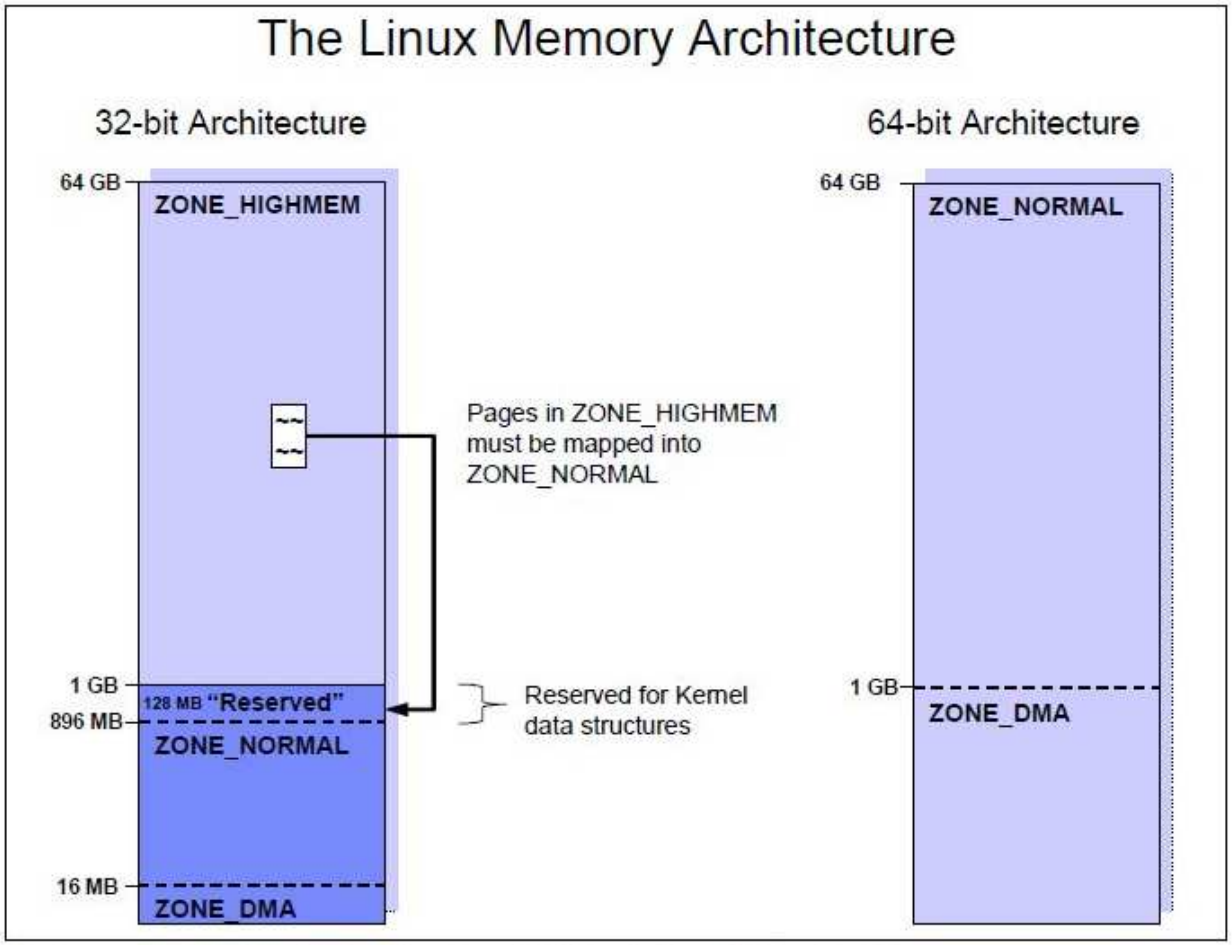
Memory Architecture

John Franco
Electrical Engineering and Computing Systems
University of Cincinnati

Virtual Memory

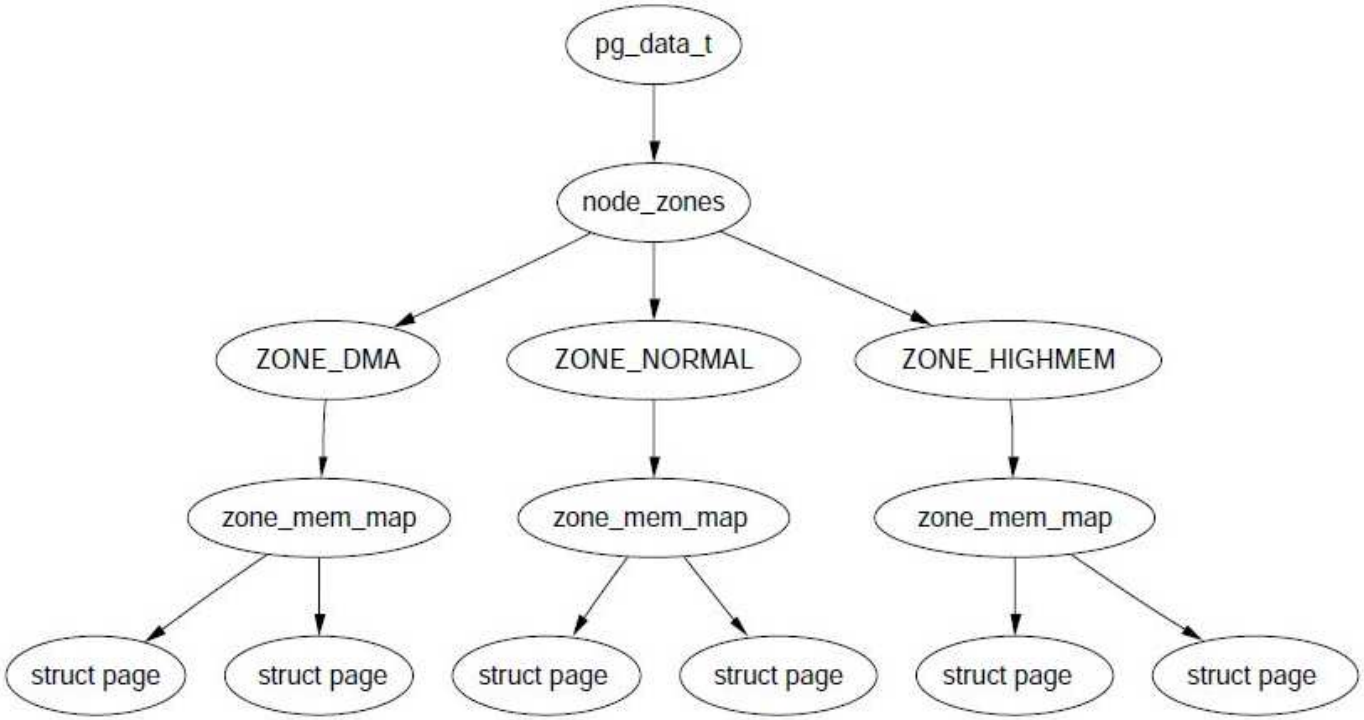
- Memory may be arranged into banks that incur a different cost to use depending on processor.
- Examples: memory assigned to each CPU, memory for DMA.
- Banks in such architectures are at varying distances,
  are called NUMA (Non-Uniform Memory Access) architectures.
- Each bank is called a node
- Each node is divided into blocks comprising zones which represent ranges within memory.
- Linux 32 bits has three zones: `ZONE_DMA`, `ZONE_NORMAL`, `ZONE_HIGHMEM`
- Node-local allocation policy: use frames from the node closest to the running CPU. As processes tend to run on the same CPU or can be explicitly bound, it is likely the memory from the current node will be used.

Virtual Memory



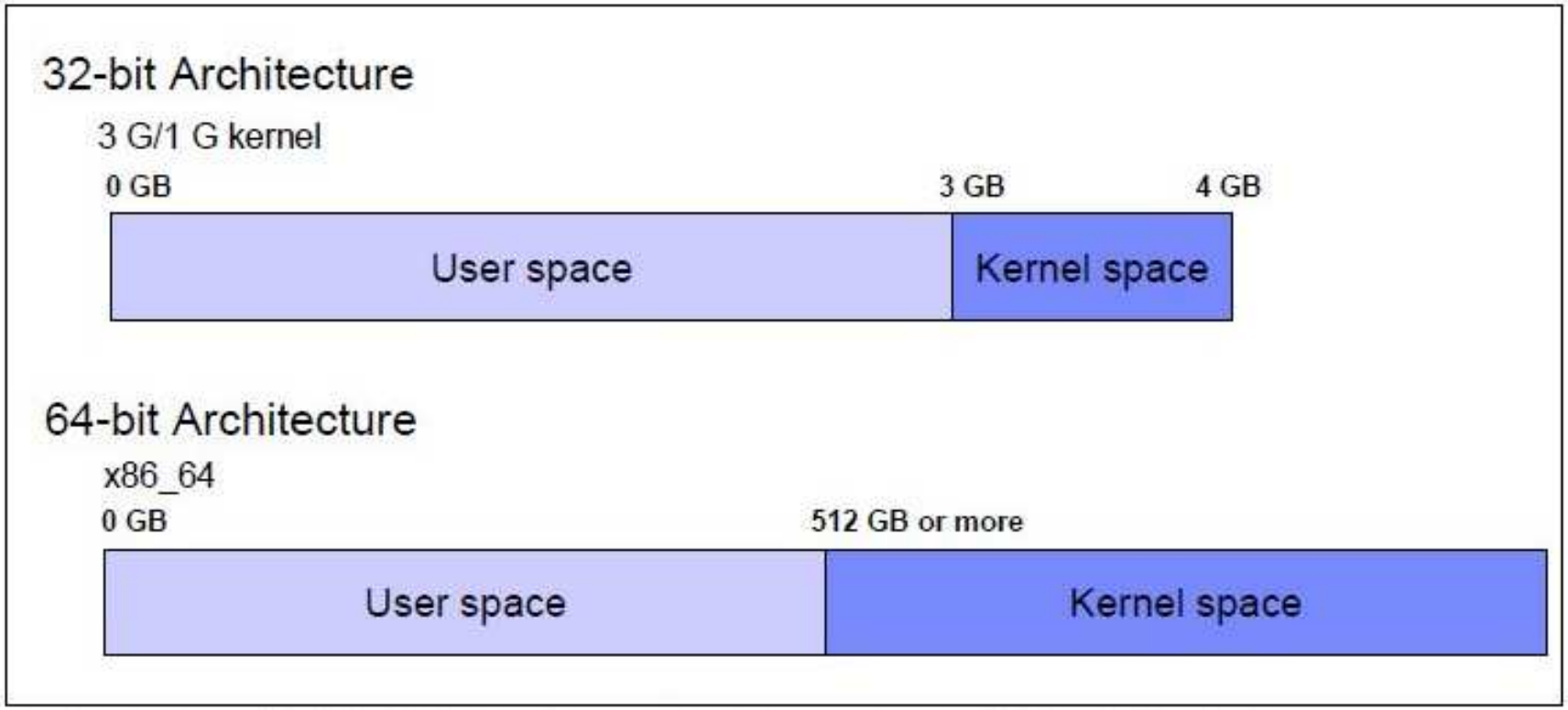
- 32 bits: only the lower 1GB can be accessed directly
- `ZONE_NORMAL`: Can be directly mapped by the kernel above 1GB of address space. Many kernel operations can only take place in `ZONE_NORMAL` so it is the most performance critical zone.
- `ZONE_DMA`: For compatibility with ISA devices with 24 bit addresses
- 64 bits: memory mapping to Normal zone is unnecessary

Virtual Memory



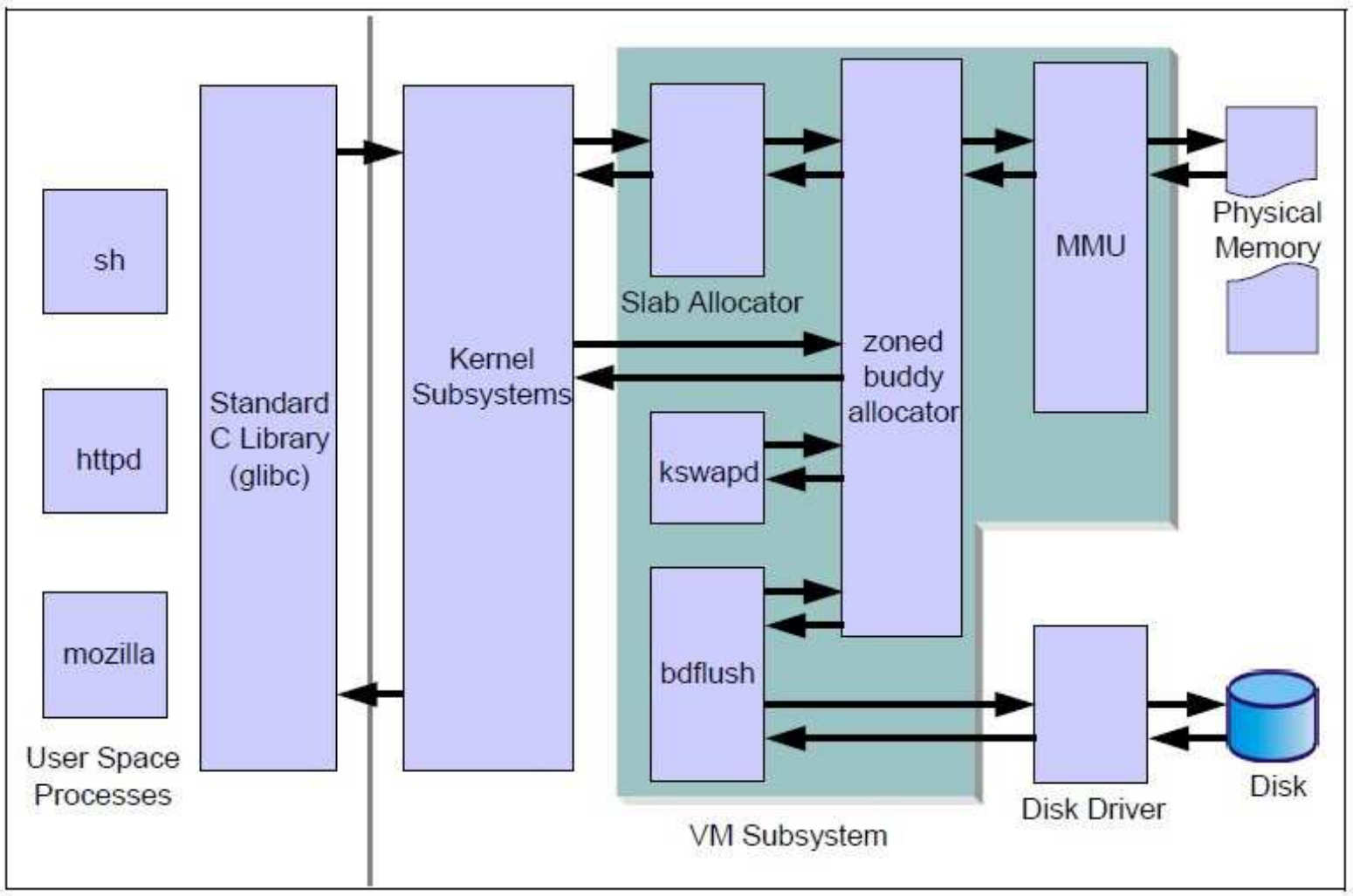
- `struct_page` objects use 45MB in `ZONE_NORMAL` to reference 1GB of frames, referencing 4GB requires 180MB which is a significant part of `ZONE_NORMAL`.

Virtual Memory



- 32 bits: only 4GB can be addressed
- 64 bits: essentially no limit to the address space

Virtual Memory Manager

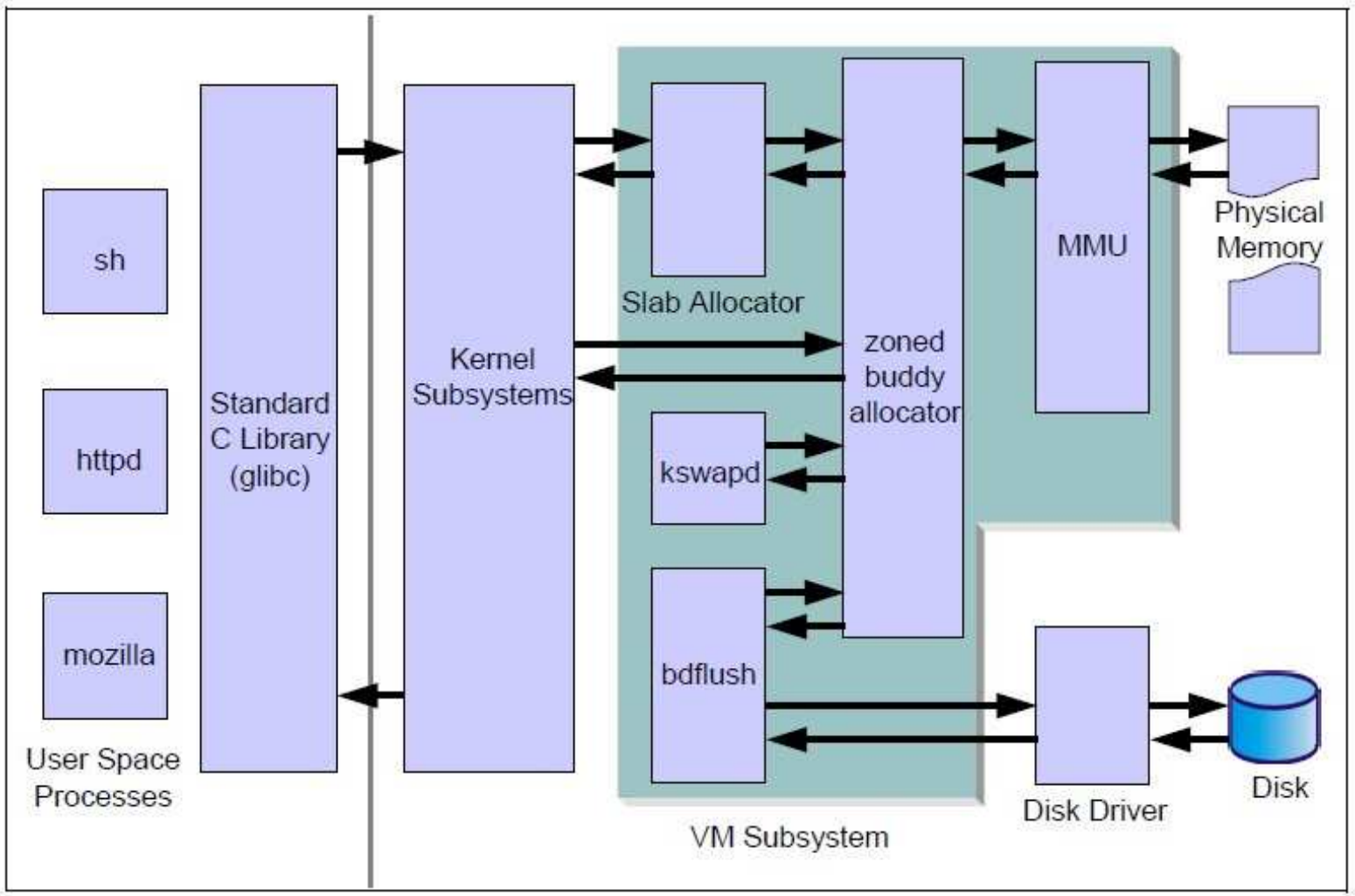


- free pages are allocated using the buddy system
- kswapd is called by the buddy allocator when a request for contiguous space is too large to be met. Pages that have not been accessed recently are evicted.

Virtual Memory Manager

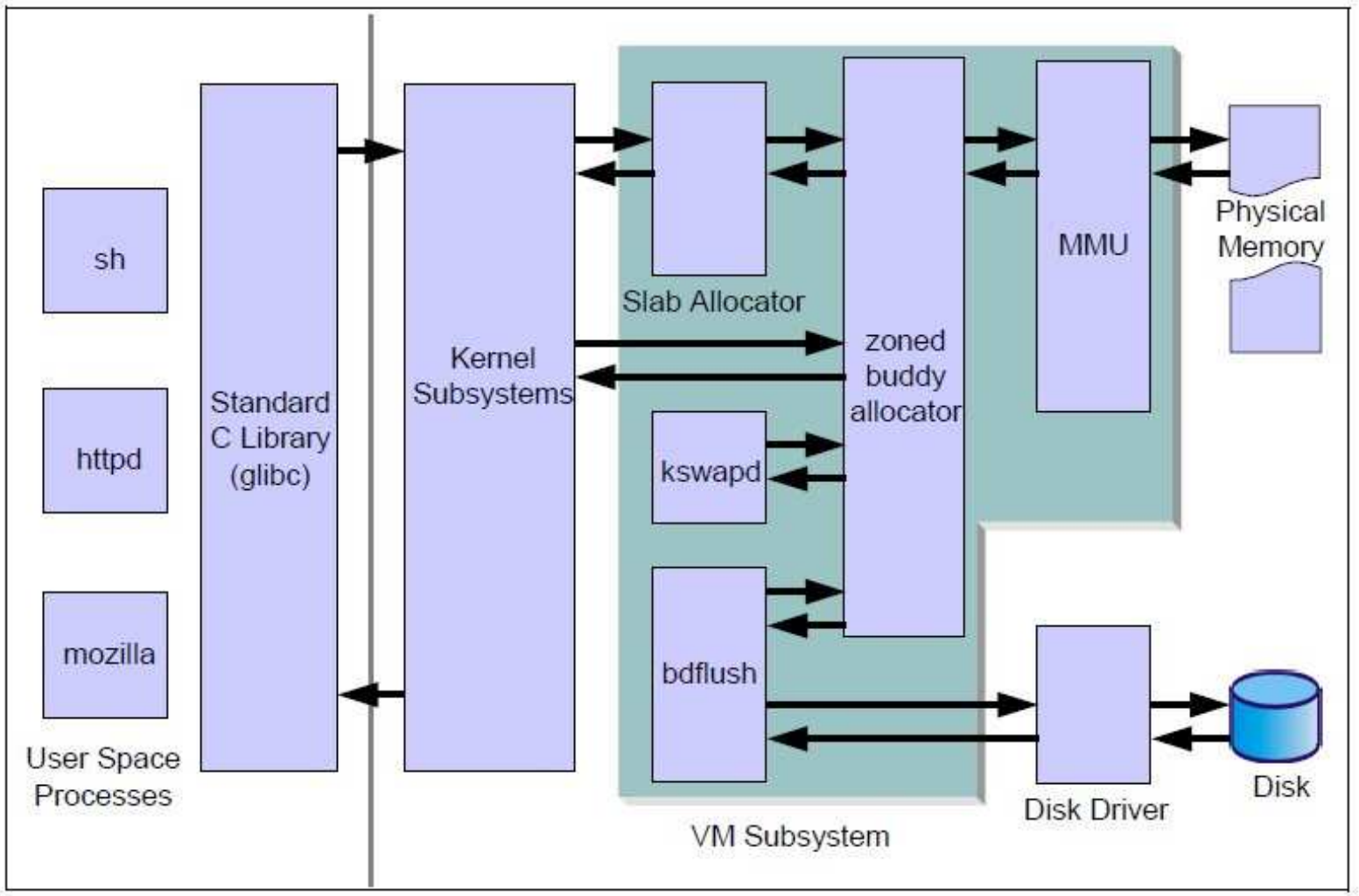
- If pages are not available when a process requests them the kernel tries to get pages for the new request by releasing certain pages (which were used before but are not used anymore and are still marked as active pages) and allocating the memory to a new process.
- This is called *page reclaiming* and is the job of `kswapd`.
- `kswapd` is usually sleeping in task interruptible state.
- It is called by the buddy system when free pages in a zone fall short of a threshold.
- It uses a Least Recently Used (LRU) algorithm for finding replaceable candidates.
- An active list and the inactive list are used to maintain candidate pages.
- `kswapd` scans part of the active list - pages not used recently are put into the inactive list.
- Check `vmstat -a`
- *page cache*: pages mapped to files on disk
  process address space: memory for heap and stack (anonymous)
  `kswapd` shrinks the page cache before swapping process address space

Virtual Memory Manager



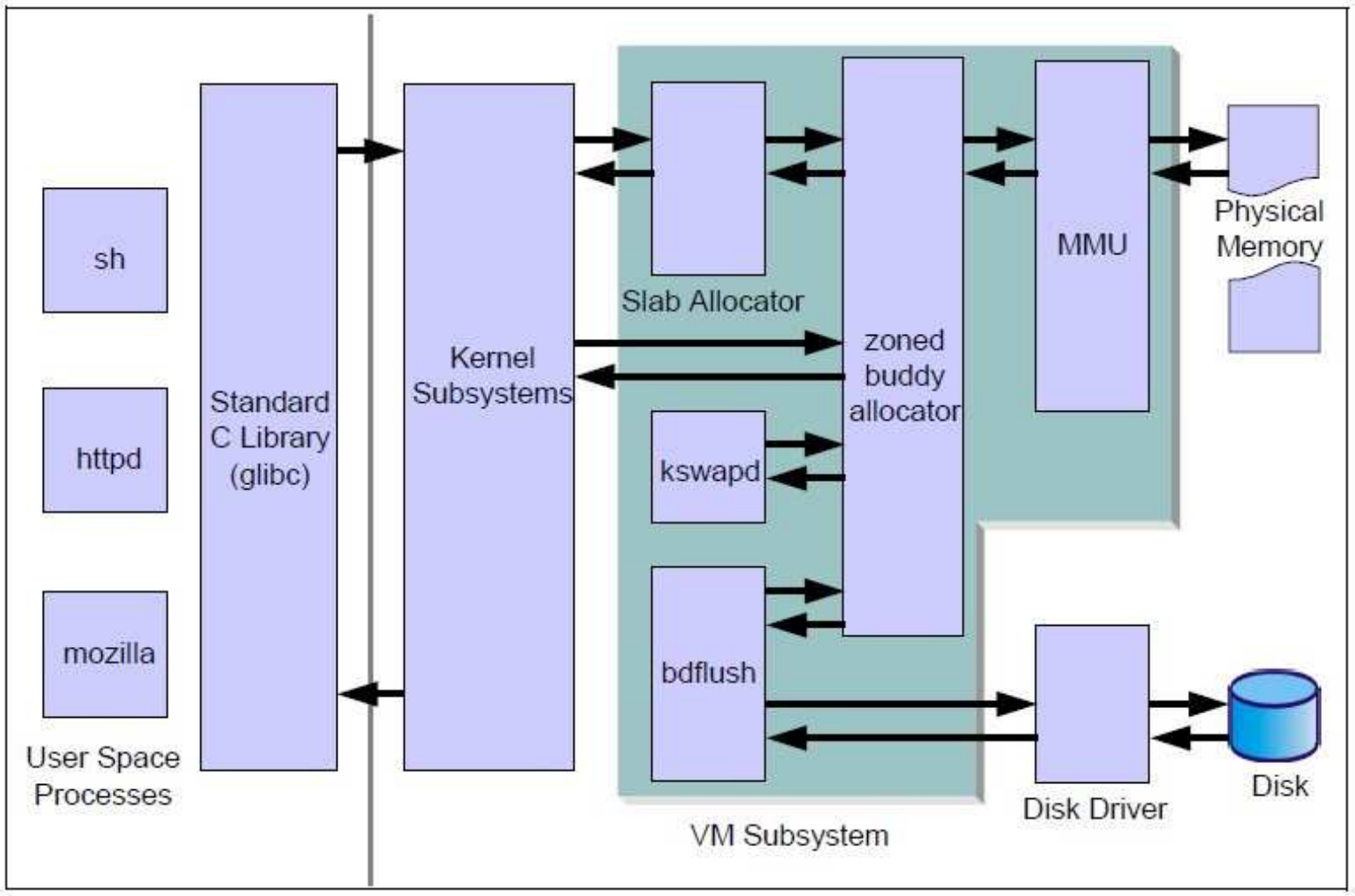
- Zone watermarks help track the pressure a zone is under.
  `pages_low`: buddy allocator wakes up `kswapd`
  `pages_min`: `kswapd` works synchronously
  `pages_high`: system sleeps until this number can be flushed to disk

Virtual Memory Manager



- pages in memory and their dups on disk are kept in sync via `bdflush` - a flush occurs when a threshold for dirty pages is reached, or periodically.
- slabs pre-allocate pages to objects of known size

Virtual Memory Manager



- The Memory Management Unit (MMU) translates virtual page numbers to physical page numbers via an associative cache called a Translation Look-aside Buffer (TLB).

Virtual Memory

- `/proc/buddyinfo`: Each column shows the number of pages of that order which are available

```
Node 0, zone DMA      27   25   21   10   5   3  3  3  1  0 0
Node 0, zone Normal 5857 9820 5450 1468 453 222 92 48 32 27 3
Node 0, zone HighMem 313  225  165   44  14   4  3  2  2  2 0
```

  In the normal zone there are 5857 chunks of unit size, 9820 chunks of size 2, 5450 chunks of size 4, and so on that are free.

- `vmstat`: prints vm statistics, average or sampled

```
 procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu---
--
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 130532 427732  81820 677768    0    2   113   113  373  449 17  7 74  1  0
```

  `r:` waiting to run, `b:` uninterruptible sleep, `swpd:` vm used (kB),
  `free:` free memory (kB), `buff:` mem used for buffers (kB),
  `cache:` mem used in cache (kB),
  `si/so:` mem swapped in/out from disk (kB/sec),
  `bi/bo:` blocks sent to/received from a device (#/sec),
  `in:` interrupts per sec, `cs:` context switches per sec,
  `us:` time spent on user procs, `sy:` time spent on kernel procs,
  `id:` time spent idle, `wa:` time spent waiting for I/O.

Virtual Memory

- `pmap`: Report how much memory processes are using

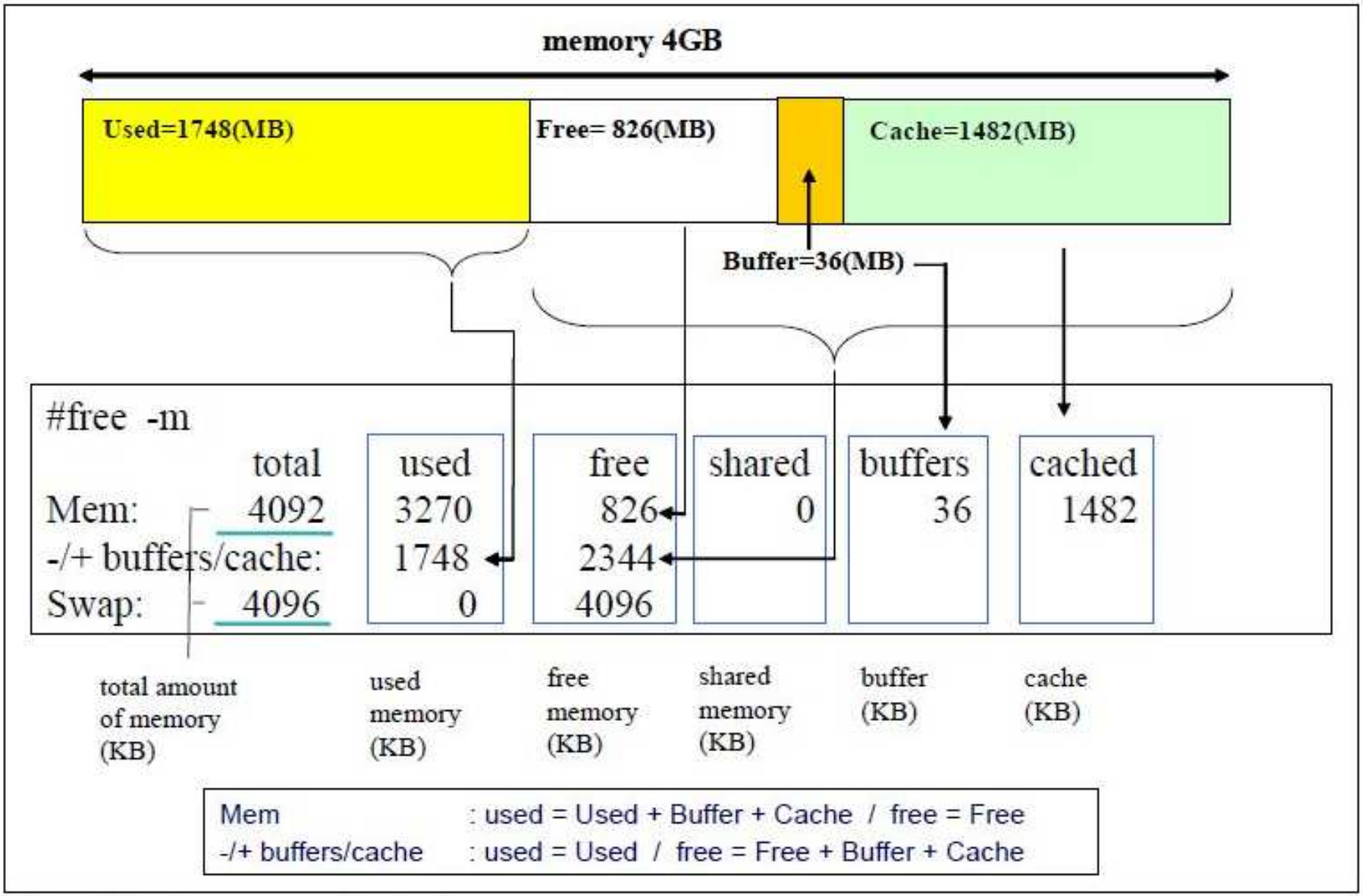
```
pmap -x 14327
14327:   C/address_space 4
Address   Kbytes     RSS   Dirty Mode   Mapping
08048000       4       4       0 r-x--  address_space
08049000       4       4       4 rw---  address_space
094f8000     132       4       4 rw---    [ anon ]
43bbf000     124     104       0 r-x--  ld-2.15.so
43bde000       4       4       4 r----  ld-2.15.so
43bdf000       4       4       4 rw---  ld-2.15.so
43be2000    1708     276       0 r-x--  libc-2.15.so
43d8d000       4       0       0 -----  libc-2.15.so
43d8e000       8       8       8 r----  libc-2.15.so
43d90000       4       4       4 rw---  libc-2.15.so
43d91000      12       8       8 rw---    [ anon ]
b77a5000      16      12      12 rw---    [ anon ]
b77a9000       4       4       0 r-x--    [ anon ]
bfa37000     132       8       8 rw---    [ stack ]
-------- ------- ------- ------- -------
total kB    2160       -       -       -
mapped: 2160K    writeable/private: 304K    shared: 0K
```

Virtual Memory

- `free`: Report amount of free and used memory

```
prompt> free -l
             total       used       free     shared    buffers     cached
Mem:       1935544    1578848     356696          0      84700     700492
Low:        875376     547052     328324
High:      1060168    1031796      28372
-/+ buffers/cache:     793656    1141888
Swap:      3899388     130584    3768804
```

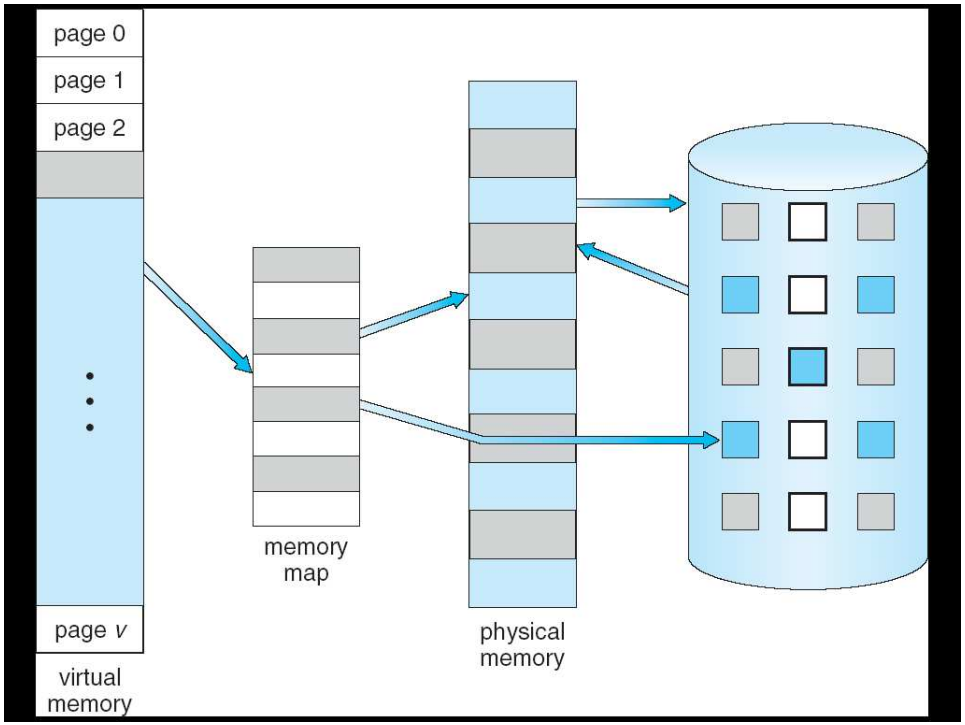
Virtual Memory



Virtual Memory

- `/proc/sys/vm`: Files for control and analysis of virtual memory
  `swappiness:` defines how aggressively memory pages are swapped to disk. Linux moves memory pages that have not been accessed for some time to the swap space even if there is enough free memory available.
- `/usr/bin/time -v date`: show system page size, page faults, etc of a process during execution.
- `size simple`: list the section sizes, and total size, for each of the object or archive files objfile in its argument list.

Virtual Memory



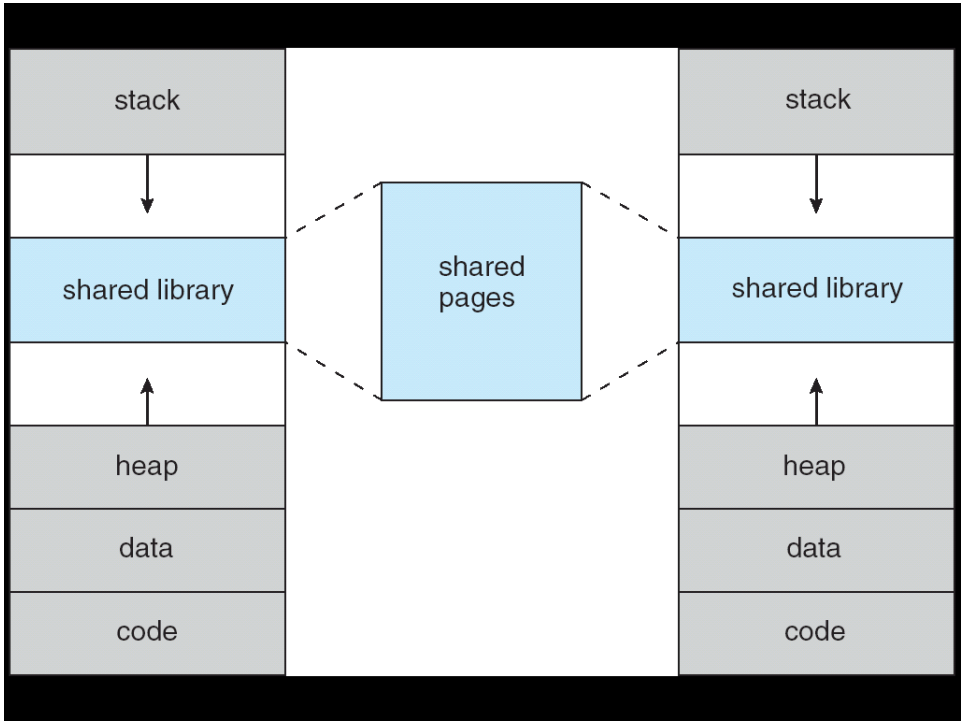
An overview of virtual memory - makes sense because:

1. A program can be run from other than main memory
2. Arrays are often over-sized for worst-case scenarios
3. Certain features of certain programs are rarely used

Benefits of VM include:

1. Programs can be written for more RAM than is available
2. CPU utilization and throughput is improved because more memory appears to be available to many processes
3. Less I/O is needed for swapping in and out of RAM

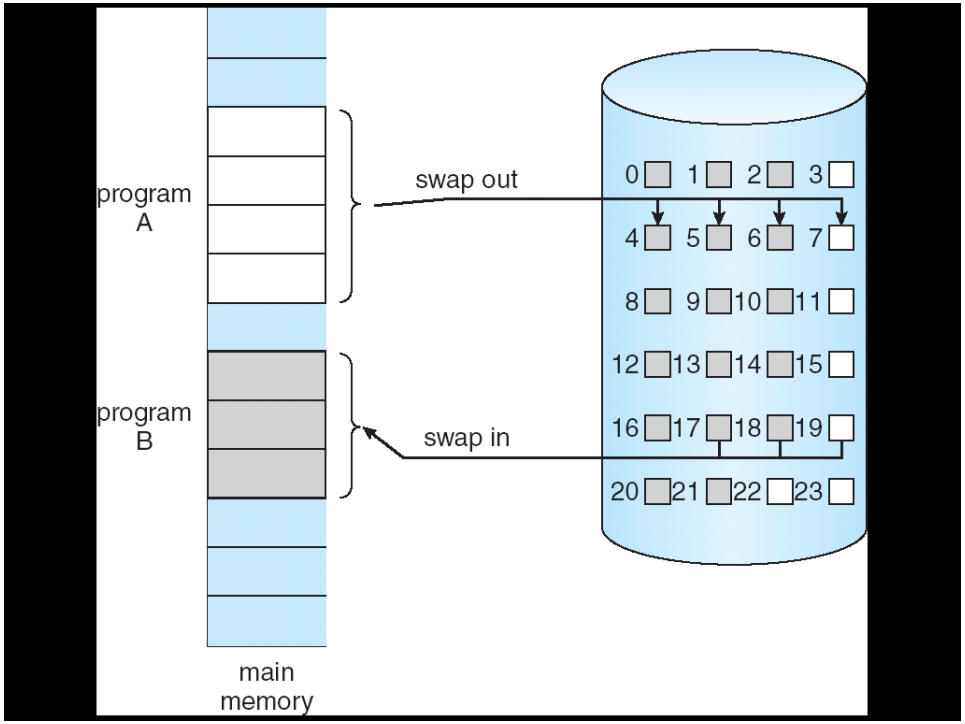
Virtual Memory



Virtual memory also allows the sharing of files and memory by multiple processes, with several benefits:

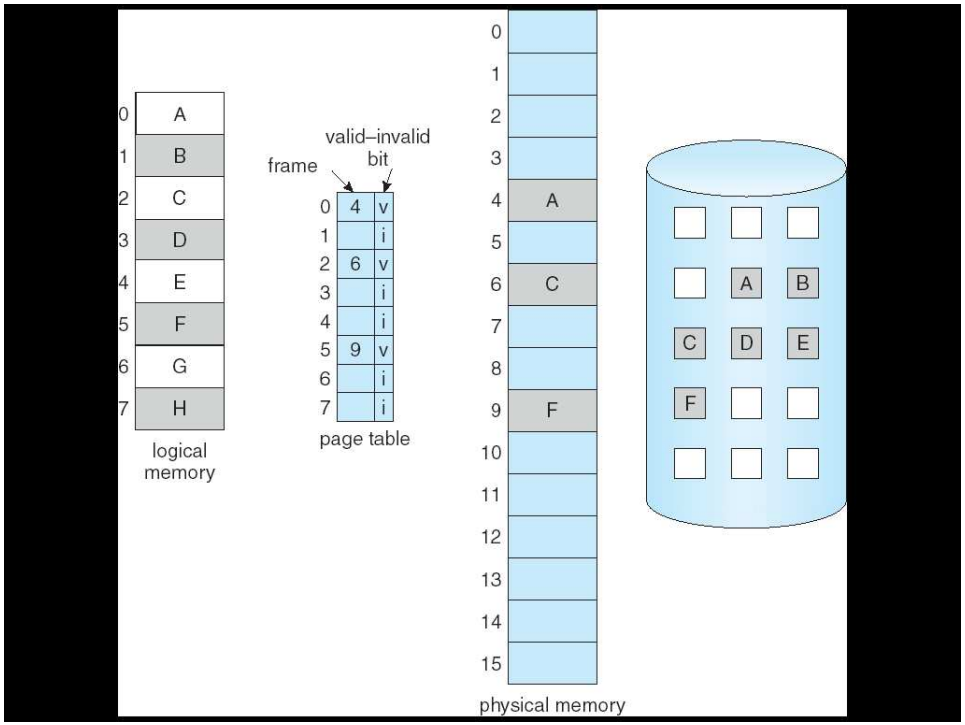
1. System libraries can be shared by mapping them into the virtual address space of more than one process
2. Processes can also share virtual memory by mapping the same block of memory to more than one process
3. Process pages can be shared during a fork() system call, eliminating the need to copy all of the pages of the original (parent) process

Virtual Memory



Demand paging: when a process is swapped in from disk its pages are not swapped in all at once: only when the process needs (demands) them. This is called a lazy swapper or pager.

Virtual Memory



Basic ideas:

1. When a process is swapped in, the pager only loads into memory those pages that it expects the process to need (right away).
2. Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit. (The rest of the page table entry may either be blank or contain information about where to find the swapped-out page on the hard drive).
3. If the process only ever accesses pages that are loaded in memory (memory resident pages), then the process runs exactly as if *all* its pages are loaded in to memory.