# AN INTRODUCTION TO FREEBSD MEMORY MANAGEMENT

ISAAC RICHTER

# AN INTRODUCTION TO FREEBSD MEMORY MANAGEMENT

VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

PHYSICAL MAPPINGS

ZONE/SLAB ALLOCATOR

GENERIC RESOURCE ALLOCATOR

# VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

PHYSICAL MAPPINGS

ZONE/SLAB ALLOCATOR

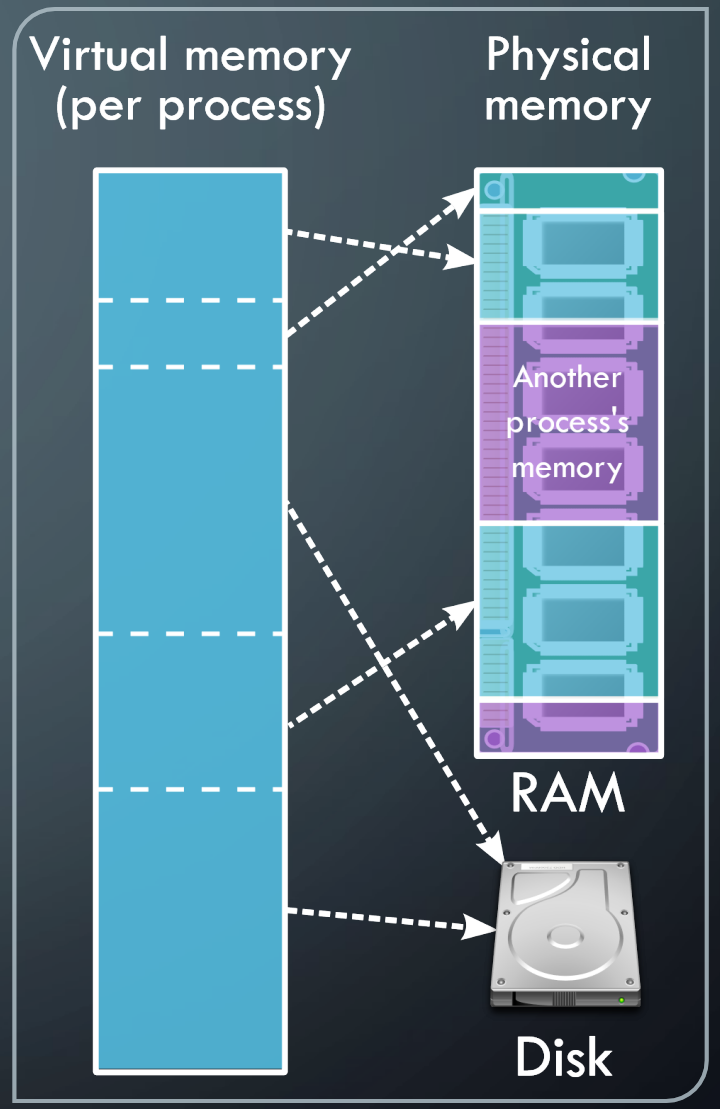
GENERIC RESOURCE ALLOCATOR

# VIRTUAL MEMORY

## Enabling paging and overlapping per-process allocations

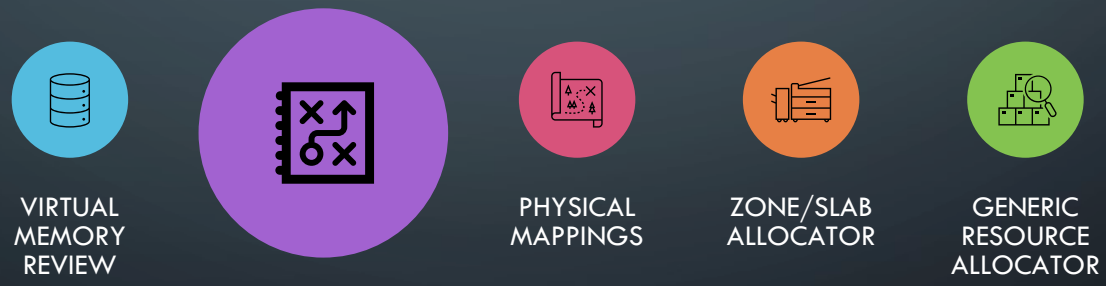
- Processes have access to arbitrary virtual addresses
- Memory-map files into virtual address space [paging]
- Allocate physical memory on-use
- Copy-on-write
- Physical fragmentation does not inhibit contiguous virtual allocations

**Kernel Task: given faulting address, what to do?**



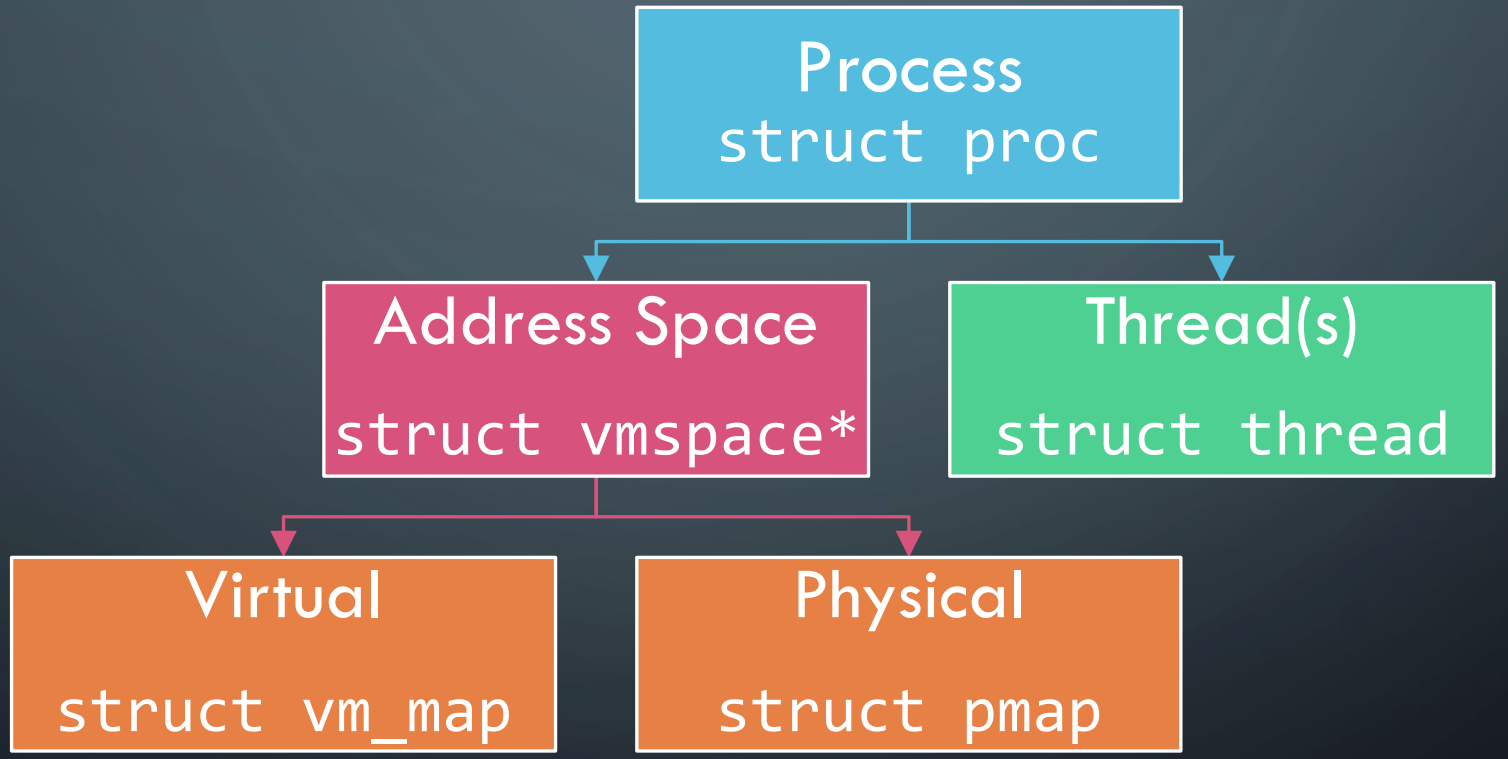

https://commons.wikimedia.org/wiki/File:Virtual_memory.svg

# PROCESS ADDRESS SPACE MANAGEMENT



# PROCESSES

Group of threads (schedulable contexts) sharing an address space



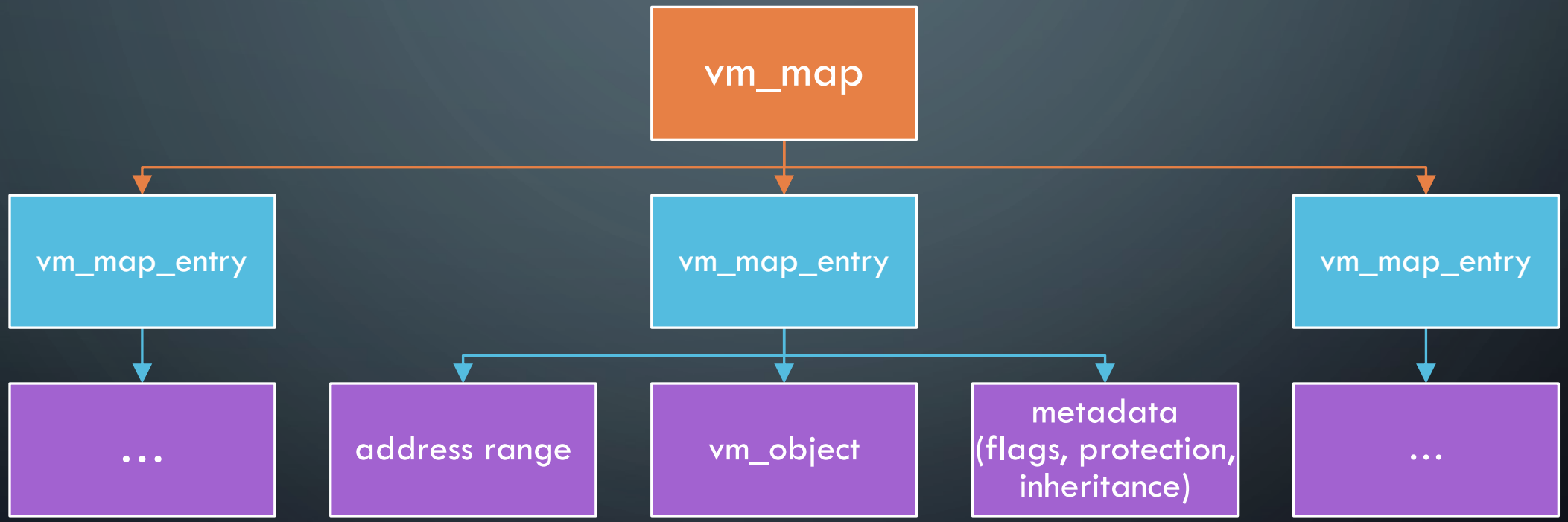
# FREEBSD PROCESSES

```
struct proc {
  pid_t                 p_pid;     /* Process identifier. */
  struct ucred          *p_ucred;  /* Process owner's identity. */
  struct filedesc       *p_fd;     /* Open files. */
  TAILQ_HEAD(, thread)              p_threads; /* all threads. */
  struct vmspace        *p_vmspace; /* Address space. */
  /* ... */
};
```

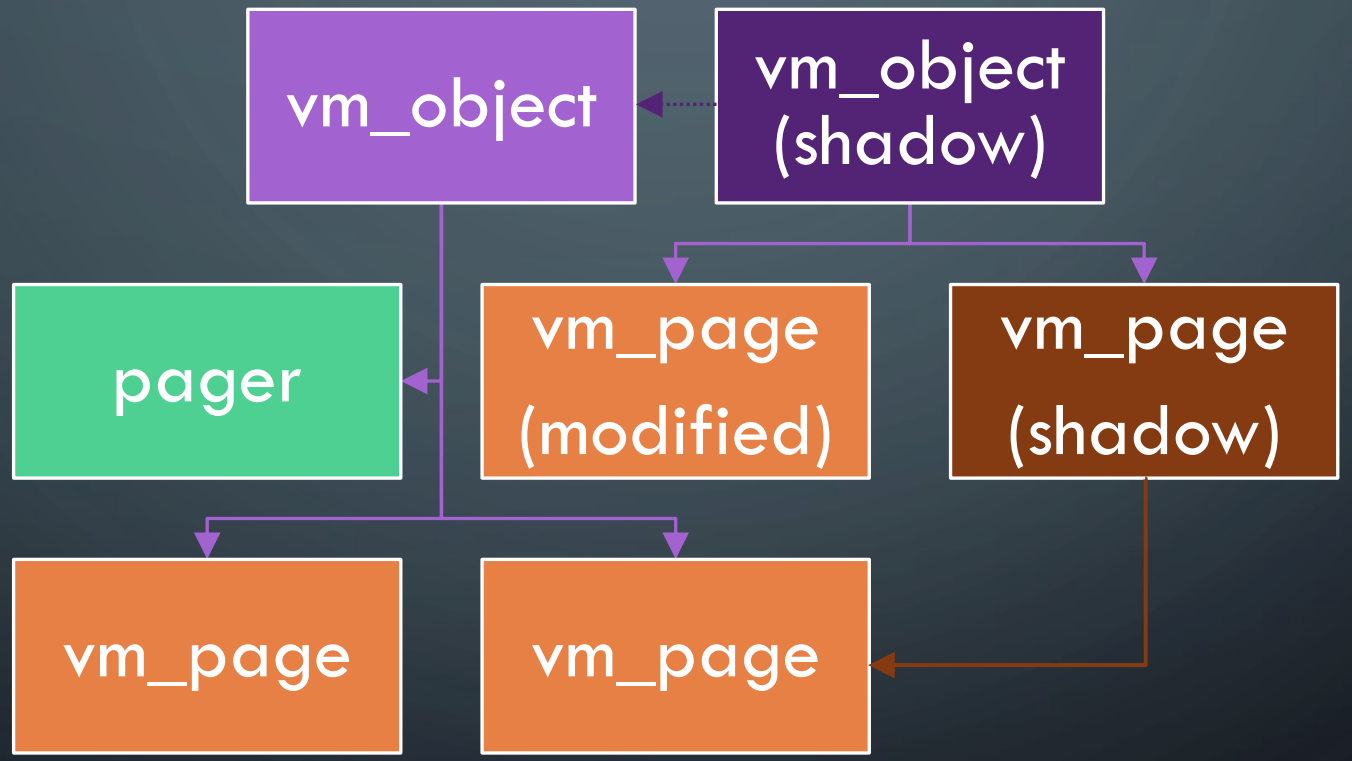
# PROCESS ADDRESS SPACE

```
/* Shareable process virtual address space. */
struct vmspace {
    struct vm_map           vm_map;    /* VM address map */
    struct shmmap_state     *vm_shm;   /* SYS5 shared memory data */
    /* accounting info */
    struct pmap             vm_pmap;   /* private physical map */
};
```

# VM MAP


vm_map
vm_map_entry
vm_map_entry
vm_map_entry
…
address range
vm_object
metadata
(flags, protection,
inheritance)
…

# VM_ OBJECT


vm_object
vm_object (shadow)
pager
vm_page (modified)
vm_page (shadow)
vm_page
vm_page

# FREEBSD PAGERS

- default (wired or not yet swapped)
- swap (anonymous, shm)
- vnode (file-system or geom-backed)
- Specialized (scatter/gather, device, …)

# VM PAGE

```
struct vm_page {
  union           plinks;    /* page queue or free list
                              * or slab/zone information */
  vm_object_t     object;    /* which object am I in */
  vm_pindex_t     pindex;    /* offset into object */
  vm_paddr_t      phys_addr; /* physical address of page */
  /* ... */
  struct md_page  md;        /* machine dependent stuff */
};
```

# PHYSICAL MAPPINGS

VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

ZONE/SLAB ALLOCATOR

GENERIC RESOURCE ALLOCATOR

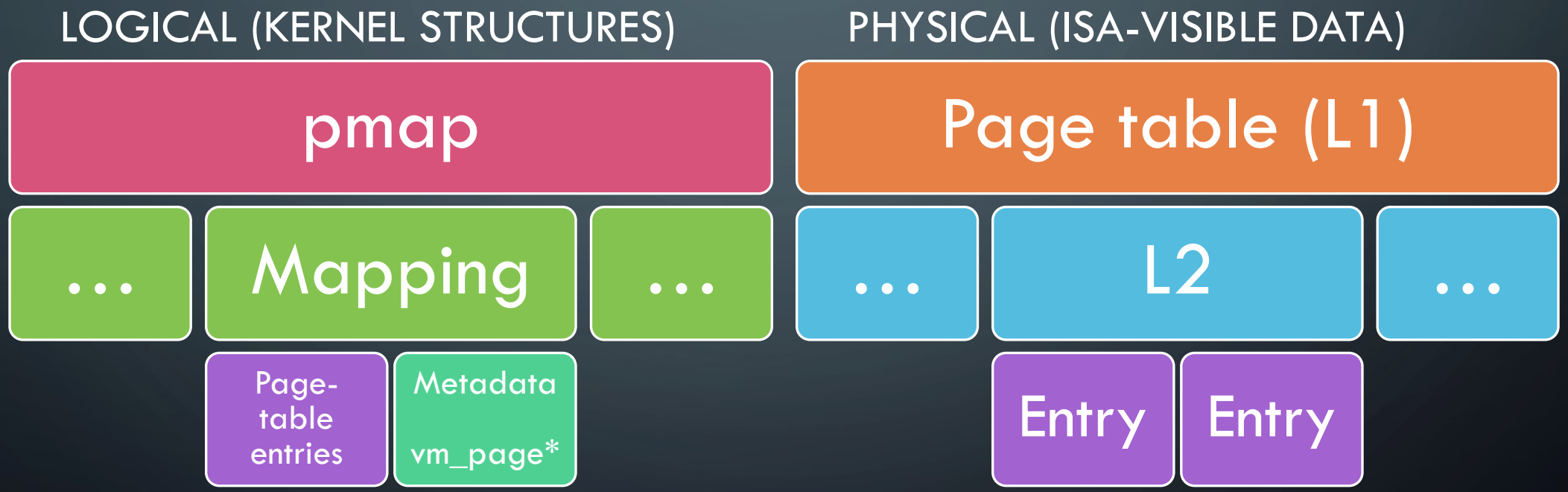
# PMAP

Architecture-independent API for managing physical mappings

```
/* Insert the given physical page (p) at
 * the specified virtual address (v) in the
 * target physical map with the protection requested. */
int   pmap_enter(pmap_t pmap, vm_offset_t va, vm_page_t m,
                 vm_prot_t prot, u_int flags,
                 int8_t psind);

/* Remove the given range of addresses from the specified
 * map. */
void  pmap_remove(pmap_t pmap, vm_offset_t sva,
                  vm_offset_t eva);
```

# PHYSICAL MAPPING
## (ARCHITECTURE-SPECIFIC)



# ZONE/SLAB ALLOCATOR

VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

PHYSICAL MAPPINGS

GENERIC RESOURCE ALLOCATOR

# ZONE/SLAB ALLOCATOR

Allocates objects of fixed size

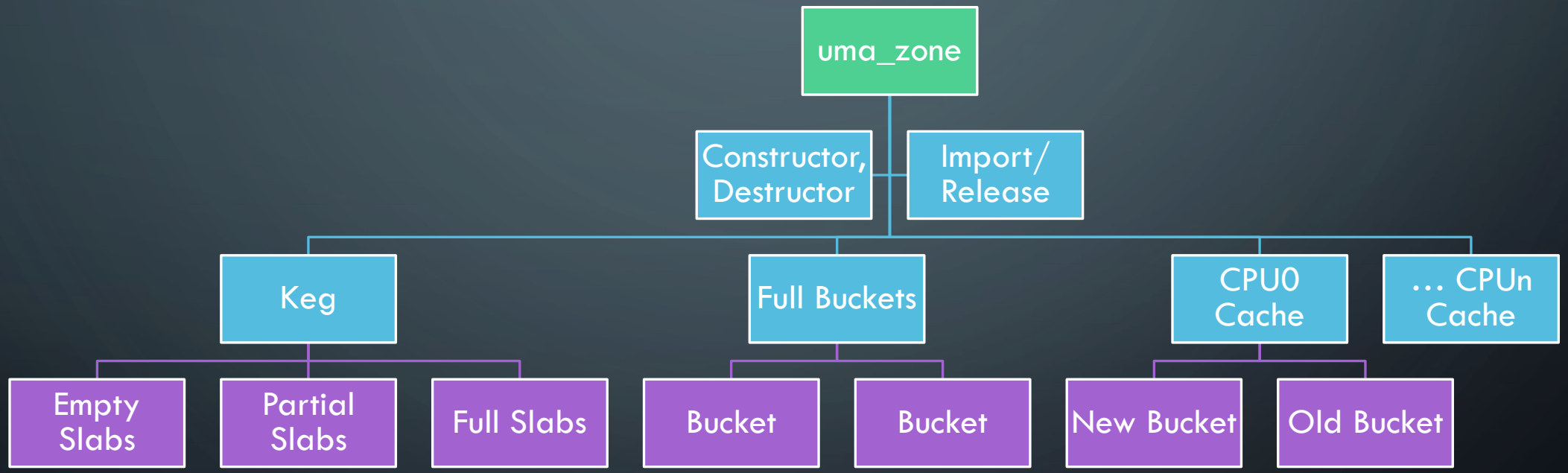
- Optional constructor/destructor
- Dynamic allocation or fixed-size cache
- Per-Zone per-CPU locking reduces global lock contention
- Used for many kernel structures
- Used by kernel malloc for < PAGE_SIZE allocations

“The slab allocator provides efficient object caching but has … limitations: its global locking doesn’t scale to many CPUs”

J. Bonwick & J. Adams, “Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources,” *Proceedings of the 2001 USENIX Annual Technical Conference*, pp. 15–34, June 2001.

```
/* Allocates an item out of a zone
 *
 * Returns: A non-null pointer to an initialized element from the zone
 */
void *uma_zalloc_arg(struct uma_zone *zone, void *arg, int flags);
```

# ZONE STRUCTURE



# SLAB HEADER

```
/* The slab structure manages a single contiguous allocation
 * from backing store and subdivides it into individually
 * allocatable items. */
struct {
  uint8_t                 *uhs_data;     /* First item */
  LIST_ENTRY(uma_slab)    us_link;       /* slabs in zone */
  uint16_t                us_freecount; /* How many are free? */
  struct noslabbits       us_free;       /* Free bitmask */
};
```

# GENERIC RESOURCE ALLOCATOR

VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

PHYSICAL MAPPINGS

ZONE/SLAB ALLOCATOR

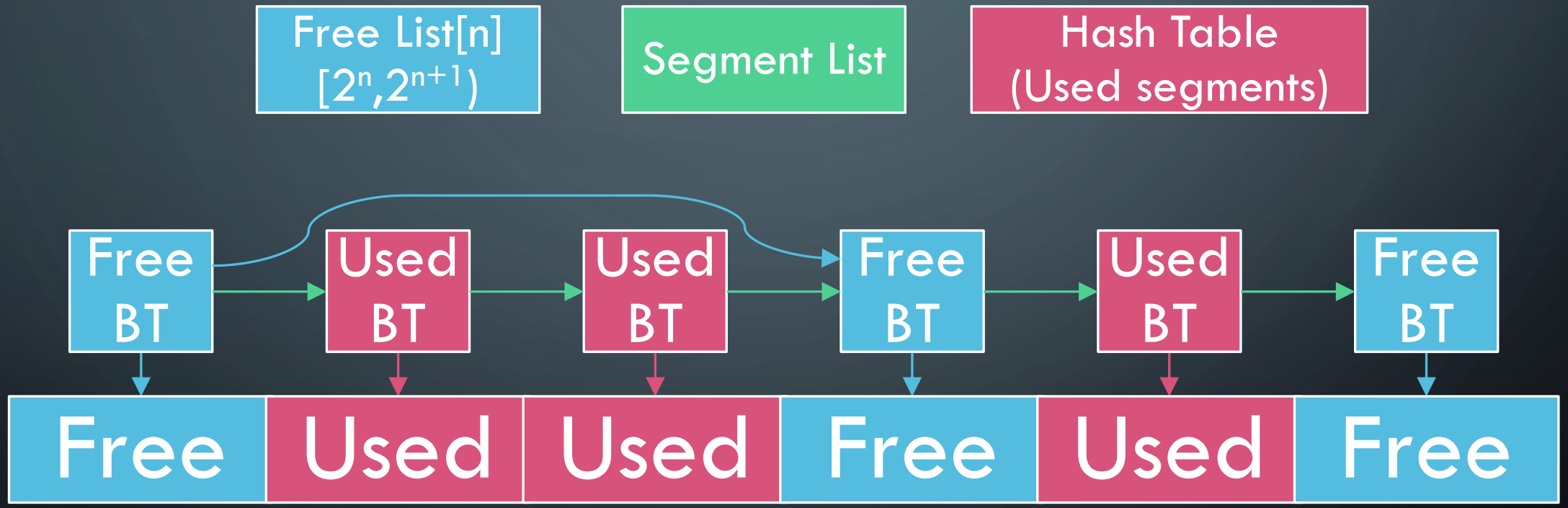
# GENERIC RESOURCE ALLOCATOR

vmem – returns contiguous chunks of a resource in constant time

- Find available items
- Usable for any allocated resource
- Arena can be expanded
- Used for kernel malloc of ≥ PAGE_SIZE

# VMEM

J. Bonwick & J. Adams, "Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources," *Proceedings of the 2001 USENIX Annual Technical Conference*, pp. 15–34, June 2001.



# QUESTIONS?

VIRTUAL MEMORY REVIEW

PROCESS ADDRESS SPACE MANAGEMENT

PHYSICAL MAPPINGS

ZONE/SLAB ALLOCATOR

GENERIC RESOURCE ALLOCATOR

# REFERENCES

McKusick, Marshall Kirk, George V. Neville-Neil, and Robert NM Watson. *The design and implementation of the FreeBSD operating system*. Pearson Education, 2014.

J. Bonwick & J. Adams, "Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources," *Proceedings of the 2001 USENIX Annual Technical Conference*, pp. 15–34, June 2001.