

Linux Memory Management

An Overview

Anshuman Khandual

Contents

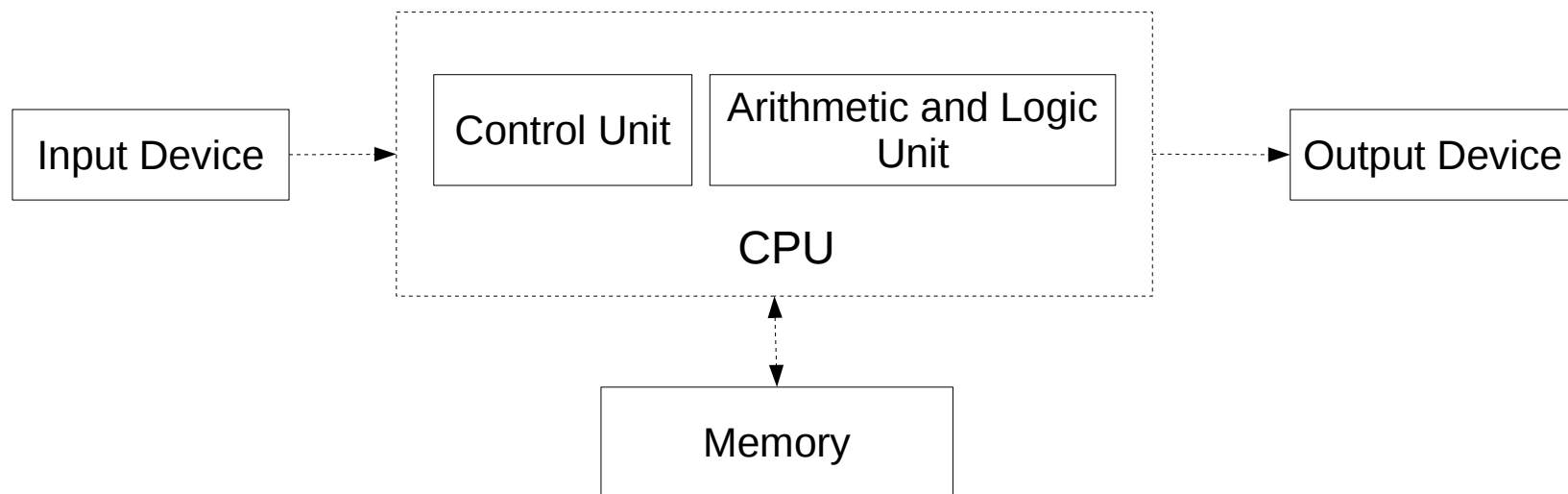
- Introduction
- Platform Memory
- MMU
- Virtual Memory
- Physical Memory
- Analyze Memory Performance

Introduction

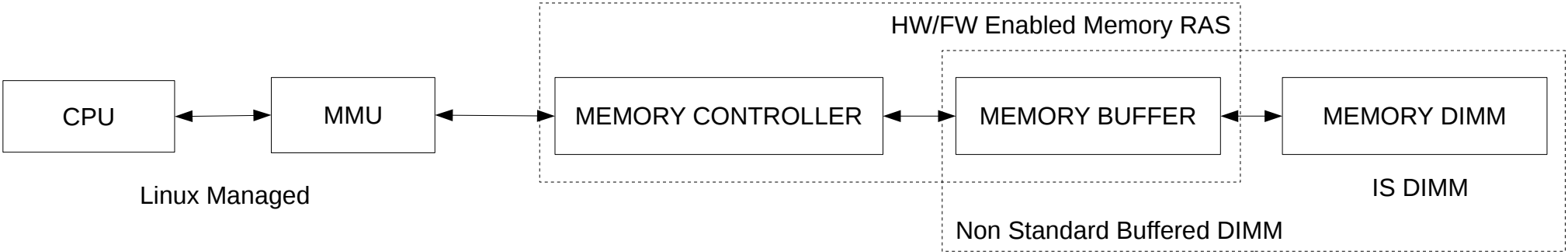
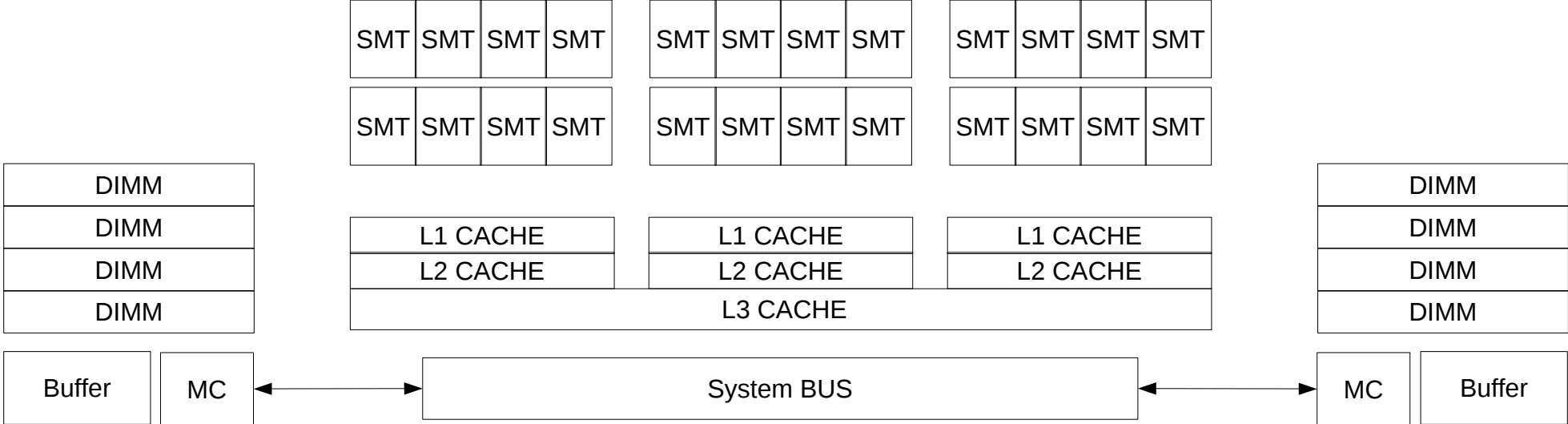
- Main Memory - A non persistent storage of memory which requires electricity
- *Joint Electron Device Engineering Council* (JEDEC) specifies standards for memory
- *Double Data Rate* (DDR) is the prevalent *Dynamic Random Access Memory* (DRAM)
- DDR standards
 - DDR3 (Released in 2007)
 - DDR4 (Released in 2014)
 - DDR5 (Will be released around 2020)
- DRAM chips are placed on *Dual In Line Memory Module* (DIMM) cards
- Memory DIMM Types
 - Non ECC DIMM (Laptop, Desktop etc)
 - ECC (Workstations, Server)
 - Industry standard DIMM (Compatible with most systems, cheaper)
 - Proprietary DIMM (Buffered, faster, more RAS features, expensive)

Why Memory is Required

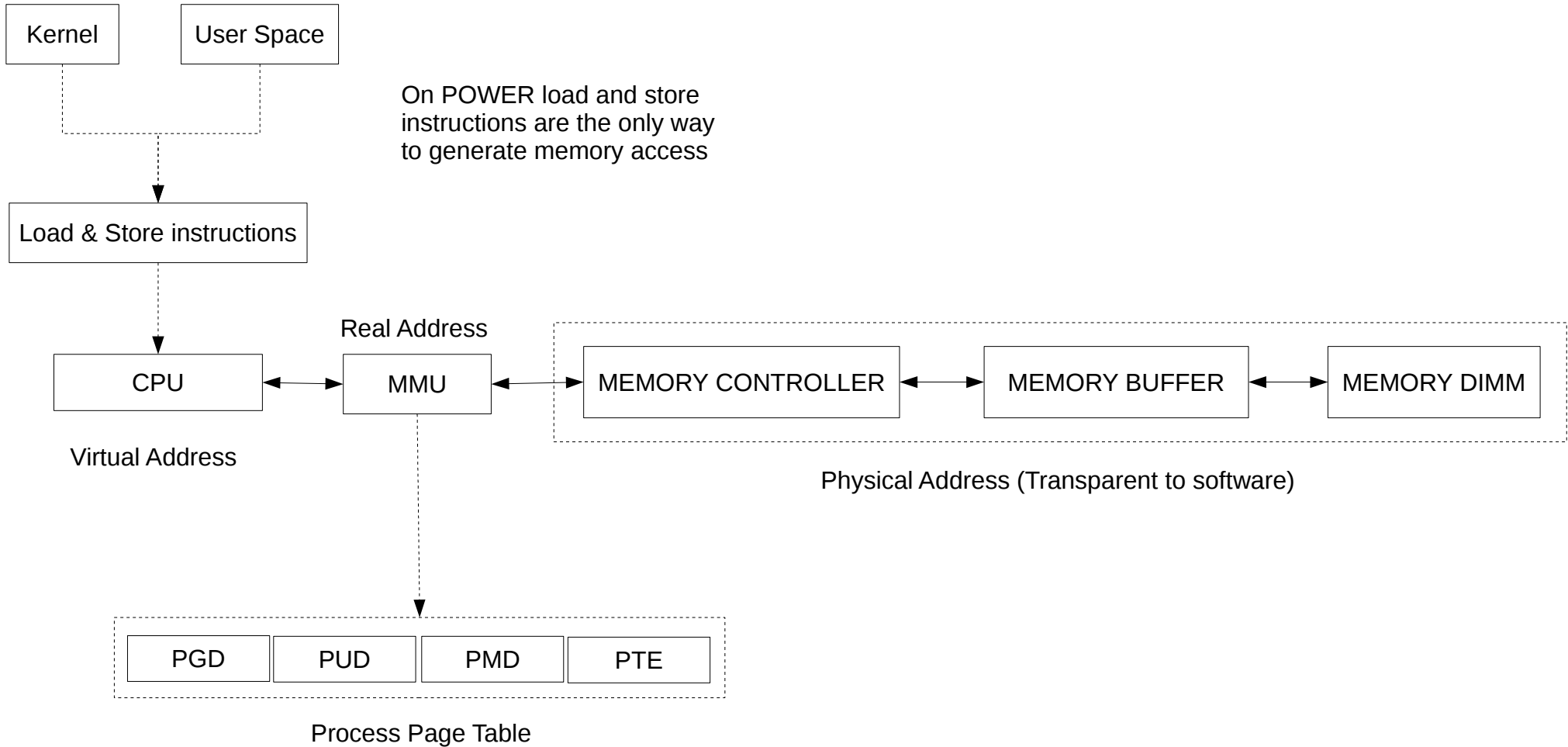
- Working set is fundamental to Jon Von Neuman computer architecture
- Similar to the concept of stack in push down automation
- There is a need to have working memory to solve a problem
- In matrix multiplication, the matrices need to be stored some where
- In image transformation, the image (it's pixels) needs to be stored some where
- Main memory (RAM) provides the infrastructure of a working set of problem data



Platform Memory



MMU Address Translation



Virtual Memory Area

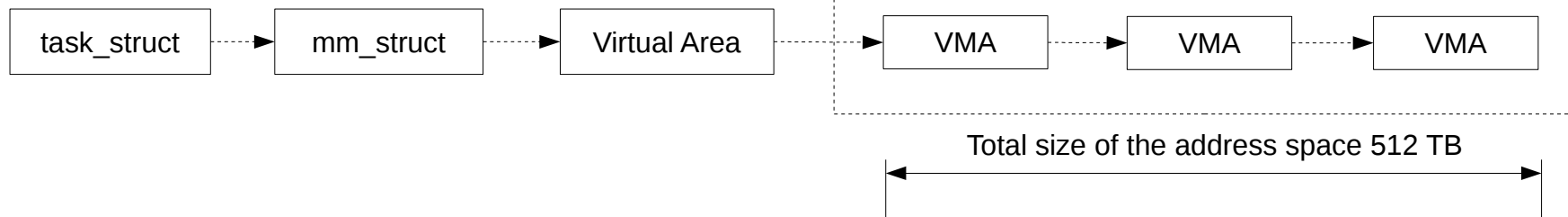
Memory Management System Calls

mprotect / madvise / mremap / mbind

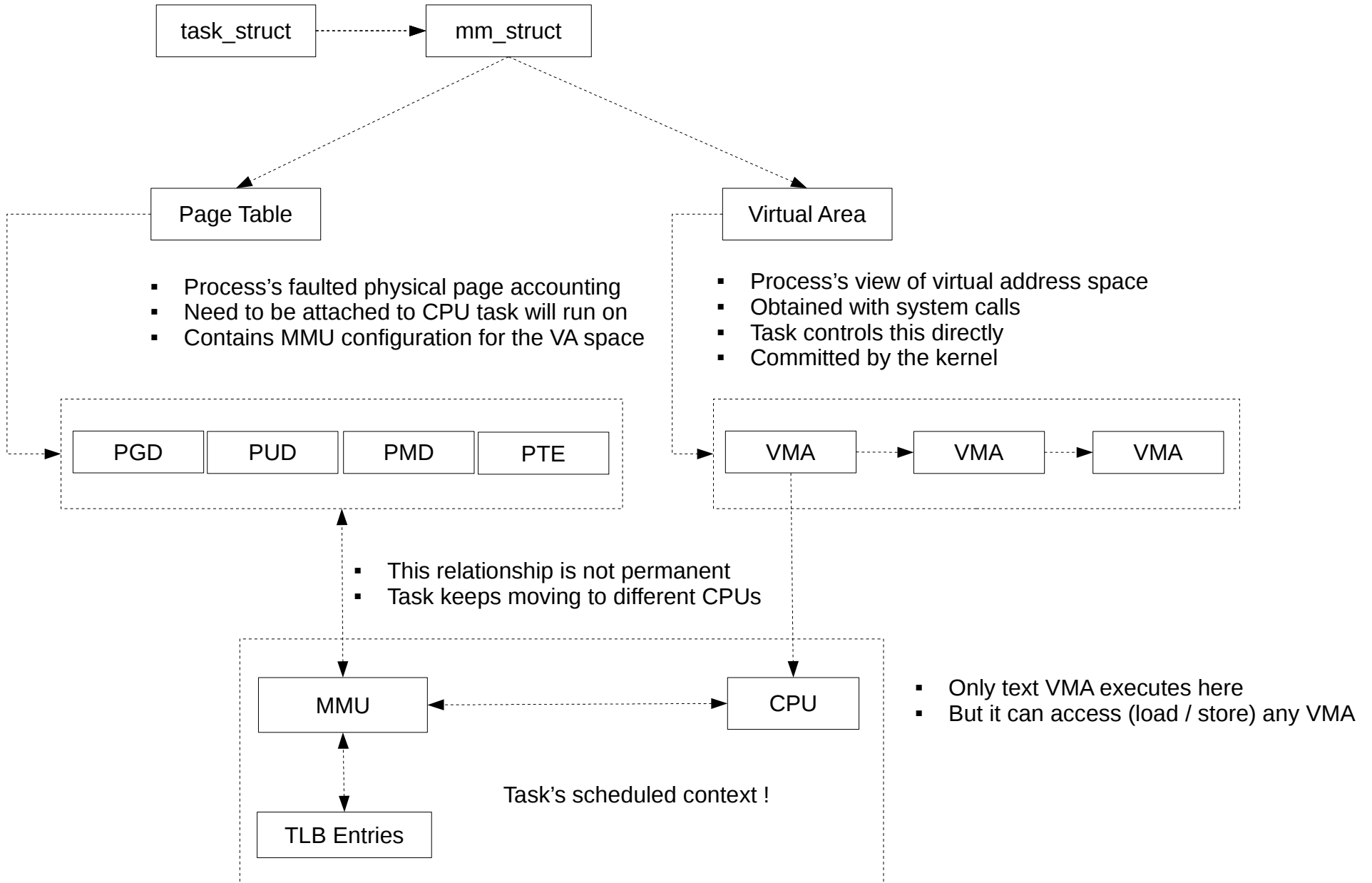
Changes existing VMA

mmap / brk / sbrk / munmap

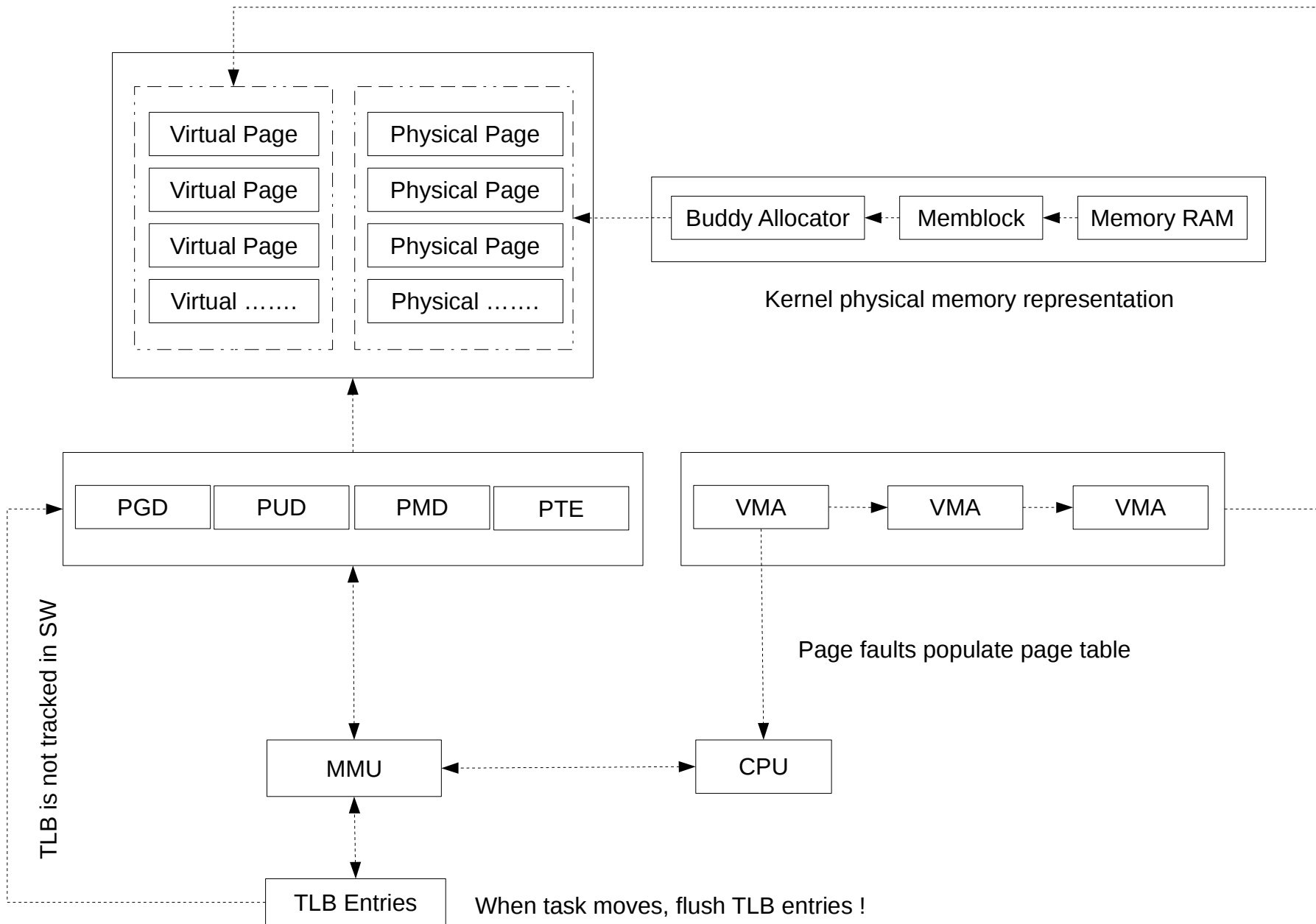
Creates, expands, contracts, destroys VMA



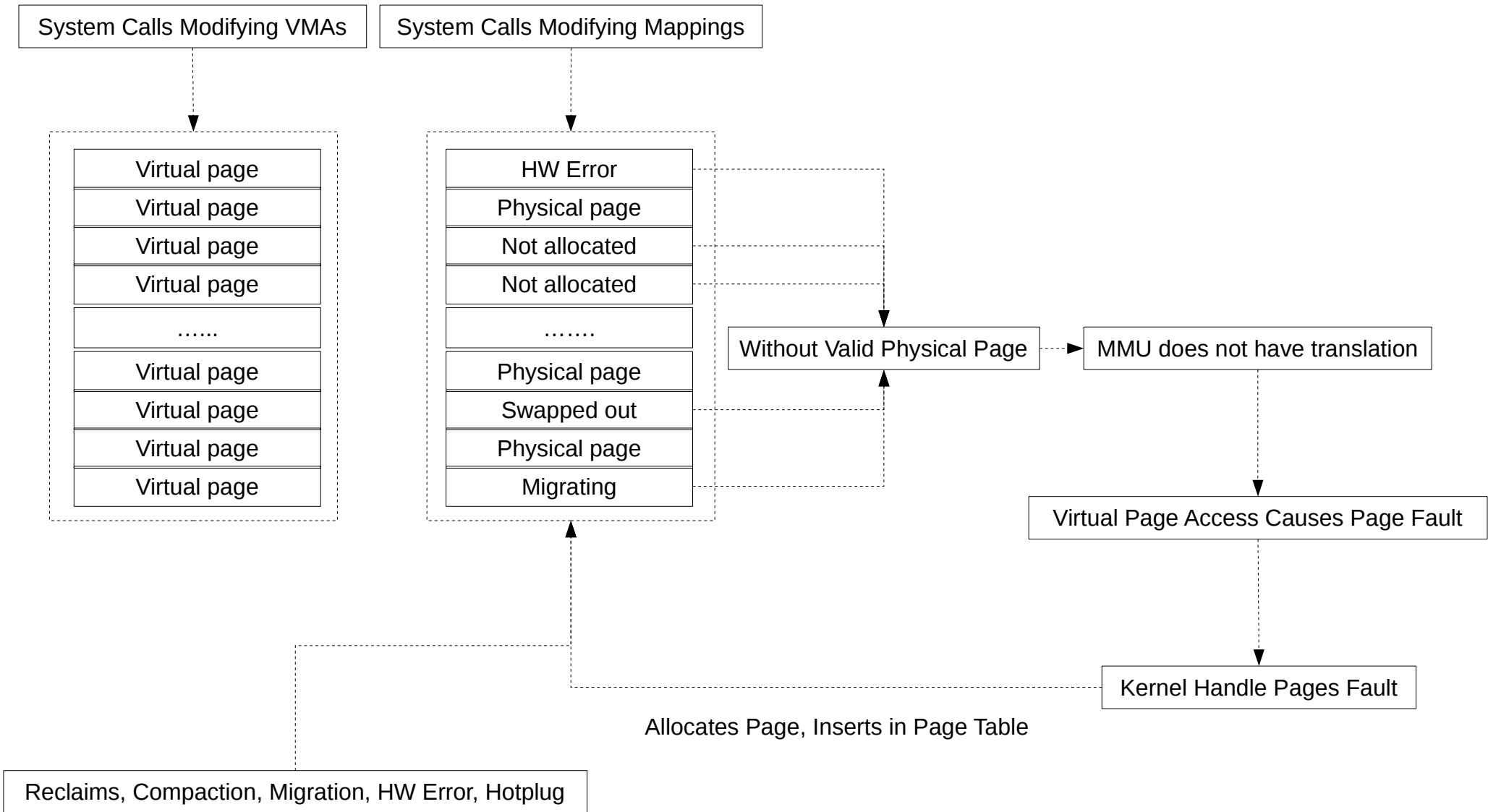
Linux Address Translation



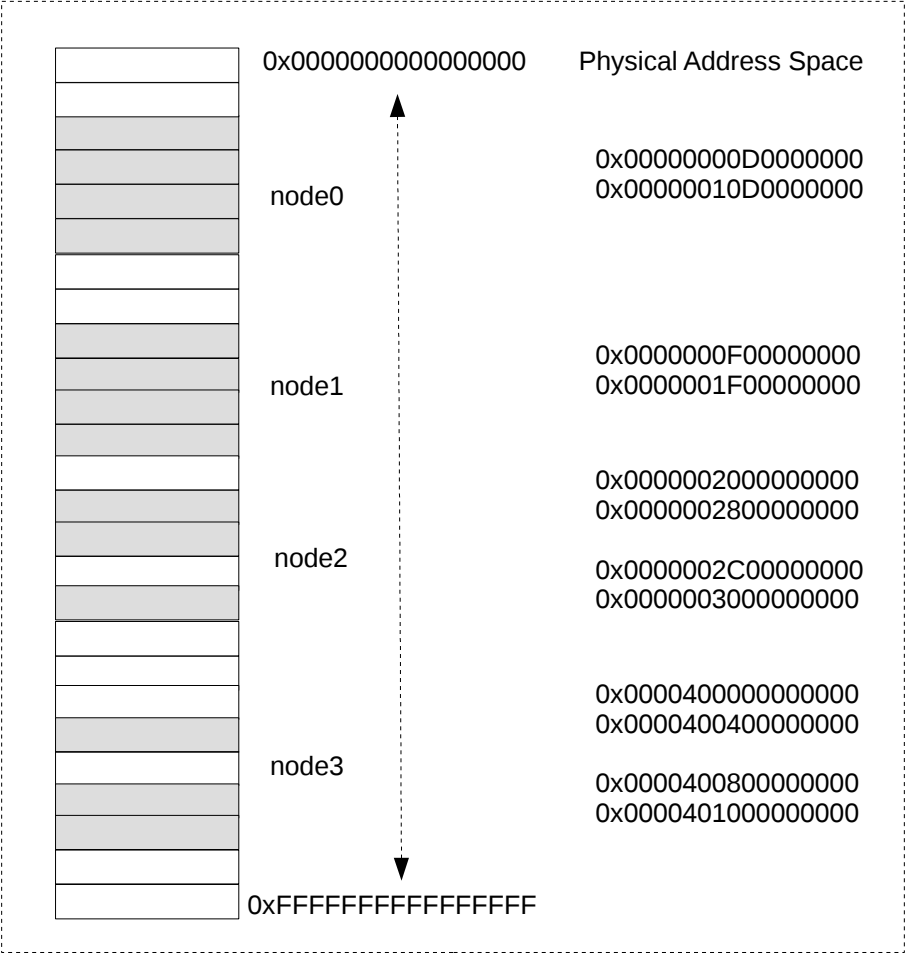
Linux Address Translation



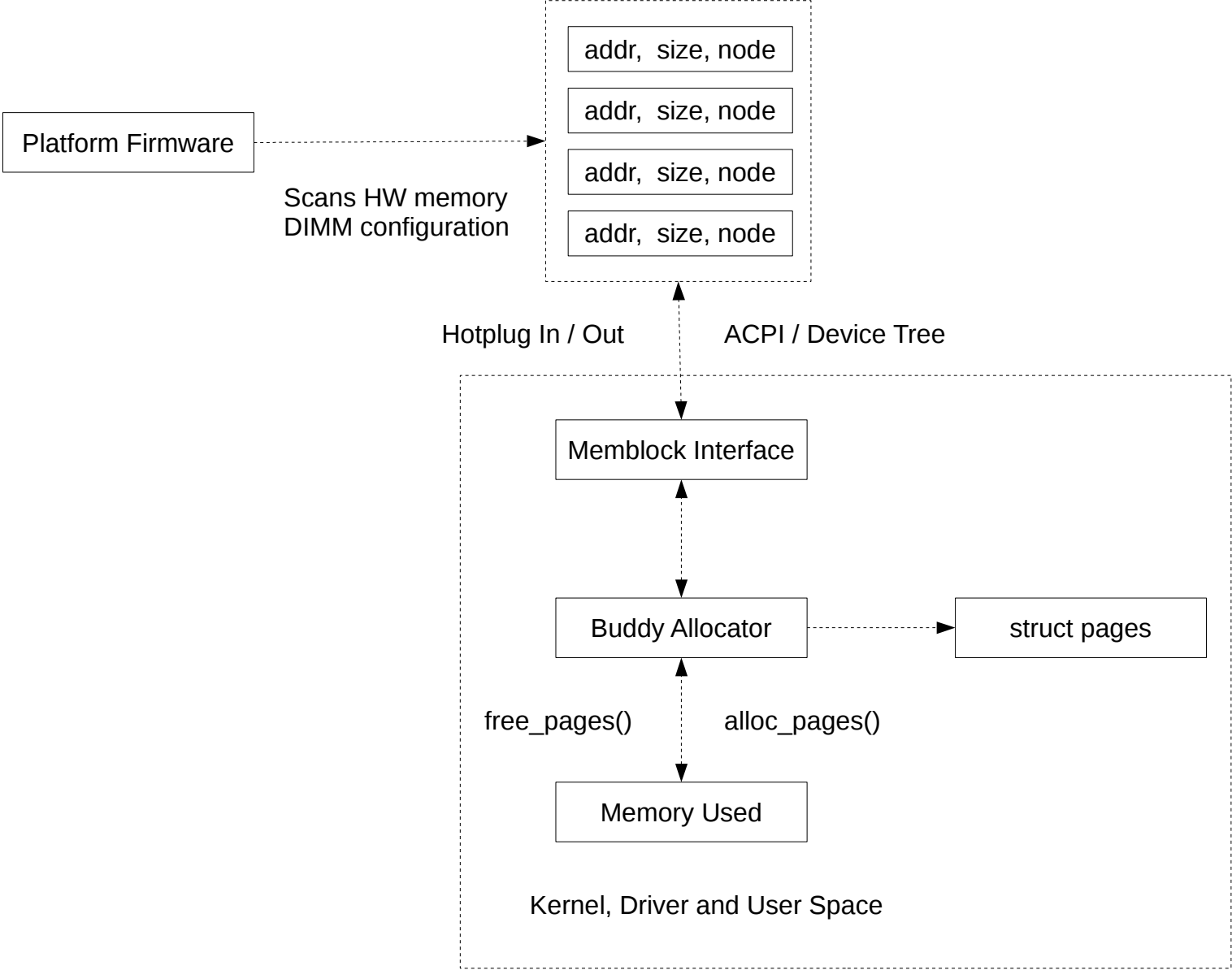
Process Page Table



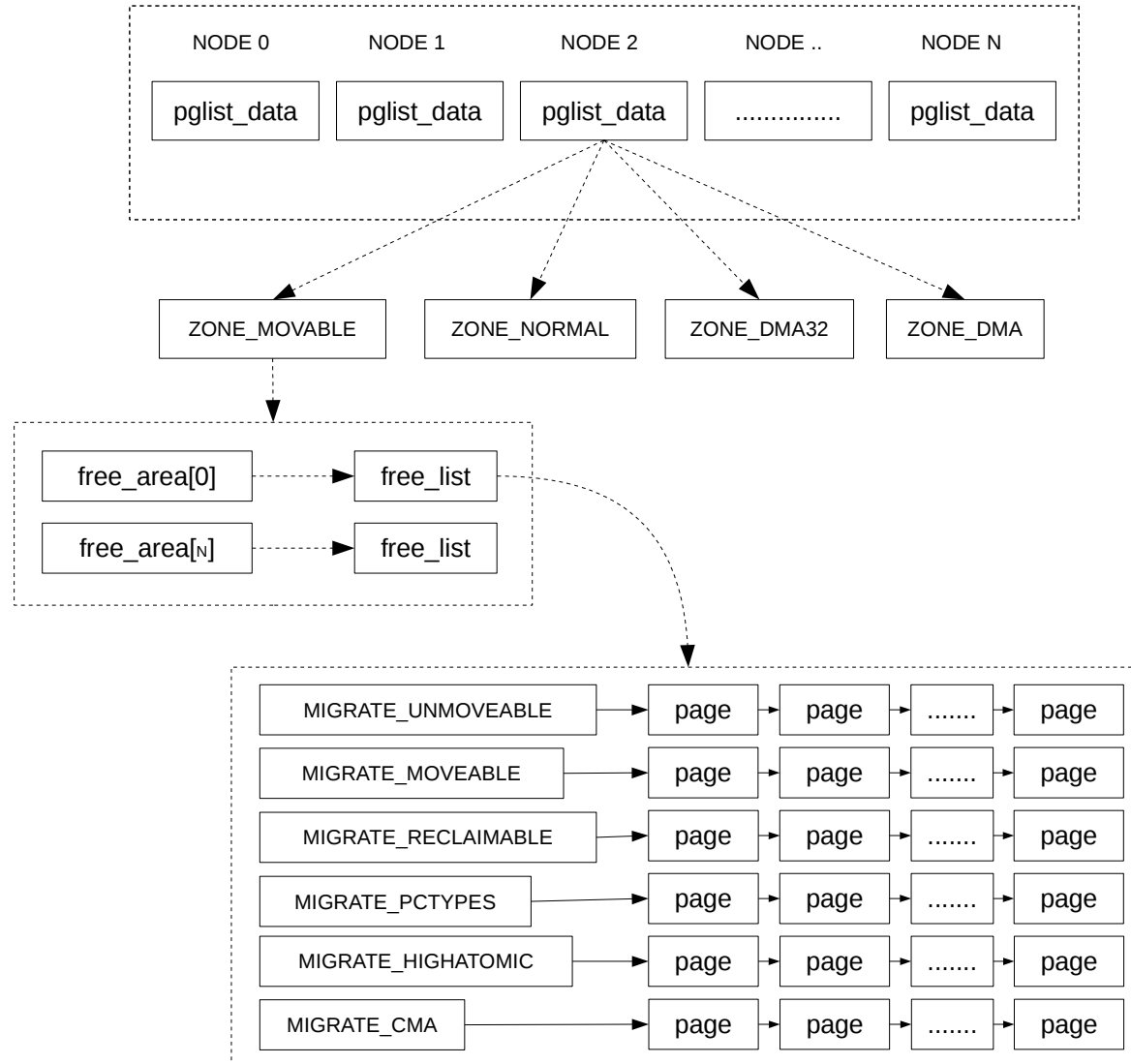
Physical Memory



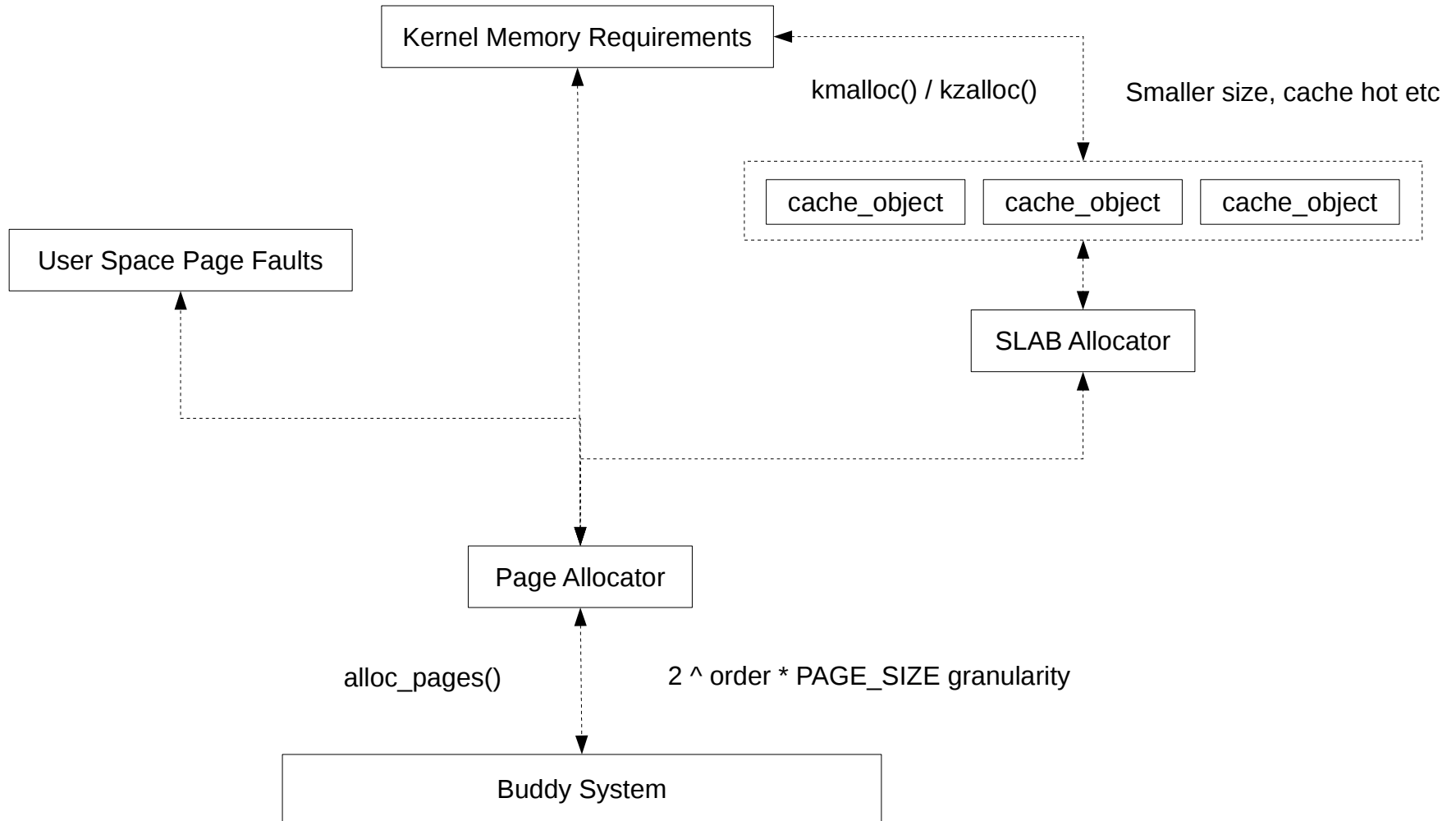
Physical Memory



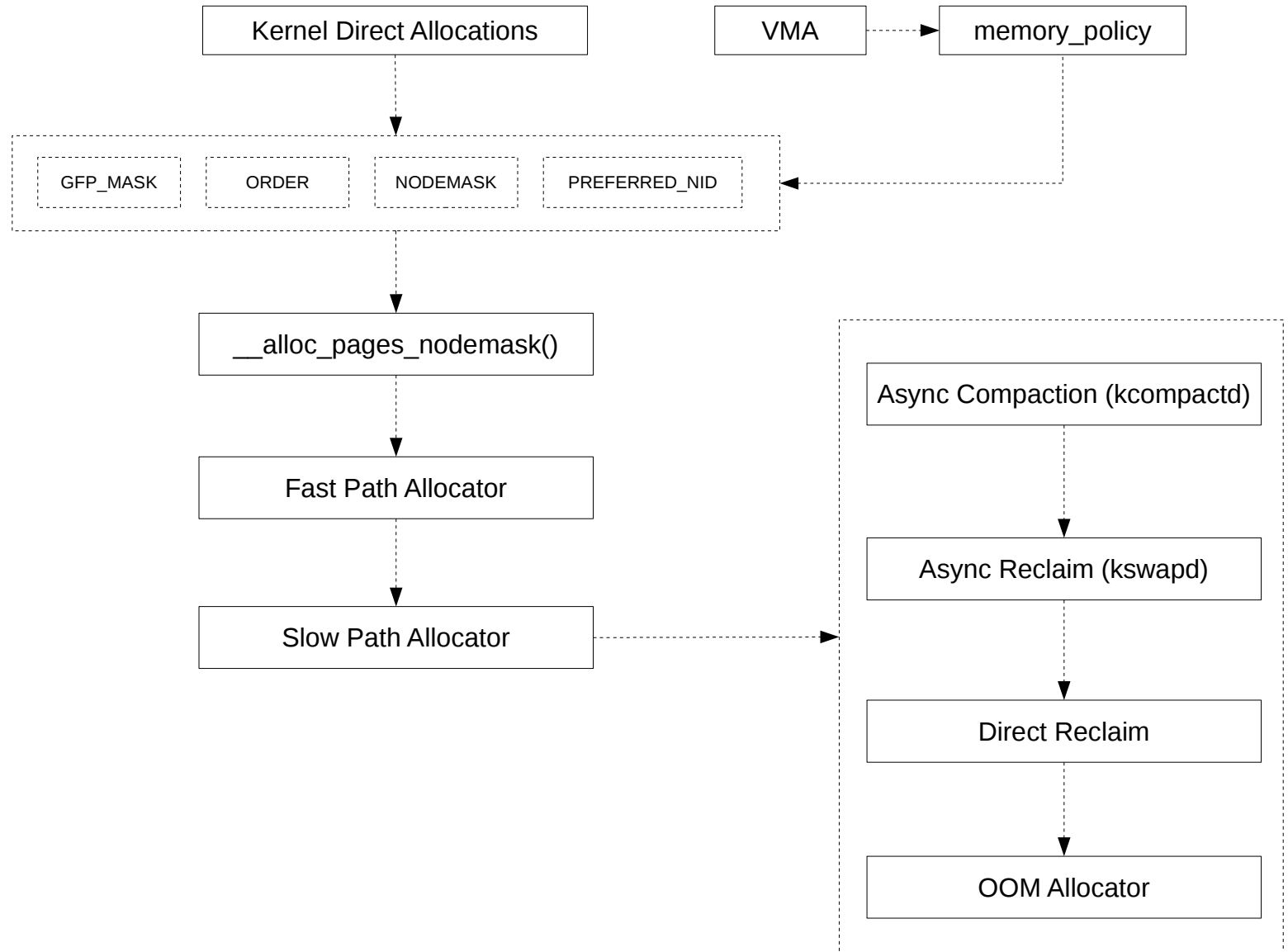
Buddy System



Physical Memory Allocator



Page Allocator



Analyze Memory Performance

```
char *ptr;
```

```
ptr = mmap(NULL, PAGE_SIZE * NR_PAGES, PROT_READ | PROT_READ,  
MAP_PRIVATE | MAP_ANON, -1, 0);
```

```
memset(ptr, 0, size);
```

```
munmap(ptr, size);
```

Allocate virtual memory buffer

Sequential access

Release virtual memory buffer

Analyze Memory Performance

```
char *ptr;
```

```
ptr = mmap(NULL, PAGE_SIZE * NR_PAGES, PROT_READ | PROT_WRITE,
MAP_PRIVATE | MAP_ANON, -1, 0);
```

```
memset(ptr, 0, size);
```

SW cost

VMA

Virtual Page

Virtual Page

Virtual Page

Virtual

Virtual Page

NULL

NULL

NULL

NULL

NULL

NR_PAGES

SW cost

First Byte Access

Page Fault

Address fetch retried !

HW cost

PAGE_SIZE Byte Accesses

Address Translation

Memory Store

RAM Access

Cache Access

RAM Access

Cache Access

.....

.....

RAM Access

Cache Access

TLB Hit

OR

TLB Miss

TLB Load

NR_PAGES

Analyze Memory Performance

```
char *ptr;
```

```
ptr = mmap(NULL, PAGE_SIZE * NR_PAGES, PROT_READ | PROT_WRITE,  
MAP_PRIVATE | MAP_ANON, -1, 0);
```

```
memset(ptr, 0, size);
```

```
munmap(ptr, size);
```

Tear down mapping for the memory buffer

Flush TLB (This is tricky !!!)

Local TLB Flush

OR

Global TLB Flush

VMA

Virtual Page

Virtual Page

Virtual Page

Virtual

Virtual Page

NULL

NULL

NULL

NULL

NULL

NR_PAGES

Free relevant portions of the page table

Free the faulted in pages if any !

Thank You