1-2

1-2

- 1번 - CPU관련
    1. 어떤 프로세스(command, pid)가 I/O 수행을 위해 얼마나 오랜 시간(ns)동안 schedule out 되었는지 분석

```
emblab@vBox:~/jelee$ sudo ./lab1-1.py
PID    COMM             TIME
356    jbd2/dm-0-8      146206
356    jbd2/dm-0-8      1736692
7552   bash             1067192
356    jbd2/dm-0-8      171634
7521   kworker/u8:2     240895
356    jbd2/dm-0-8      4362554
356    jbd2/dm-0-8      166265
356    jbd2/dm-0-8      1943834
356    jbd2/dm-0-8      156375
356    jbd2/dm-0-8      3533384
430    journal-offline  84759
356    jbd2/dm-0-8      83542
356    jbd2/dm-0-8      1117502
430    systemd-journal  153069
356    jbd2/dm-0-8      141321
356    jbd2/dm-0-8      1809666
```

    2. 결론

        요구하는 command, pid와 얼마나 오랜시간동안 schedule out되어 있는지 나타내었다. io_schedule에 kprobe/kretprobe를 부착하여 구현하였고, 해당 함수가 호출될때마다 원하는 정보를 가공하여 출력하였다.

- 2번 - Memory 관련
    1. 어떤 파일(filename)에 대해 얼마나 많은 page fault가 발생했는지(횟수) 분석

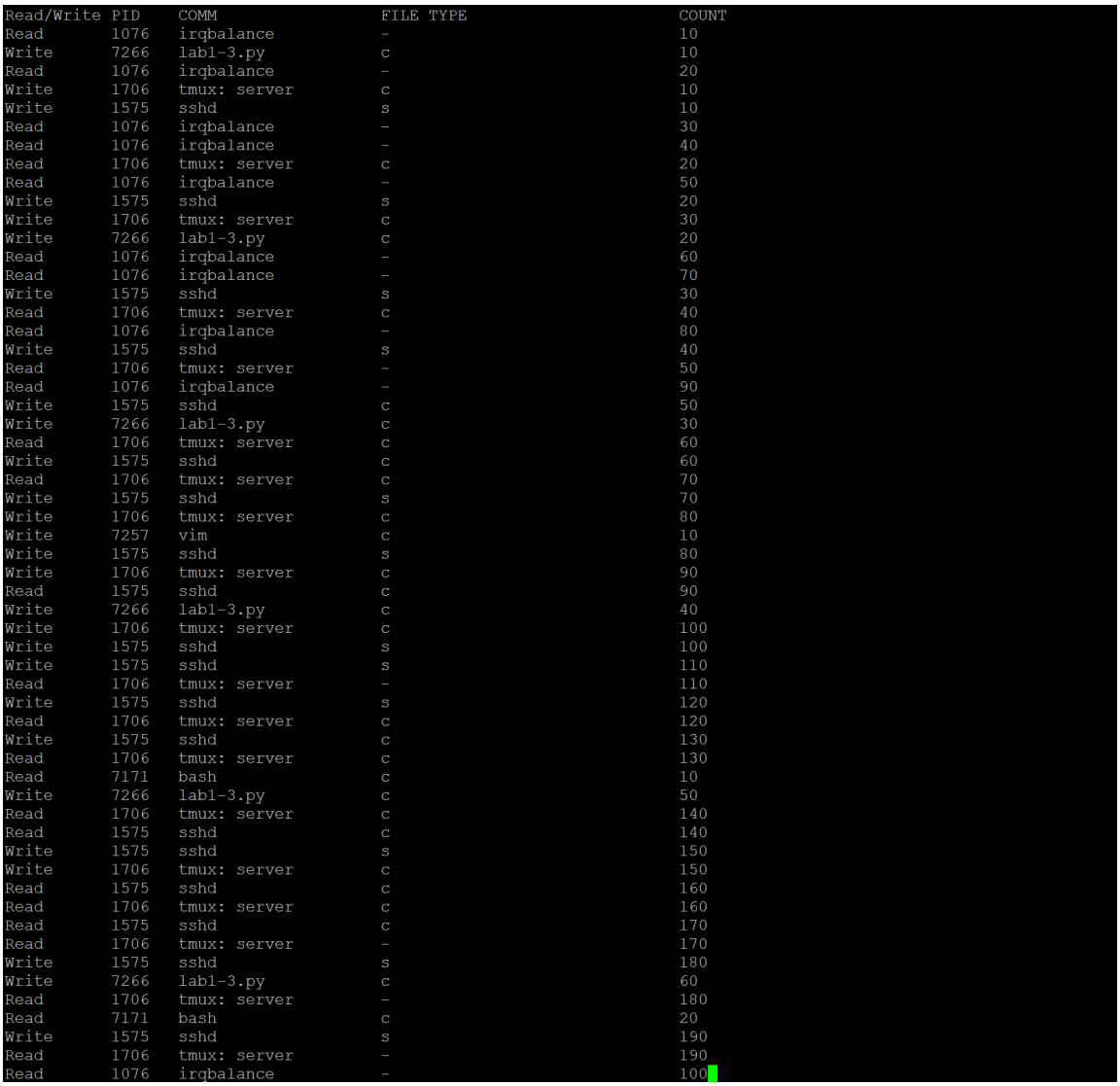
```
FILE                            PAGE_FAULT_COUNT
jelee                           4
jelee                           5
jelee                           6
jelee                           7
jelee                           8
jelee                           9
jelee                           10
jelee                           11
es2020                          1
es2020                          2
es2020                          3
es2020                          4
es2020                          5
es2020                          6
es2020                          7
jelee                           12
jelee                           13
jelee                           14
es2020                          8
es2020                          10
es2020                          9
es2020                          11
es2020                          12
es2020                          13
es2020                          14
es2020                          15
es2020                          16
es2020                          17
es2020                          19
es2020                          21
es2020                          23
es2020                          25
es2020                          26
es2020                          28
es2020                          18
es2020                          20
es2020                          22
es2020                          24
es2020                          27
es2020                          29
es2020                          30
es2020                          31
es2020                          32
es2020                          33
es2020                          34
es2020                          35
es2020                          36
es2020                          37
es2020                          38
es2020                          39
es2020                          40
es2020                          41
es2020                          42
es2020                          43
es2020                          44
jelee                           15
```

2. 결론

요구하는 filename과 해당 file이 얼마나 많은 page_fault를 유발하는지 나타내었다. handle_mm_fault 함수에 kprobe를 부착하여 구현하였다. handle_mm_fault가 발생할때마다 filename과 이전에 저장되어있던 page_fault횟수를 증가시켜주고 해당 정보를 출력한다.

- 3번 - File System 관련

    1. 어떤 프로세스(command)가 어떤 타입의 파일에(socket, regular, fifo, etc...) 어떤 입출력을(read/write) 얼마나 수행했는지 분석



Read/Write	PID	COMM	FILE TYPE	COUNT
Read	1076	irqbalance	-	10
Write	7266	lab1-3.py	c	10
Read	1076	irqbalance	-	20
Write	1706	tmux: server	c	10
Write	1575	sshd	s	10
Read	1076	irqbalance	-	30
Read	1076	irqbalance	-	40
Read	1706	tmux: server	c	20
Read	1076	irqbalance	-	50
Write	1575	sshd	s	20
Write	1706	tmux: server	c	30
Write	7266	lab1-3.py	c	20
Read	1076	irqbalance	-	60
Read	1076	irqbalance	-	70
Write	1575	sshd	s	30
Read	1706	tmux: server	c	40
Read	1076	irqbalance	-	80
Write	1575	sshd	s	40
Read	1706	tmux: server	-	50
Read	1076	irqbalance	-	90
Write	1575	sshd	c	50
Write	7266	lab1-3.py	c	30
Read	1706	tmux: server	c	60
Write	1575	sshd	c	60
Read	1706	tmux: server	c	70
Write	1575	sshd	s	70
Write	1706	tmux: server	c	80
Write	7257	vim	c	10
Write	1575	sshd	s	80
Write	1706	tmux: server	c	90
Read	1575	sshd	c	90
Write	7266	lab1-3.py	c	40
Write	1706	tmux: server	c	100
Write	1575	sshd	s	100
Write	1575	sshd	s	110
Read	1706	tmux: server	-	110
Write	1575	sshd	s	120
Read	1706	tmux: server	c	120
Write	1575	sshd	c	130
Read	1706	tmux: server	c	130
Read	7171	bash	c	10
Write	7266	lab1-3.py	c	50
Read	1706	tmux: server	c	140
Read	1575	sshd	c	140
Write	1575	sshd	s	150
Write	1706	tmux: server	c	150
Read	1575	sshd	c	160
Read	1706	tmux: server	c	160
Read	1575	sshd	c	170
Read	1706	tmux: server	-	170
Write	1575	sshd	s	180
Write	7266	lab1-3.py	c	60
Read	1706	tmux: server	-	180
Read	7171	bash	c	20
Write	1575	sshd	s	190
Read	1706	tmux: server	-	190
Read	1076	irqbalance	-	100

    2. 결론

        요구하는 command와 filed의 타입, 어떤 입출력을 몇번 수행하는지 분석하였다. vfs_read, vfs_readv, vfs_write, vfs_writev에 kprobe를 부착하여 구현하였다. 해당 함수들이 호출될때 마다 구현해놓은 함수를 호출하게 구현하고, 요구하는 정보들을 얻어와 출력하였다. 파일 타입의 경우 나타내기 나름이지만 본 프로그램에서는 정규파일 = '-', 디렉토리 = 'd', 문자장치 = 'c'. 블록장치 = 'b', 링크파일 = 'l', 파이프파일 = 'p', 소켓파일 = 's'로 나타내었다.

- 4번 - Disk I/O 관련

1. 개인 컴퓨터에 부착된 디스크(device file name)에 대해, read I/O 패턴과 write I/O 패턴을 분석 (RANDOM/SEQUENTIAL)

```
emblab@vBox:~/SNU_EMBLAB$ sudo ./lab1-4.py
Random/Sequential    COMM  DISK    SECTOR               BYTES
Random               Read          18446744073709551615 8
Random               Write sda     6444632              4096
Random               Write sda     6447312              4096
Random               Write sda     6448784              4096
Random               Write sda     23356544             53248
Random               Write sda     6445288              4096
Random               Write sda     6446888              4096
Random               Write sda     6447896              4096
Random               Write sda     6449320              4096
Random               Write sda     6449440              8192
Random               Write sda     6449800              4096
Random               Write sda     6449840              8192
Random               Write sda     6449864              4096
Random               Write sda     6449960              4096
Random               Write sda     6450024              4096
Random               Write sda     6450040              4096
Random               Write sda     6450240              4096
Random               Write sda     6450384              4096
Random               Write sda     6450408              4096
Random               Write sda     6450424              4096
Random               Write sda     6450472              4096
Random               Write sda     6450696              4096
Random               Write sda     6450712              4096
Random               Write sda     6450728              8192
Random               Write sda     6450776              12288
Random               Write sda     16189440             4096
Random               Write sda     36968584             4096
Random               Read  sda     18446744073709551615 0
Random               Write sda     23356648             4096
Random               Read  sda     18446744073709551615 0
Sequential           Read          18446744073709551615 8
Sequential           Read          18446744073709551615 8
Sequential           Read          18446744073709551615 8
Random               Write sda     23356656             8192
Random               Read  sda     18446744073709551615 0
Random               Write sda     23356672             4096
Random               Read  sda     18446744073709551615 0
Sequential           Read          18446744073709551615 8
```

2. 결론

   요구하는 정보에 대해 판단하기 위해 blk_start_request()에 kprobe를 부착하여 호출될 때마다 read/write정보, sector정보와 데이터 길이를 가져와 random인지 sequential인지 패턴을 파악했다. 피피티에 명시되어있는 blk_mq_start_request가 아닌 blk_start_request에 kprobe를 부착해야했다. 이유는 우분투에서 멀티큐를 사용하지 않는 것이 이유로 추정된다. 위와 같은 출력을 나타냈다.