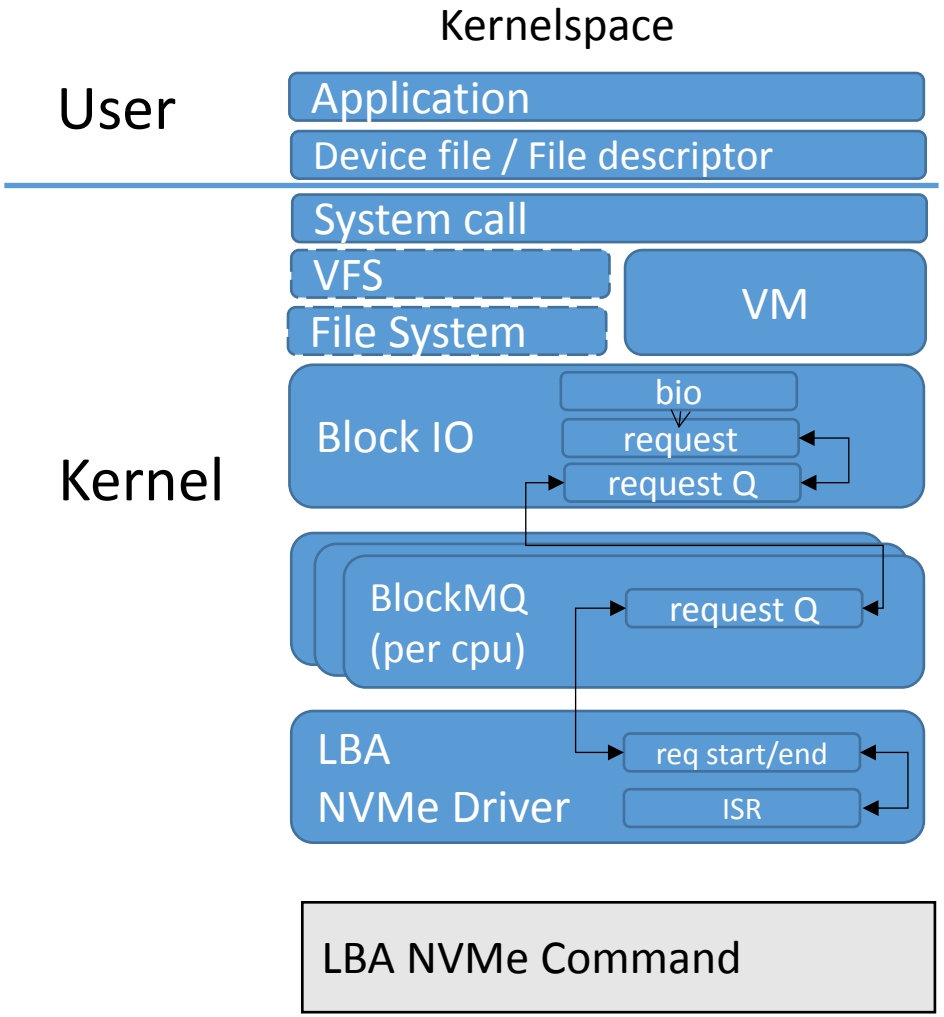
- **Why Userspace driver?** UDD can provides an optimized IO stack with smaller OS overhead and dependency

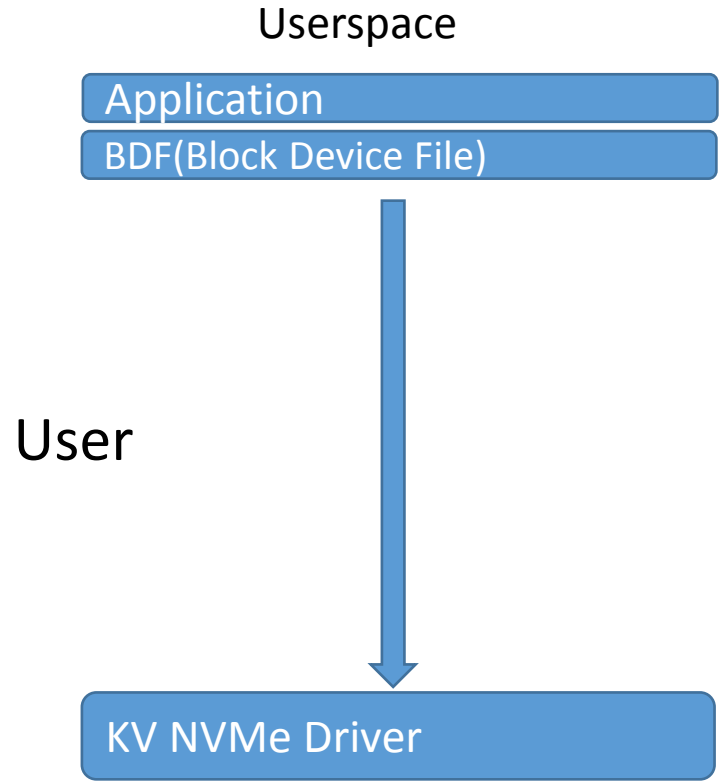


➔ For the reasons, uNVMe has been implementing as extension of *Intel SPDK.* uNVMe further provides means to optimize host cpu/dram usage and cope with diverse IO characteristics

- Apparently, KDD is worthwhile in that it covers various IO pattern out of real world application over commercial systems and a fundamental approach to widespread KV SSD echo system. Thus, Samsung provides both KDD and UDD

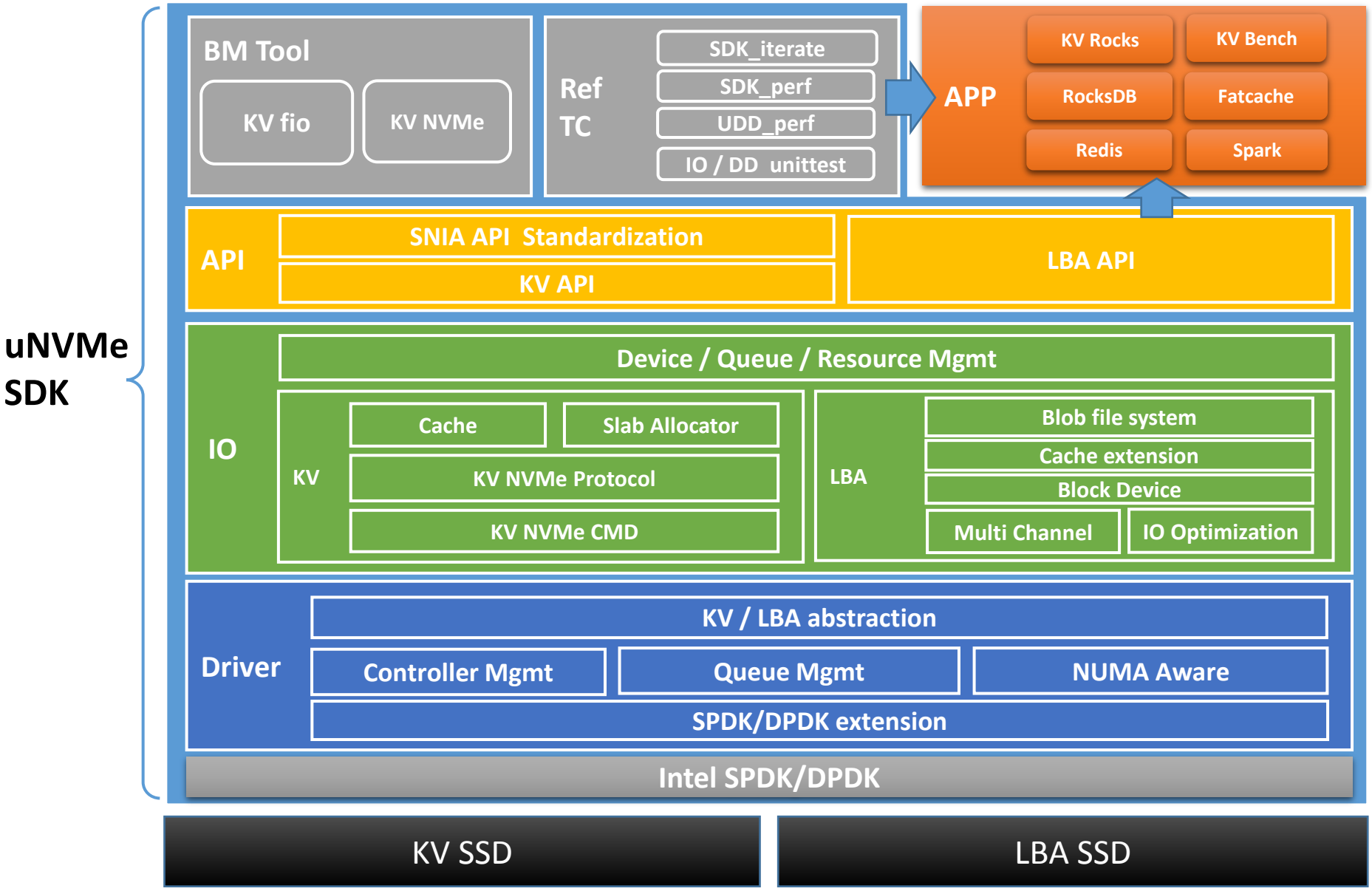
Kernel / Userspace IO Flow





KV NVMe Command			
Key size	Value size	Key space	Command
4~255B	0~2048KB	Multiple Keyspace	Store (Put)
			Retrieve (Get)
			Delete
			Iterate Request
			Iterate Read
			Exist
			Utilization / WAF / Iterate-info / etc

uNVMe SDK – Oct 2018



Programming Syntax Change(1)

IO on LBA SSD

Application

fd = open(“path,,)

KV to Buffer

write/read(fd, buffer)

Buffer to KV

close(fd)

IO on KV SSD

Application

write/read(KV)

Programming Syntax Change(2)

- **uNVMe primary API and data structures below**
- **Please be aware that SNIA API standardized is also provided**

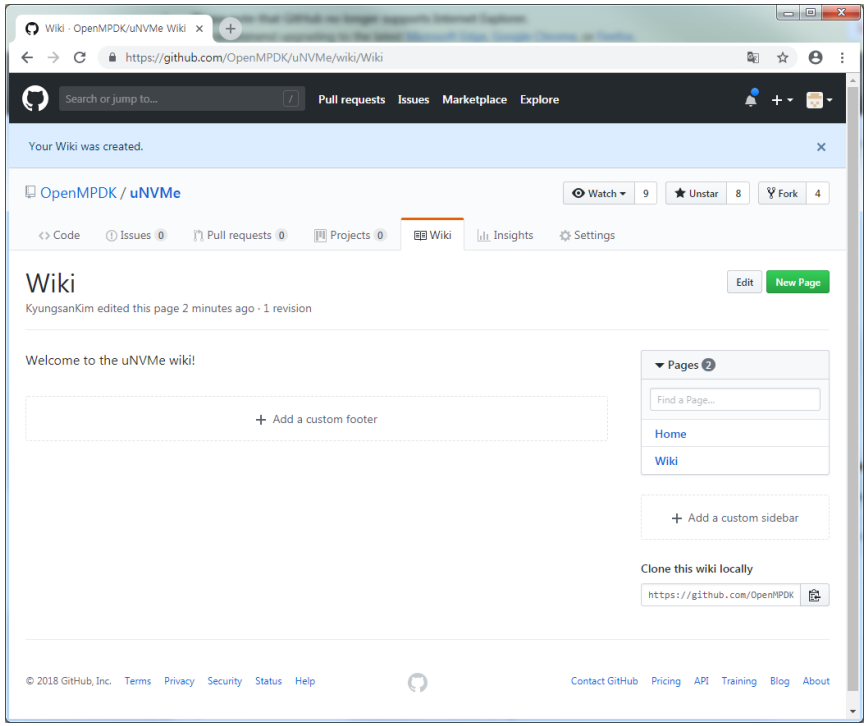
```
/** SDK init / finalize **/
int kv_sdk_init(int init_from, void *option);
int kv_sdk_load_option(kv_sdk *sdk_opt, char* log_path);
int kv_sdk_finalize(void);
void kv_sdk_info(void);


/** KV SSD IO **/
int kv_store(uint64_t handle, kv_pair *kv);
int kv_store_async(uint64_t handle, kv_pair *kv);
int kv_retrieve(uint64_t handle, kv_pair *kv);
int kv_retrieve_async(uint64_t handle, kv_pair*kv);
int kv_delete(uint64_t handle, kv_pair *kv);
int kv_delete_async(uint64_t handle, kv_pair *kv);
uint32_t kv_iterate_open(uint64_t handle, const uint8_t keyspace_id, const
uint32_t bitmask, const uint32_t prefix, const uint8_t iterate_type);
int kv_iterate_close(uint64_t handle, const uint8_t iterator);
int kv_iterate_read(uint64_t handle, kv_iterate* it);
int kv_iterate_read_async(uint64_t handle, kv_iterate* it);
int kv_iterate_info(uint64_t handle, kv_iterate_handle_info* info, int nr_handle);
int kv_exist(uint64_t handle, kv_pair* kv);
int kv_exist_async(uint64_t handle, kv_pair* kv);
int kv_format_device(uint64_t handle, int erase_user_data);


/** Utilities **/
uint64_t kv_get_total_size(uint64_t handle);
uint64_t kv_get_used_size(uint64_t handle);
uint64_t kv_get_waf(uint64_t handle);
uint32_t kv_get_sector_size(uint64_t handle);
int kv_get_log_page(uint64_t handle, uint8_t log_id, void* buffer, uint32_t
buffer_size);
```

```
/**  * @brief A pair of structures of key, value, and kv_param. */
typedef struct {
    uint8_t keyspace_id;
    kv_key key;
    kv_value value;
    kv_param param;
} kv_pair;

typedef struct {
    void *key;              /**< a pointer to a key */
    uint16_t length;        /**< key length in byte unit */
} kv_key;

/**  * @brief A value consists of a buffer pointer and its length  */
typedef struct {
    void *value;        /**< buffer address for value */
    uint32_t length;    /**< value buffer size in byte unit for input and the retuned value
length for output*/
    uint32_t actual_value_size;     /* full value size that is stored in disk (only applied on KV
SSD, not on LBA SSD) */
    uint32_t offset;        /**< offset for value */
} kv_value;

/** * @brief A structure which contains I/O option, and callback function(for async I/O)  */
typedef struct {
    void (*async_cb)();         /**< async notification callback (valid only for async I/O) */
    void* private_data;         /**< private data address used in callback (valid only for async
I/O) */
    union {
        int store_option;
        int retrieve_option;
        int delete_option;
        int iterate_request_option;
        int iterate_read_option;
        int exist_option;
    }io_option;             /**< options for operations */
} kv_param;
```

Extra materials

Guide Document

- uNVMe/doc/uNVMe2.0_SDK_Programmung_Guide_v1.1.pdf
- uNVMe/doc/uNVMe2.0_SDK_Evaluation_Guide_v1.1.pdf



https://github.com/OpenMPDK/uNVMe

Features Business Explore Marketplace Pricing Search / Sign in or Sign up

Open Memory Platform Development Kit

Repositories 3 People 0 Projects 0

Grow your team on GitHub

GitHub is home to over 28 million developers working together. Join them to grow your own development teams, manage permissions, and collaborate on projects.

Sign up

Dismiss

Find a repository... Type: All Language: All

KVSSD

KV SSD host software package

C 4 5 Updated 10 days ago

uNVMe

C++ 8 4 Updated on 6 Jul

HPBDriver

HPB (Host-aware Performance Booster)

2 Updated on 5 Jul

Top languages

C++ C

People 0

This organization has no public members. You must be a member to see who's a part of this organization.

THE NEXT CREATION STARTS HERE

Placing **memory** at the forefront of future innovation and creative IT life