# REPORT

## Aim

To understand the difference between how virutal memory is alloted and mapped using kmalloc and vmalloc.

## Methodology

### Software Walk

1. Given page VA, we can perform a software page table walk doing the following in order,
    - pgd_offset() - returns pgd offset given mm_struct of task and VA
    - pud_offset() - returns pud offset given VA and pgd offset
    - pmd_offset() - returns pmd offset given VA and pud offset
    - pte_offset_map() - returns pte given VA and pmd offset
2. If any of the above returns a none and bad value we exit with return value as 0
3. If everything executes succesfully we obtain the struct page using pte_page() and using the page struct we obtain the Physical address of page (PA) using page_to_phys()
4. Now we can obtain the PA of VA by taking the last 12 bits of VA and Oring it with the page PA.
5. Now this PA is returned

### Module (a) logic

1. Allocate memory for a user defined struct using kmalloc()/vmalloc()
2. Initialize the struct data to a test value
3. Type cast the pointer to the newly created struct to unsigned long which gives us the VA
4. Now display the VA
5. get the PA using the virt_to_phys() macro by passing pointer to struct and display it
6. get the PA using the software walk method as described above using VA

### Module (b) logic

1. Keeping module above module loaded
2. Load a new module with above displayed VA as a parameter

3. Access the struct data that was input in the above module and print it
4. Now try to access the VA using the same above two methods
5. for virt_to_phys() make sure to cast the address it to struct type before invoking the macro

# Experiments

### 1- Kmalloc

```
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo insmod part2_a.ko choice=0
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo insmod part2_b.ko va=18446612135152665992
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo rmmod part2_b.ko
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo rmmod part2_a.ko
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$
```

*Illustration 1: commands to run module with kmalloc()*

1. Insert module(a) with choice=0 (invoked kmalloc)
2. With keeping the above module running insert module(b)
3. It is observed that the output using both the virt_to_phys() and software walk produce the same correct output in both modules
4. The **PA obtained in all 4 cases are the same** and correct
5. The second module is rightly able to access the first module's struct data
6. It is to be noted that along the path of traversal of the software page table walk, the **pgd-offset value is different** (can be seen below) **for the two modules** but map to the same pud-offset and hence the walks there after are the same.

```
Feb 16 23:59:33 prashanth-VirtualBox kernel: [22523.665023] Part-2: ----------------------INSERTED-----------------------
Feb 16 23:59:33 prashanth-VirtualBox kernel: [22523.665028] Node-P: ffff8800a92f4588 VA: 18446612135152665992
Feb 16 23:59:33 prashanth-VirtualBox kernel: [22523.665030] pgd-off:ffff88008c8c9880 pud-off:ffff880001ff8010 pmd-off:ffff88015fffea48 pte:80000000a92f4163
Feb 16 23:59:33 prashanth-VirtualBox kernel: [22523.665031] Physical Address Obtained from virt_to_phys : 00000000a92f4588
Feb 16 23:59:33 prashanth-VirtualBox kernel: [22523.665032] Physical Address Obtained from software walk: 00000000a92f4588
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506937] Part-2-Access: ----------------------INSERTED-----------------------
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506943] Node-P: ffff8800a92f4588 VA: 18446612135152665992
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506945] Value at Node: 10
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506947] pgd-off:ffff8800a5084880 pud-off:ffff880001ff8010 pmd-off:ffff88015fffea48 pte:80000000a92f4163
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506949] Physical Address Obtained from virt_to_phys : 00000000a92f4588
Feb 16 23:59:54 prashanth-VirtualBox kernel: [22544.506951] Physical Address Obtained from software walk: 00000000a92f4588
Feb 17 00:00:03 prashanth-VirtualBox kernel: [22553.287929] Part-2-Access: ----------------------REMOVED----------------------
Feb 17 00:00:05 prashanth-VirtualBox kernel: [22555.707895] Part-2: ----------------------REMOVED----------------------
```

*Illustration 2: output for kmalloc()*

### 2- Vmalloc

```
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo insmod part2_a.ko choice=1
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo insmod part2_b.ko va=18446683600585457664
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo rmmod part2_b.ko
prashanth@prashanth-VirtualBox:~/Modules/a1/part2$ sudo rmmod part2_a.ko
```

*Illustration 3: Commands to run modules with vmalloc()*

1. Insert module(a) with choice=1 (invoked vmalloc)

2. With keeping the above module running insert module(b)
3. It is observed that the **only software walk fetches the same correct PA** in both modules, and **virt_to_phys() fetches the wrong PA**.
4. The second module is rightly able to access the first module's struct data with input VA
5. It is to be noted that along the path of traversal of the software page table walk, the **pgd-offset value is different** (can be seen below) **for the two modules** but map to the same pud-offset and hence the walks there after are the same.

```
Feb 16 23:57:52 prashanth-VirtualBox kernel: [22422.057947] Part-2: ----------------------INSERTED-----------------------
Feb 16 23:57:52 prashanth-VirtualBox kernel: [22422.057963] Node-P: ffffc90000eb8000 VA: 18446683600585457664
Feb 16 23:57:52 prashanth-VirtualBox kernel: [22422.057967] pgd-off:ffff8800a9340c90 pud-off:ffff88015a09a000 pmd-off:ffff88015a09b038 pte:800000014e745163
Feb 16 23:57:52 prashanth-VirtualBox kernel: [22422.057970] Physical Address Obtained from virt_to_phys : 0000410000eb8000
Feb 16 23:57:52 prashanth-VirtualBox kernel: [22422.057972] Physical Address Obtained from software walk: 000000014e745000
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990737] Part-2-Access: ----------------------INSERTED-----------------------
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990746] Node-P: ffffc90000eb8000 VA: 18446683600585457664
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990749] Value at Node: 10
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990754] pgd-off:ffff8800bc9c1c90 pud-off:ffff88015a09a000 pmd-off:ffff88015a09b038 pte:800000014e745163
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990757] Physical Address Obtained from virt_to_phys : 0000410000eb8000
Feb 16 23:58:23 prashanth-VirtualBox kernel: [22452.990759] Physical Address Obtained from software walk: 000000014e745000
Feb 16 23:58:28 prashanth-VirtualBox kernel: [22458.045997] Part-2-Access: ----------------------REMOVED------------------------
Feb 16 23:58:33 prashanth-VirtualBox kernel: [22463.118761] Part-2: ----------------------REMOVED------------------------
```

*Illustration 4: Output for vmalloc()*

## Observations

1. While performing **software walks** on the kernel memory using different modules using kmalloc()/vmalloc() **always points us to the right PA**
2. The **pgd-offset** which performing software walks for the same kernel VA from different modules would be **different** but they ultimately map to the same pud-offset and there by inheritantly to the right PA

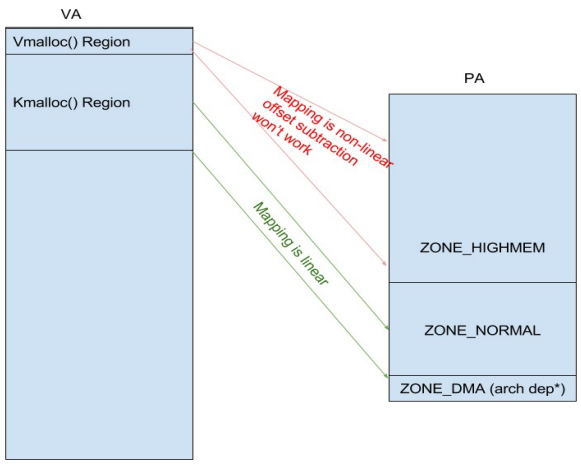


*Illustration 5: Mapping of kmalloc vs vmalloc*

3. **virt_to_phys()** macro fetches the **right PA while using kmalloc()** as it just subtracts the kernel memory offset for current architecture from VA to give its PA. And this will work for kmalloc as it is mapped on the zone_normal region of the physical address space with a linear offset.
4. **virt_to_phys()** macro fetches the **incorrect PA while using vmalloc()** as mapping is non-linear using vmalloc() is it may allocate space from any free space in highmem.