커널 변경 사항

LG전자 김준수
js1304@gmail.com

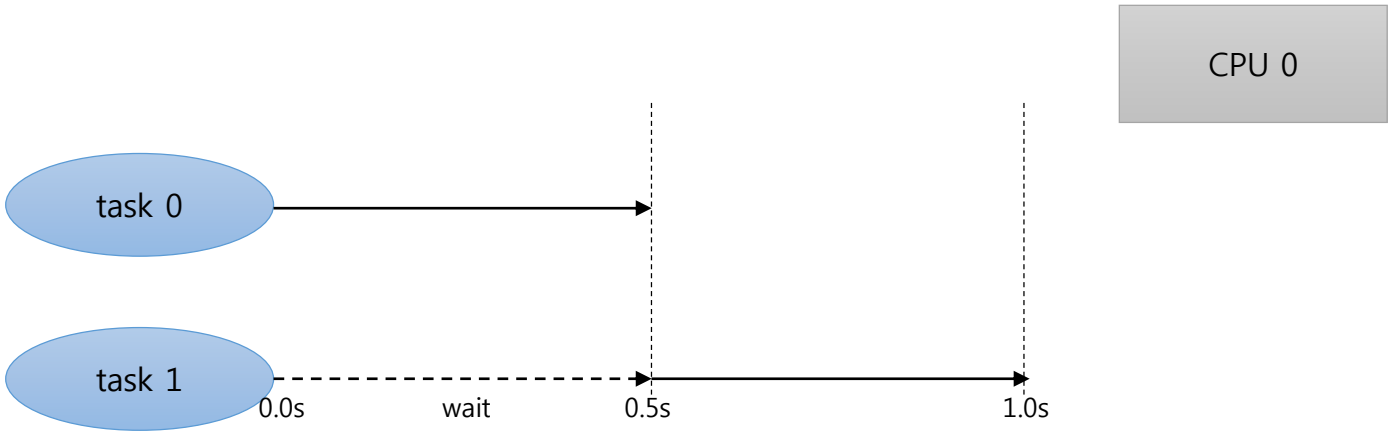
목차

- v4.20
  - Pressure Stall Information
  - TCP: switch to Early Departure Time model
  - Revisit CPU security bugs
- v5.0
  - Energy-aware scheduling
  - KASAN: software tag based mode
  - ZRAM improvements
- v5.1
  - High performance asynchronous I/O
  - Use persistent memory as RAM
  - TEO, an alternative CPU idle governor

v4.20

Pressure Stall Information (PSI)

- 시스템 자원이 경쟁 상태에 있음을 나타내는 새로운 성능지표
- 경쟁 상태로 인해 지연된 작업이 존재하는 시간 비율



- 출력 예시

```
some avg10=2.04 avg60=0.75 avg300=0.40 total=157656722
```

Pressure Stall Information (PSI)

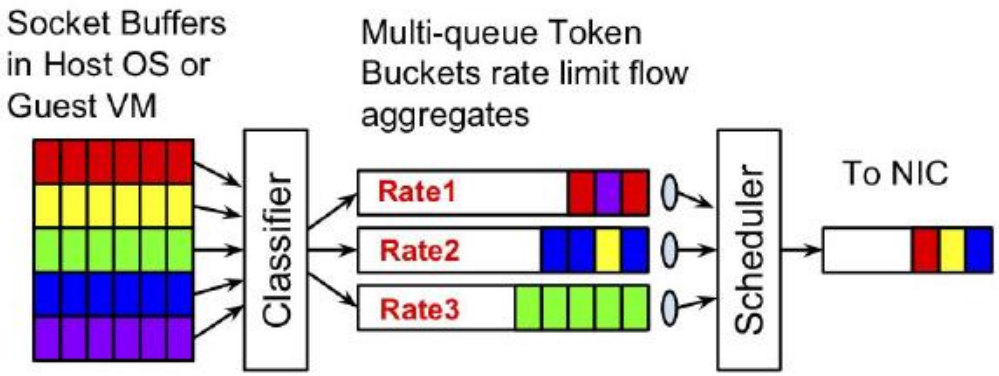
- workload 와 CPU 로 생산성을 측정
  - SOME state: nr_delayed_tasks != 0
  - FULL state: nr_delayed_task != 0 && nr_running_tasks == 0

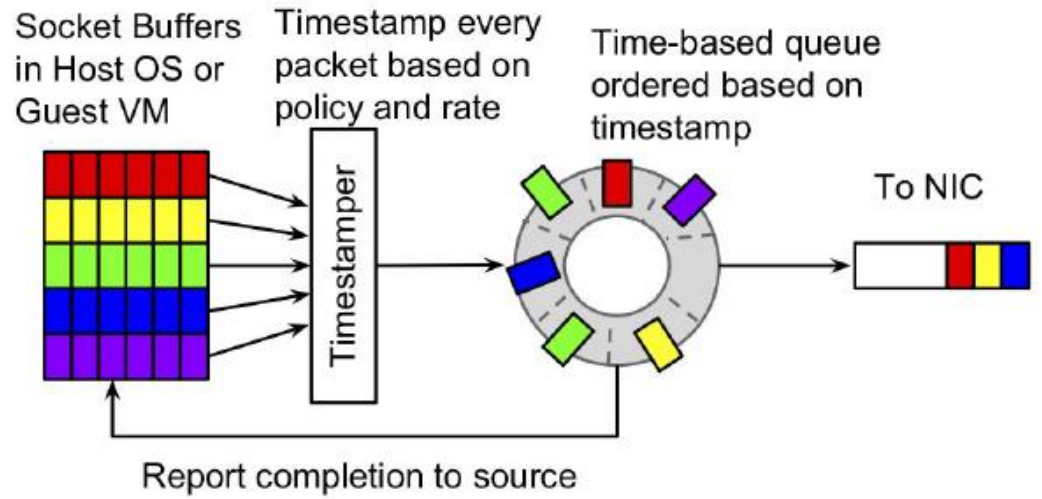
- 측정 하는 자원
  - CPU, MEMORY, IO

- Future work
  - Pressure stall monitor

TCP: switch to Early Departure Time model

- 네트워크 트래픽을 조절하는 새로운 모델

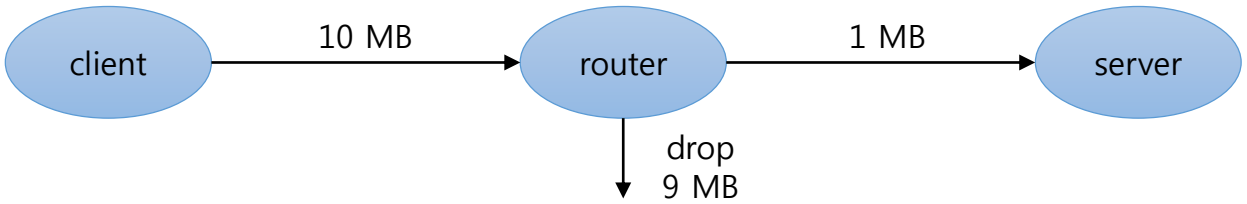




출처: www.netdevconf.org

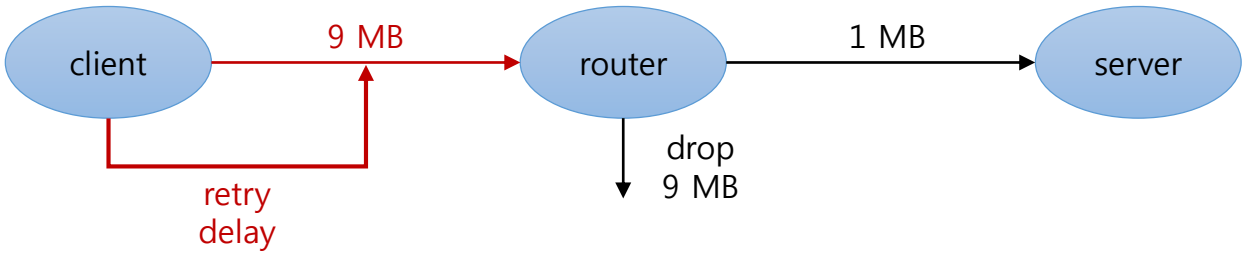
TCP: switch to Early Departure Time model

- 네트워크 속도의 한계
- AFAP (as fast as possible)
- AFAN (as fast as necessary)



TCP: switch to Early Departure Time model

- 네트워크 속도의 한계
- AFAP (as fast as possible)
- AFAN (as fast as necessary)



TCP: switch to Early Departure Time model

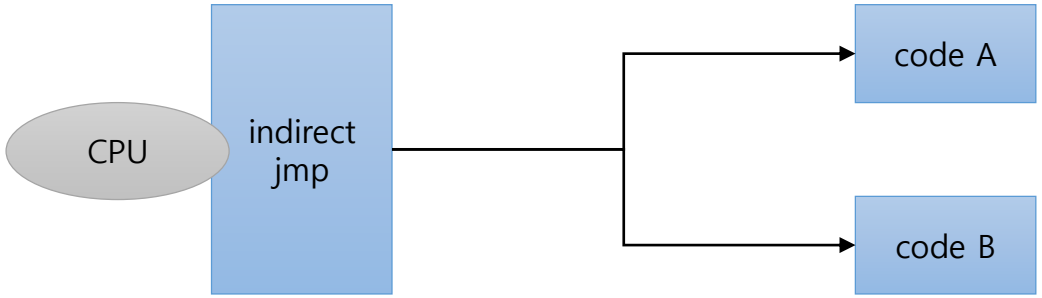
- 기존의 queue 구현에 비해 빠름
- queue 를 완전히 (backward compatible) 대체 가능
- NIC API
  - send (what) → send (what, when)

Revisit CPU security bugs

- Spectre v2

  - Spectre(v2)?
    - 인접한 cpu의 indirect 분기를 일시적으로 제어할 수 있는 하드웨어 버그(?)를 이용해서 해커가 원하는 코드(미리 읽기 코드)로 분기하게 하여 미리 읽기를 유도한 후 캐시를 분석하여 미리 읽은 값을 찾아내는 해킹 방법

출처: kernel-dev-ko.github.io

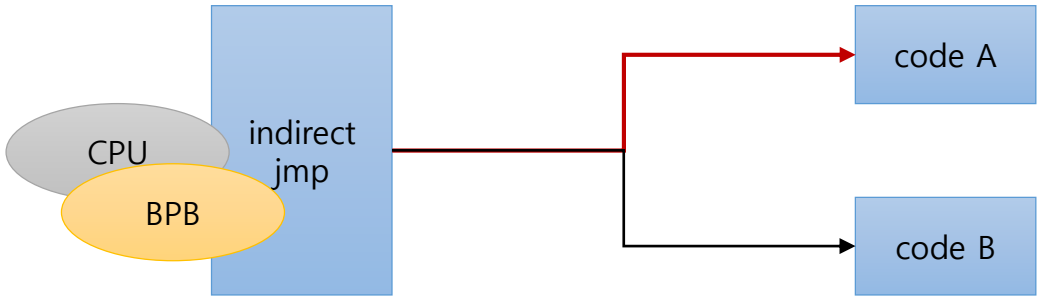


Revisit CPU security bugs

- Spectre v2

  - Spectre(v2)?
    - 인접한 cpu의 indirect 분기를 일시적으로 제어할 수 있는 하드웨어 버그(?)를 이용해서 해커가 원하는 코드(미리 읽기 코드)로 분기하게 하여 미리 읽기를 유도한 후 캐시를 분석하여 미리 읽은 값을 찾아내는 해킹 방법

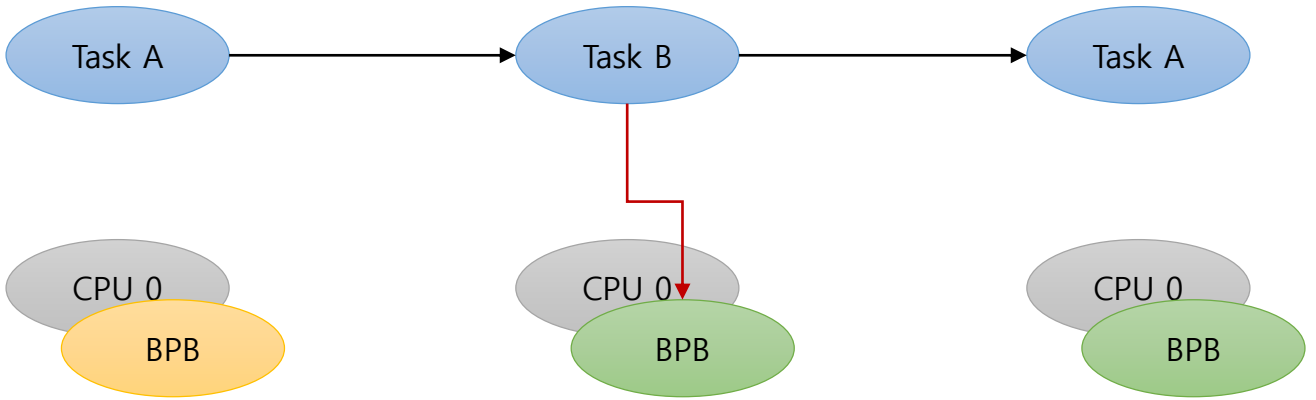
출처: kernel-dev-ko.github.io



Revisit CPU security bugs

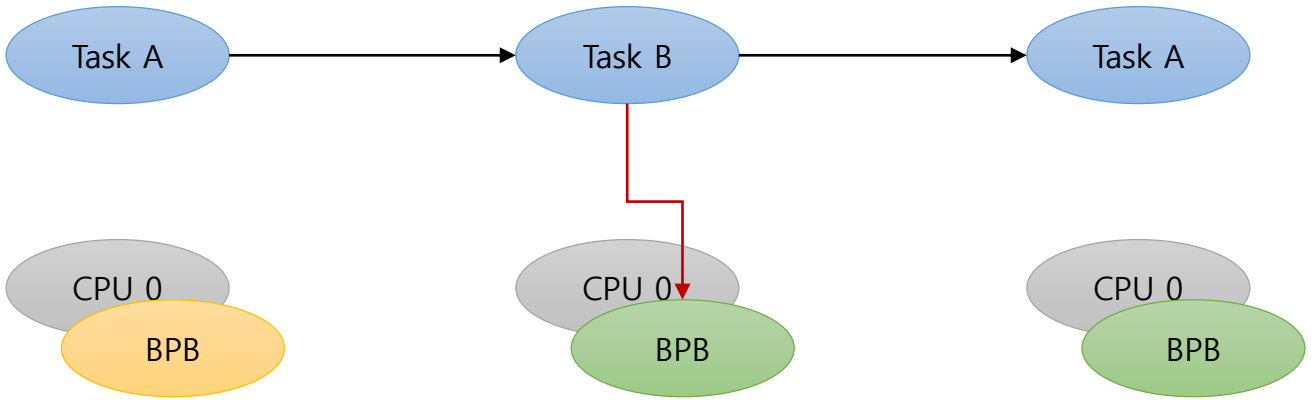
- retpoline
  - jmp → ret

- user-space to user-space attack



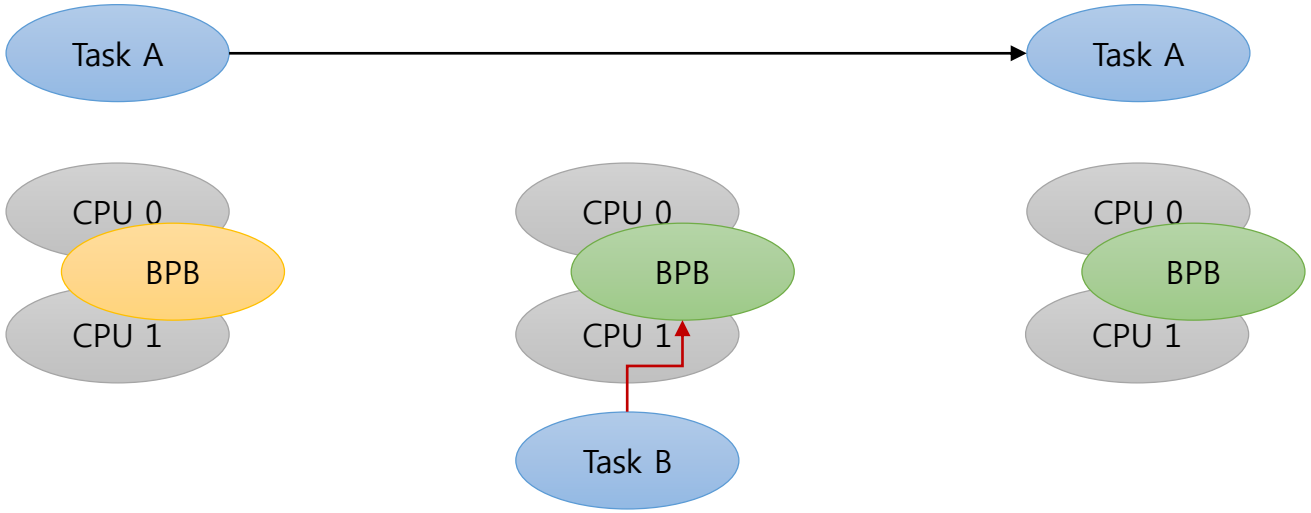
Revisit CPU security bugs

- IBPB (indirect branch prediction barrier)



Revisit CPU security bugs

- IBPB (indirect branch prediction barrier)
- STIBP (single thread indirect branch predictors)



v5.0

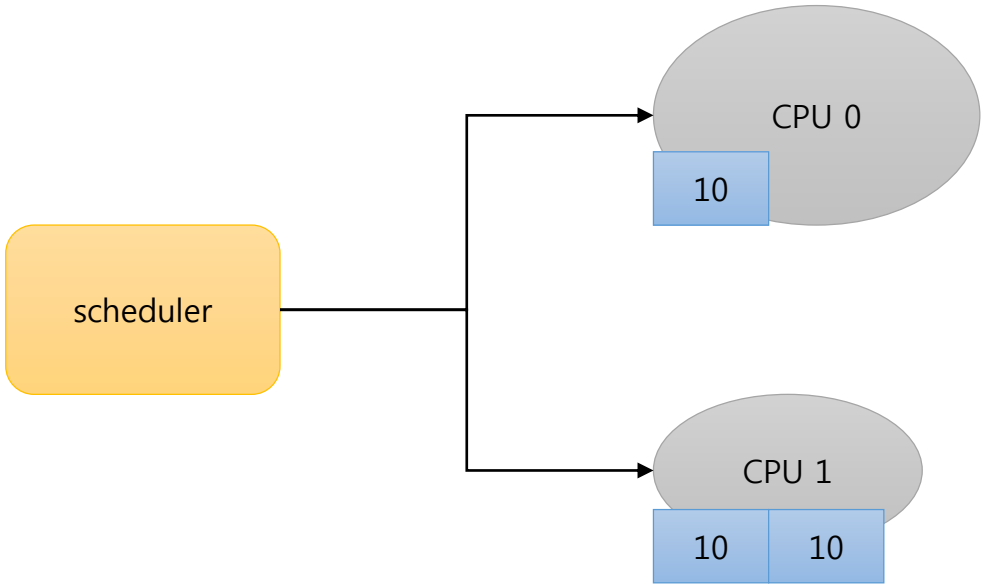
Energy-aware scheduling

- 에너지 소모량을 최소화 하기 위한 스케쥴링 알고리즘

- ARM big.LITTLE 을 위한 스케쥴링 알고리즘

```
struct capacity_state {
        unsigned long cap;          /* compute capacity */
        unsigned long power;        /* power consumption at this compute capacity */
};

struct sched_energy_model {
        int nr_cap_states;
        struct capacity_state *cap_states;
}
```

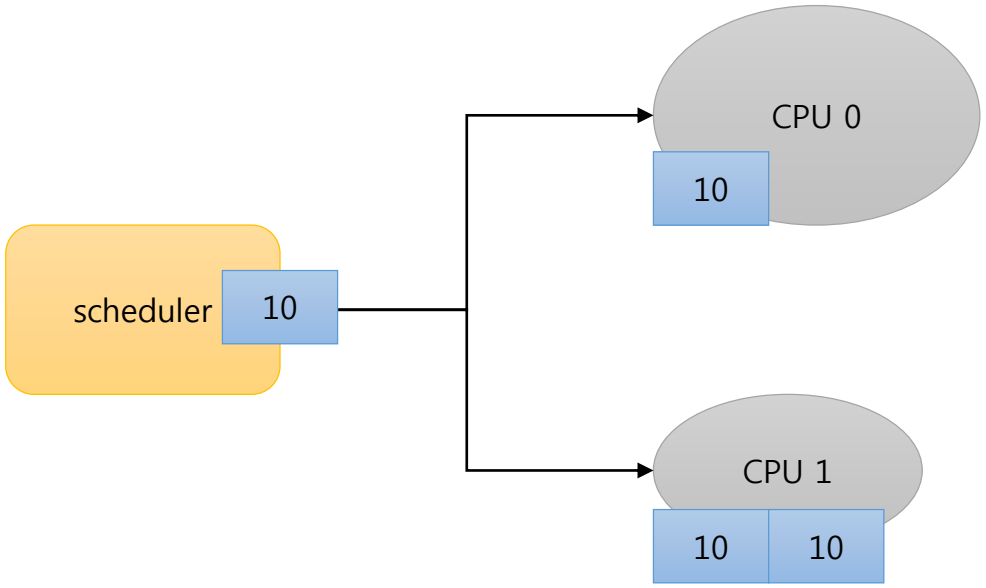
Energy-aware scheduling



State	Cap	Pow
0	10	10
1	50	50
…	…	…

State	Cap	Pow
0	10	10
1	20	20
2	30	30
…	…	…

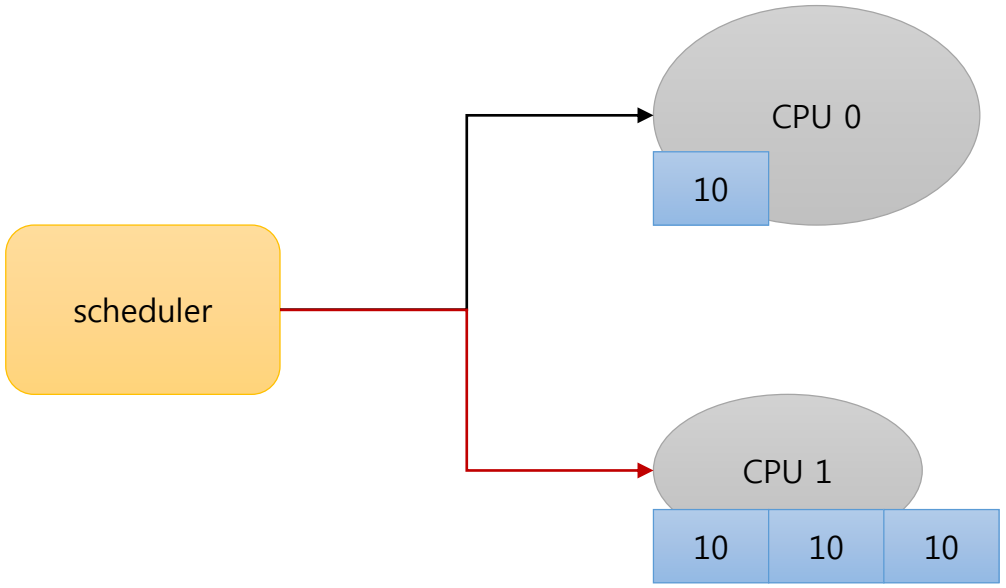
Energy-aware scheduling



State	Cap	Pow
0	10	10
1	50	50
…	…	…

State	Cap	Pow
0	10	10
1	20	20
2	30	30
…	…	…

Energy-aware scheduling



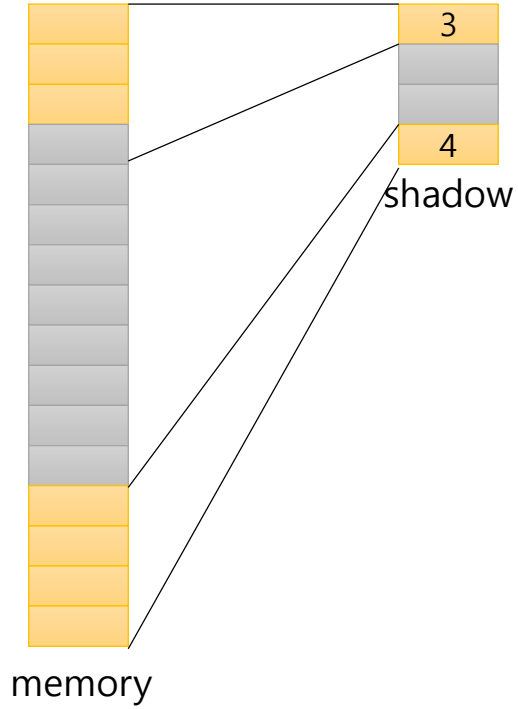
State	Cap	Pow
0	10	10
1	50	50
…	…	…

State	Cap	Pow
0	10	10
1	20	20
2	30	30
…	…	…

KASAN: software tag-based mode

- Compiler instrumentation
- Shadow memory
  - 할당 상태에 대한 정보

```
int a[5];
int b;


----------------→ hook
b= a[5]
```

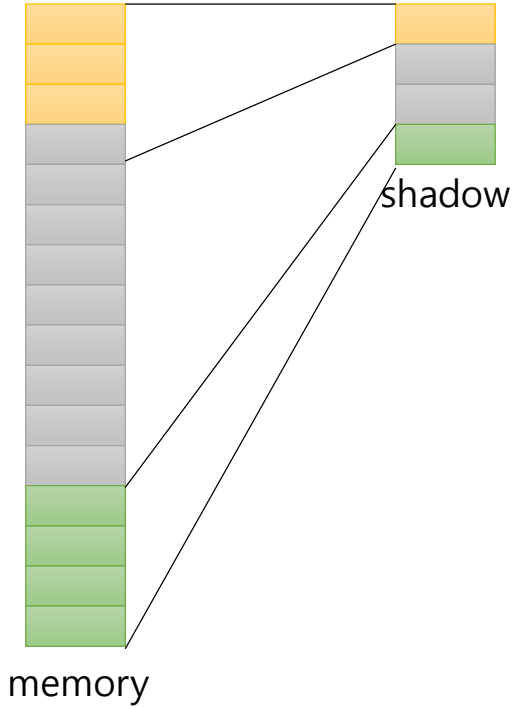


KASAN: software tag-based mode

- Compiler instrumentation

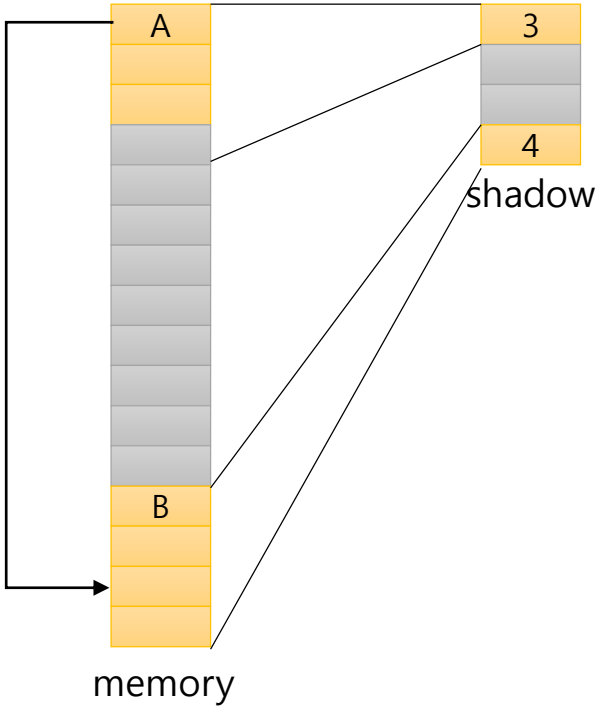
```
int a[5];
int b;


----------------→ hook
b= a[5]
```

- Shadow memory
  - 할당 별로 tag 를 저장

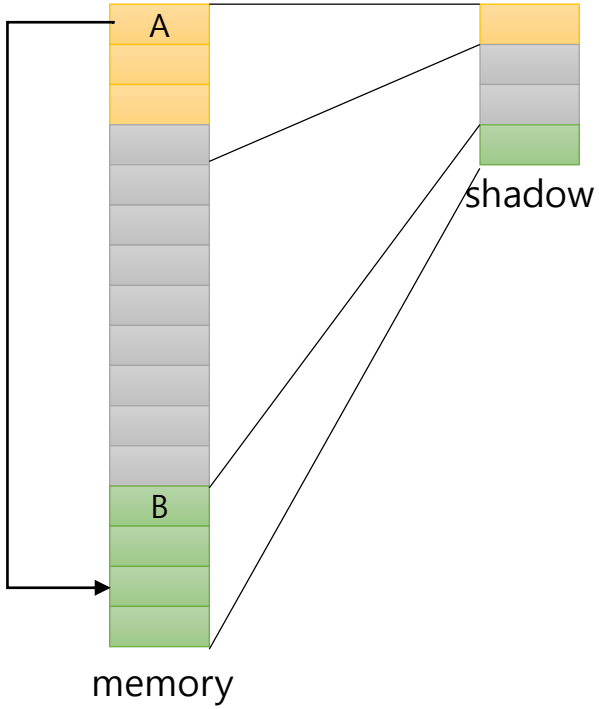


KASAN: software tag-based mode

- x = A[14]



- x = A[14]



KASAN: software tag-based mode

- 동작 방법
  - ARM64 에서 지원하는 Top Byte Ignore (TBI) 기능을 이용
  - 커널 메모리(SLAB object) 할당 성공시, 반환되는 주소 top byte 에 tagging
  - shadow memory 에 tag 를 저장
  - Compiler instrumentation 으로 memory read/write instruction 앞에 추가된 hook 을 이용
  - Hook 에서 shadow memory 의 tag 와 메모리 접근에 사용된 pointer 의 tag 를 비교
  - Tag 값이 다르면, 올바르지 않은 접근

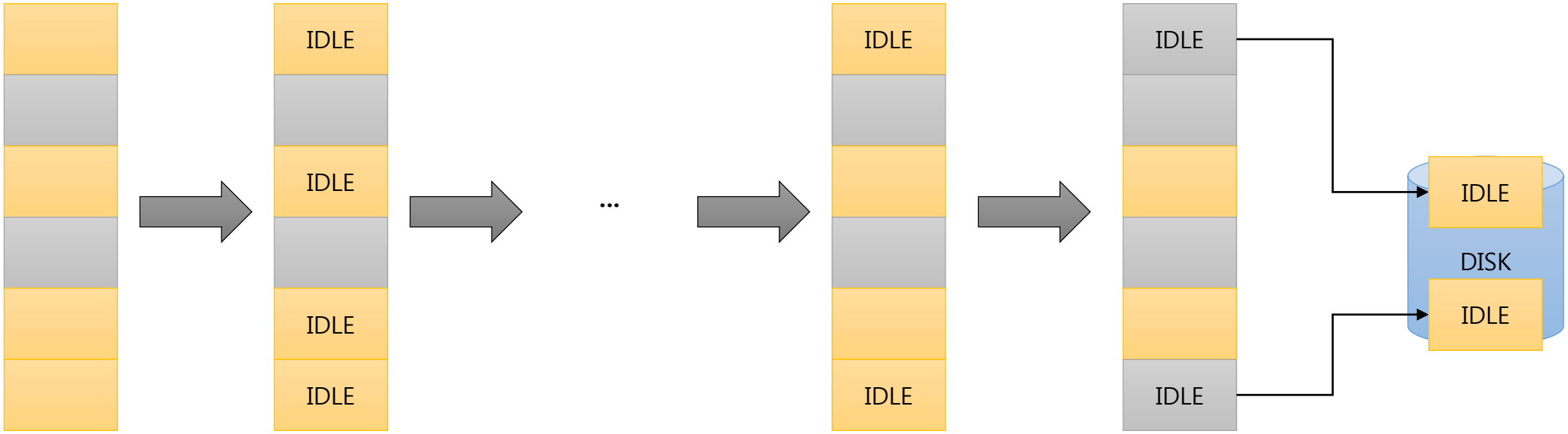
KASAN: software tag-based mode

- 장점
  - 메모리 절약
    - use-after-free 를 검출하기 위해 quarantine 기능 불필요
  - memory overflow 로 다른 할당된 메모리에 접근하는 경우에도 검출

- 단점
  - CPU specific feature
  - 현재 gcc 는 지원하지 않음. Clang 7.0 이상이 요구됨
  - global/stack variable 은 지원하지 않음

ZRAM improvements

- 요약
  - zram 에 존재하는 특정 타입의 page 들을 disk 로 옮길 수 있는 기능
  - 메모리 절약 가능



ZRAM improvements

- 기반 설비
  - IDLE mark 를 위한 interface 제공
  - IDLE/HUGE page 를 선택적으로 writeback 할 수 있는 interface 제공
  - writeback throttling 기능 구현

```
while (1) {
        # mark allocated zram slot to IDLE
        echo all > /sys/block/zram0/idle

        # leave system working for several hours unless there is no access for some blocks on zram
        # they are still IDLE marked pages

        # write the IDLE marked slot into backing device and free the memory
        echo "idle" > /sys/block/zram0/writeback
}
```

v5.1

High-performance asynchronous I/O

- 리눅스에서 asynchronous I/O 를 사용하기 위한 새로운 API
- 이전에 존재 했던 여러 가지 문제점 해결
- User-space library 제공
  - liburing

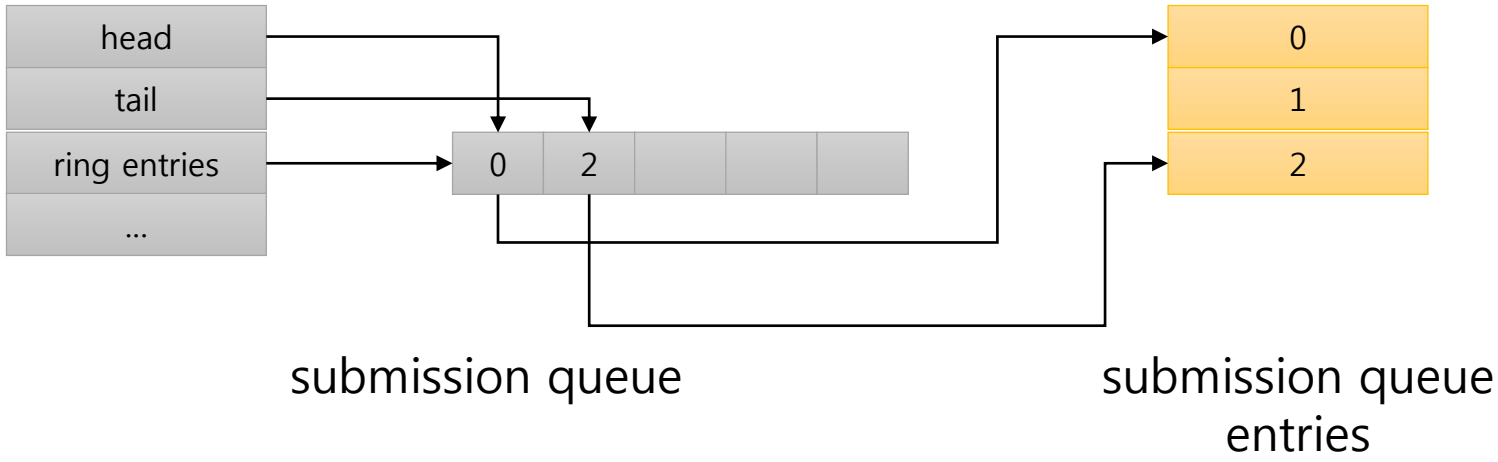
High-performance asynchronous I/O

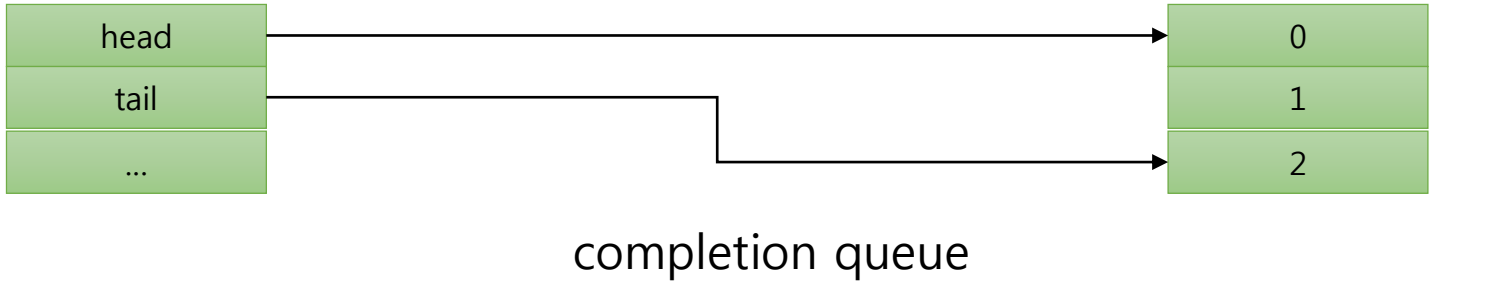
- 600 line 이 넘는 sample code

```
static int setup_ring(struct submitter *s)
{
                struct io_uring_params p;
                ...
                fd = io_uring_setup(depth, &p);
                s->ring_fd = fd;
                ptr = mmap(0, p.sq_off.array + p.sq_entries * sizeof(__u32),
                                PROT_READ | PROT_WRITE, MAP_SHARED | MAP_POPULATE, fd,
                                IORING_OFF_SQ_RING);
                ...
                s->sqes = mmap(0, p.sq_entries * sizeof(struct io_uring_sqe),
                                PROT_READ | PROT_WRITE, MAP_SHARED | MAP_POPULATE, fd,
                                IORING_OFF_SQES);

                ptr = mmap(0, p.cq_off.cqes + p.cq_entries * sizeof(struct io_uring_cqe),
                                PROT_READ | PROT_WRITE, MAP_SHARED | MAP_POPULATE, fd,
                                IORING_OFF_CQ_RING);
                ...
}
```

출처: git.kernel.dk/cgit/fio/plain/t/io_uring.c

High-performance asynchronous I/O





High-performance asynchronous I/O

- int io_uring_setup(int entries, struct io_uring_params *params);

- 세 개의 공간에 대한 mmap
  - submission queue
  - submission queue entries
  - complete queue

```
struct io_uring_params {
        __u32 sq_entries;
        __u32 cq_entries;
        __u32 flags;
        __u16 resv[10];
        struct io_sqring_offsets sq_off;
        struct io_cqring_offsets cq_off;
};
```

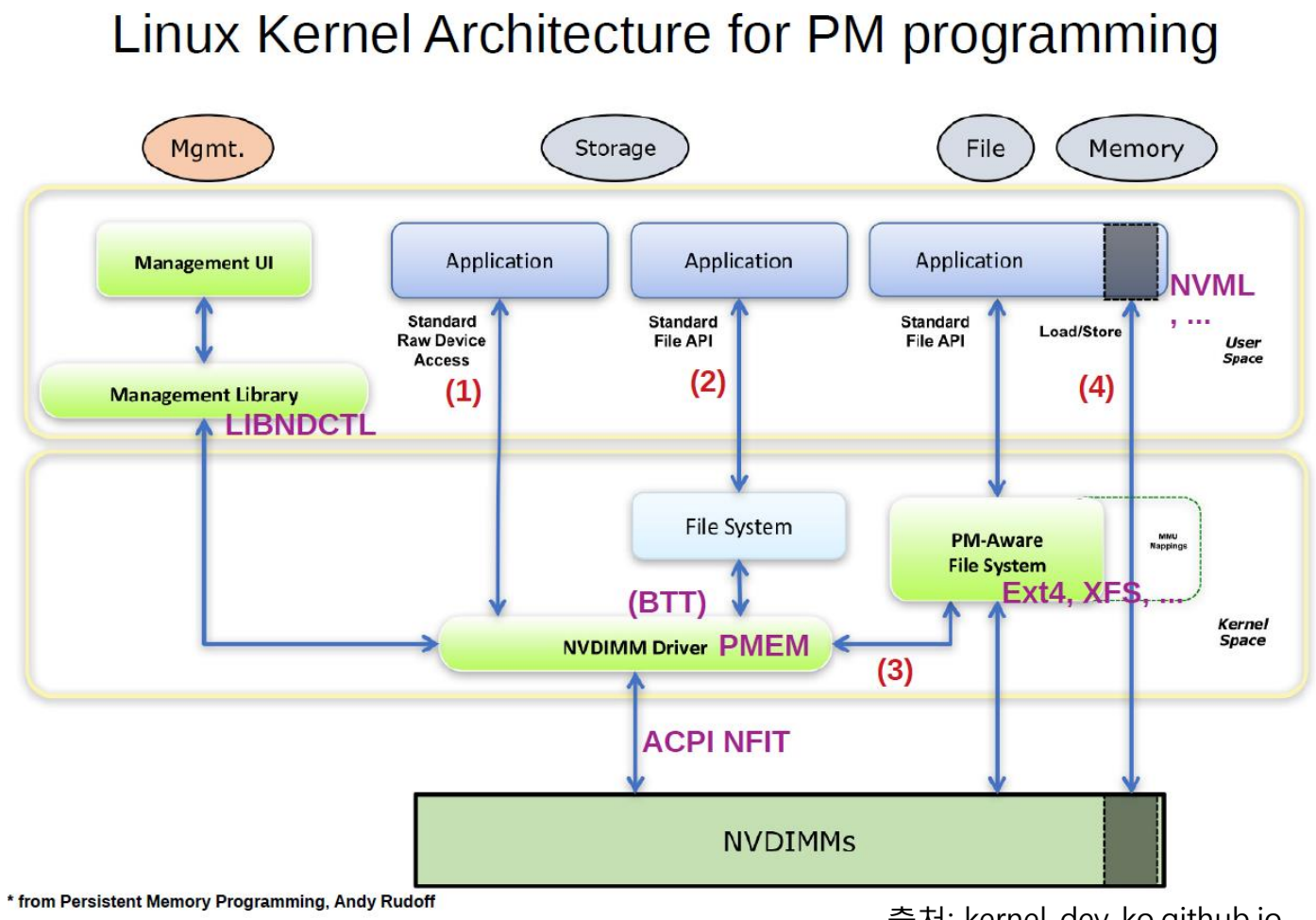
- int io_uring_enter(unsigned int fd, u32 to_submit, u32 min_complete, u32 flags);

Use persistent memory as RAM

- persistent memory 를 memory 로 사용하는 손쉬운 방법

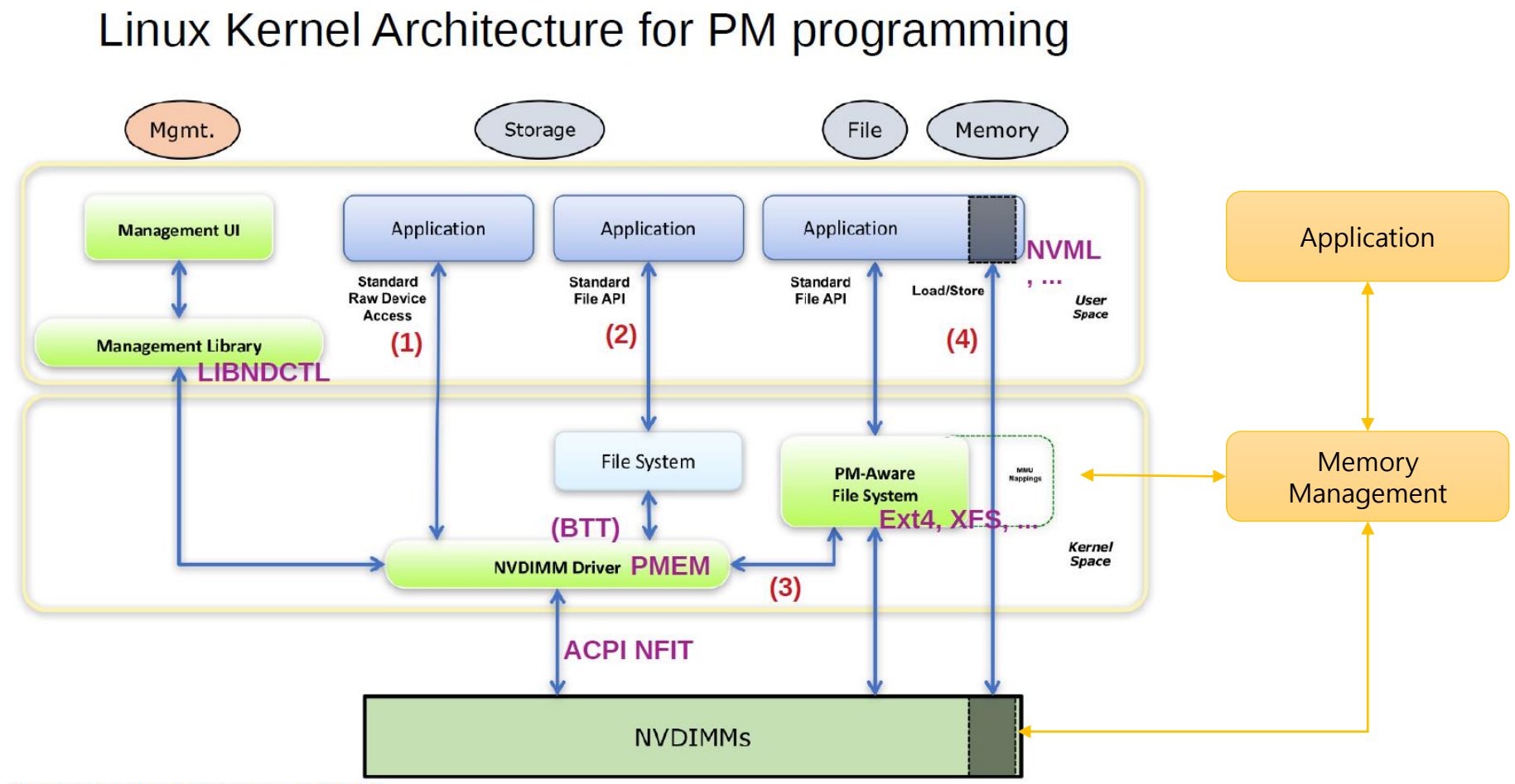
- persistent memory
  - non-volatile
  - byte-addressable
  - load/store access
  - DRAM-like speed

Use persistent memory as RAM



출처: kernel-dev-ko.github.io

Use persistent memory as RAM



출처: kernel-dev-ko.github.io

Use persistent memory as RAM

- Memory hot-plug interface 이용
  “echo dax0.0 > /sys/bus/dax/drivers/device_dax/unbind”
  “echo dax0.0 > /sys/bus/dax/drivers/kmem/new_id”
  “echo online > /sys/devices/system/memory/memoryXXX/state”

- 커널이 일반적으로 사용하는 메모리와 동일하게 인식됨

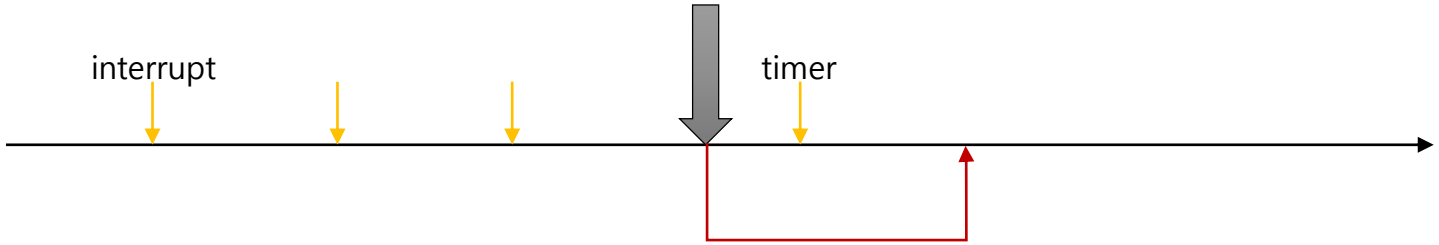
- NUMA node 로 분리하여 관리할 수 있음

Timer Events Oriented (TEO) governor

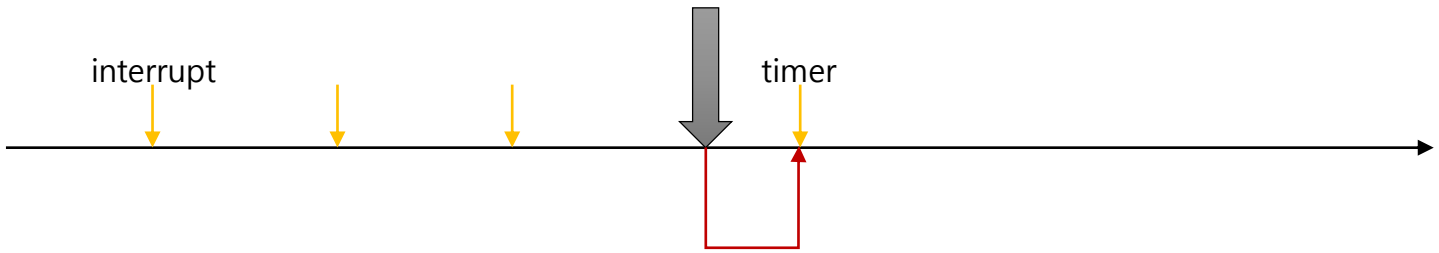
- CPUIDLE governor
- ladder
- menu
- TEO
  - next timer 정보를 주로 고려하여 idle 상태를 결정
  - cpuidle.governor=teo

Timer Events Oriented (TEO) governor

- menu



- TEO



?