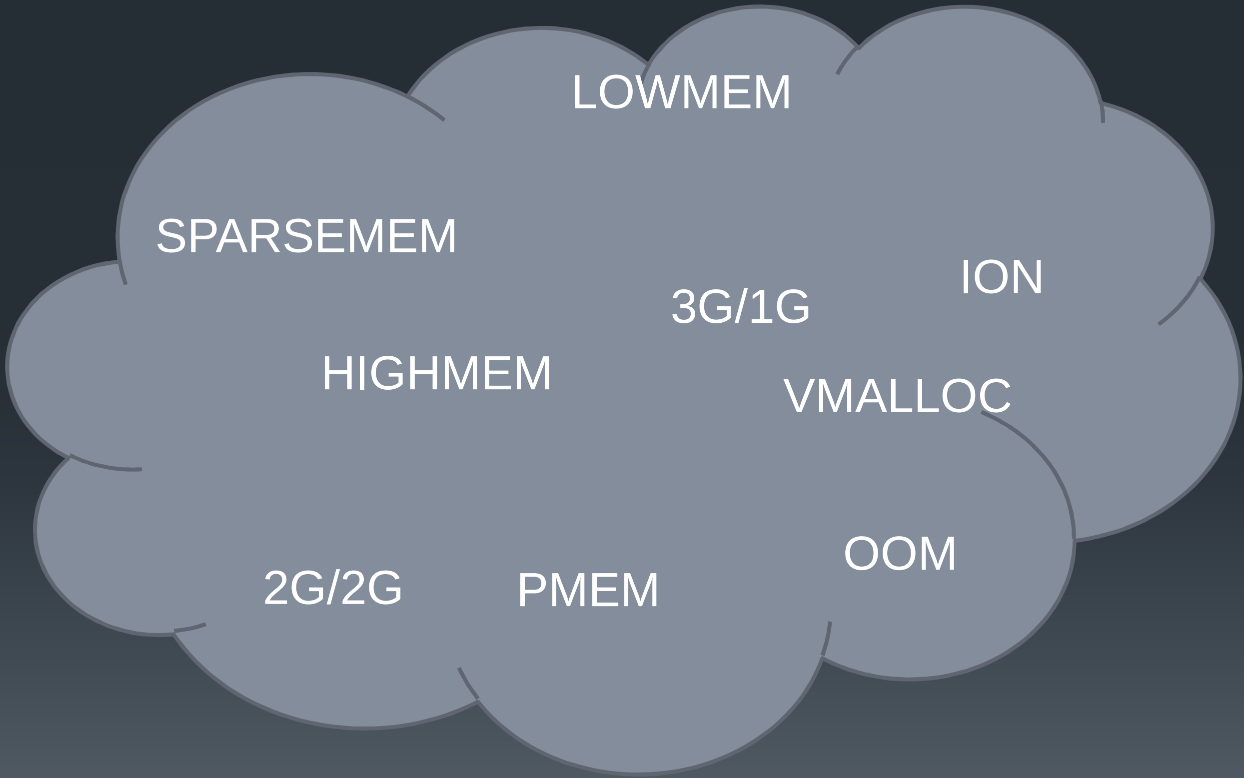
Kernel Memory Management
An Overview

Naveen Ramaraj

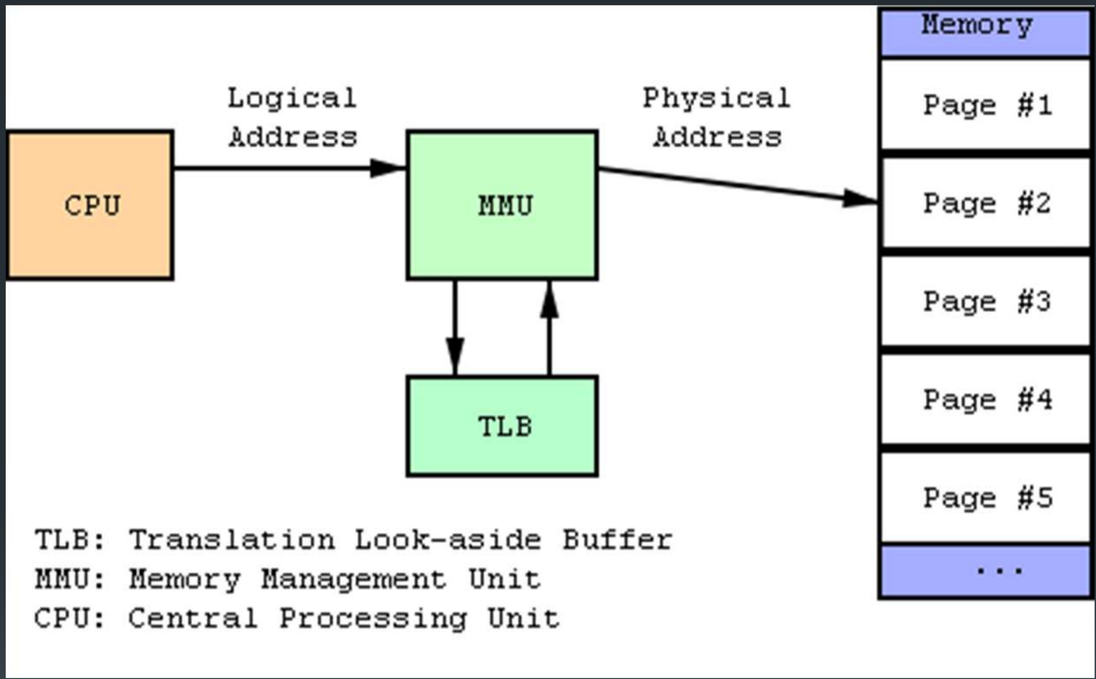
Goal: Demystify the Jargons



The Golden Rule

“All problems in computer science can be solved by another level of indirection – David Wheeler

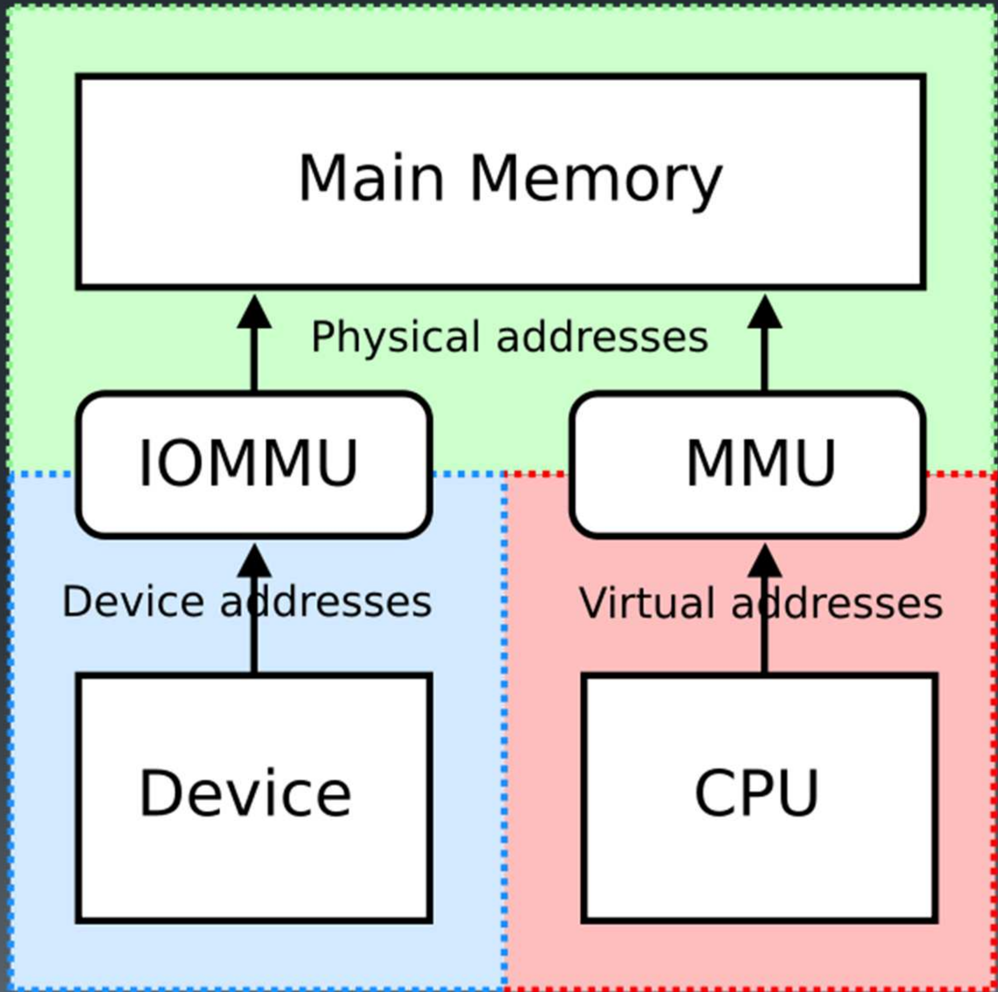
Indirections in Hardware: MMU



More Indirections in Hardware

Q: What's about IOMMU aka SMMU ?

Indirections in Hardware

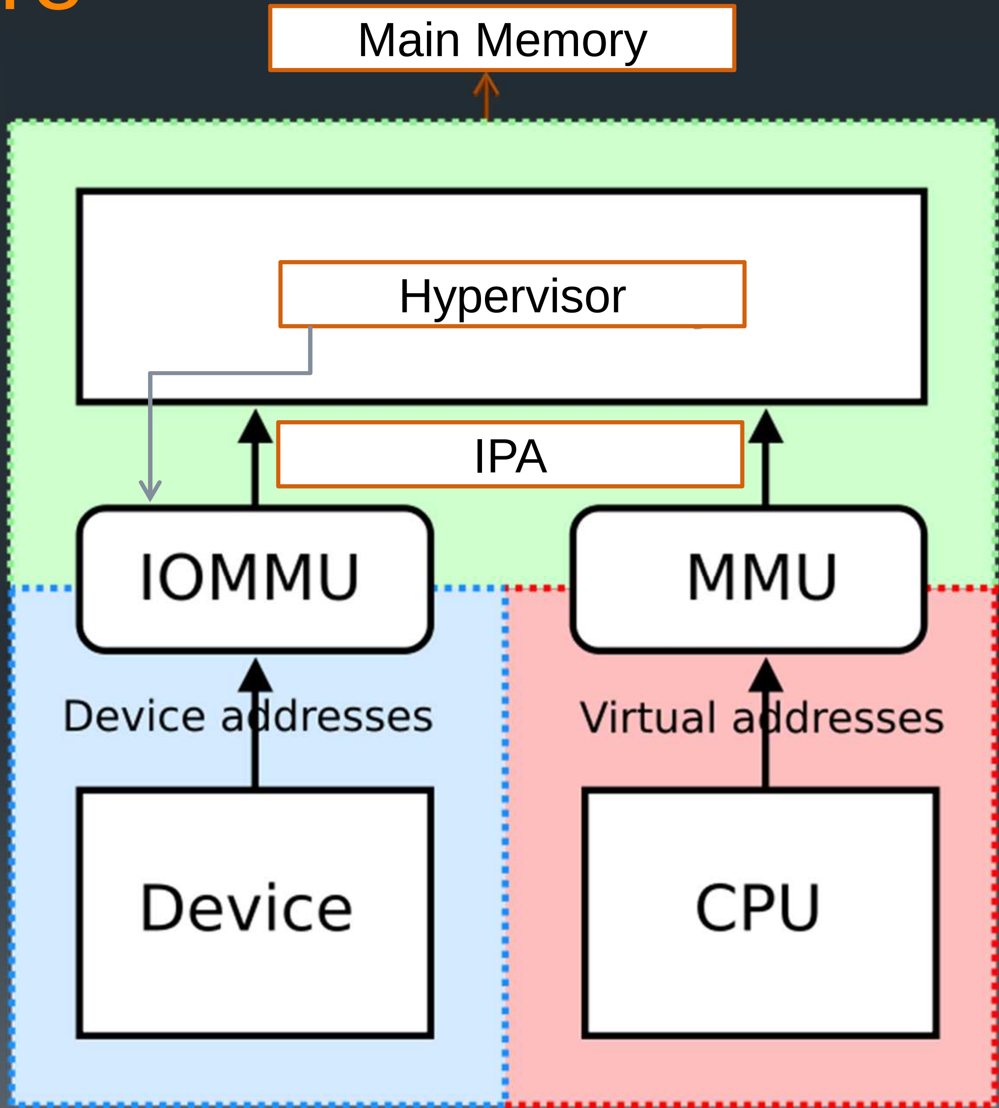


Indirections in Hardware: IOMMU

Yes, but not limited to eliminating PMEM

- Expands Device Address Space to entire physical memory
- Device Address Space is contiguous
- Memory Protection: Device can only read or write to addresses that are mapped
- Translate to an IPA for Virtualization
  - Helps Guest OSes to make use of DMA like Host H/W by handling the remapping

Indirections in Hardware + Software



The Golden Rule

"*All problems in computer science can be solved by another level of indirection*……Except for the problem of too many layers of indirection – David Wheeler

Example: Aliased Mappings

“multiple mappings of the same physical address region with differing memory type (strongly ordered, device, normal memory) becomes unpredictable” – RMK

- ARM architecture has several correctness requirements
- ioremap() of System RAM is prohibited
- Kernel Maps all System RAM as “normal with WB cache”
- Caution is required to not break correctness
- Forces the memory for such purposes to be set aside from System Memory

Kernel Memory Management

Kernel Memory Management

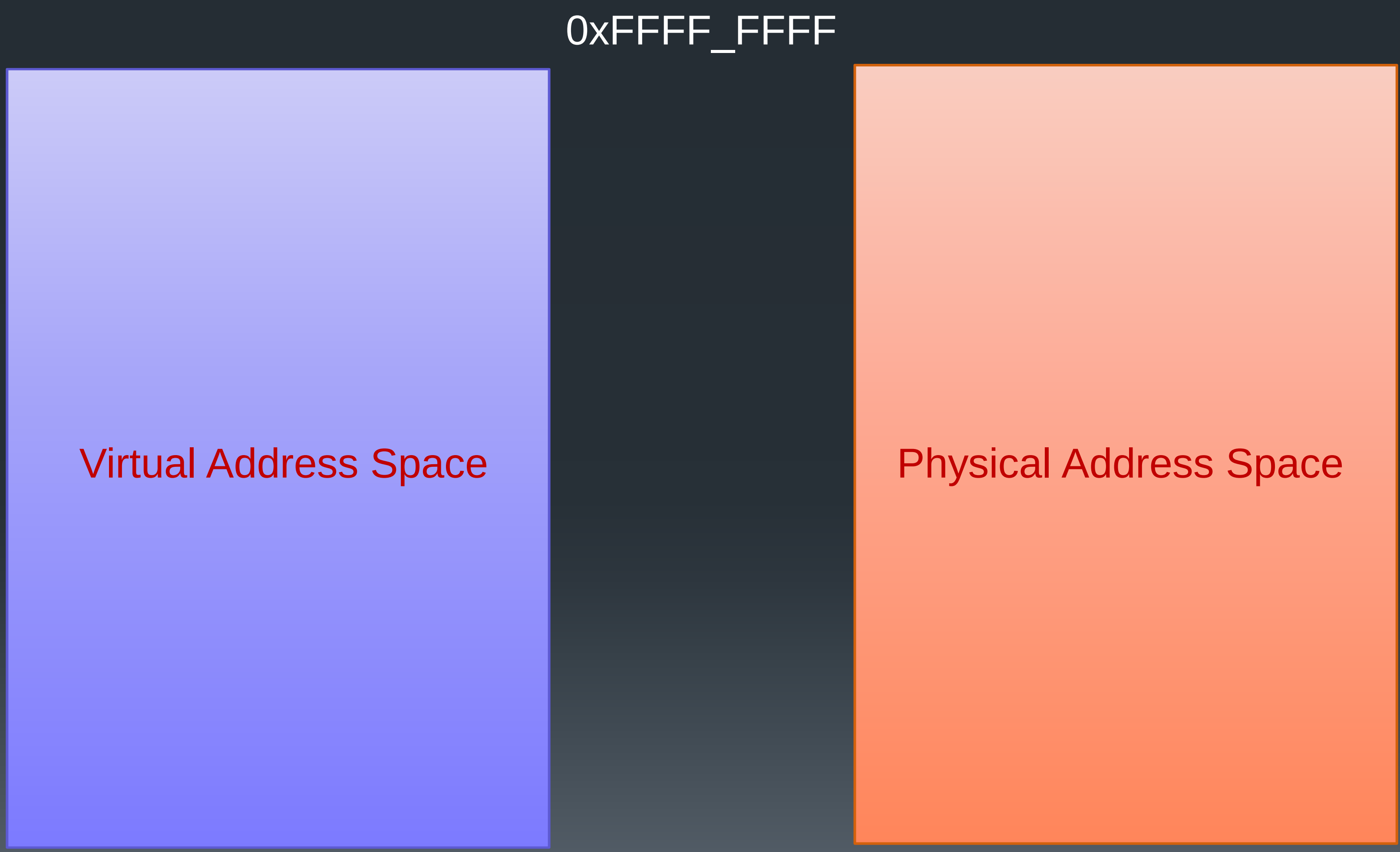
“On Mon, 25 Jan 1999, Alan Cox wrote:
>
> Oh good, whats the configuration setting for a 4Gig Xeon box. I've got
> people dying to know. So I'm not full of it.

Oh, the answer is very simple: it's not going to happen. EVER.
….
This is not negotiable.

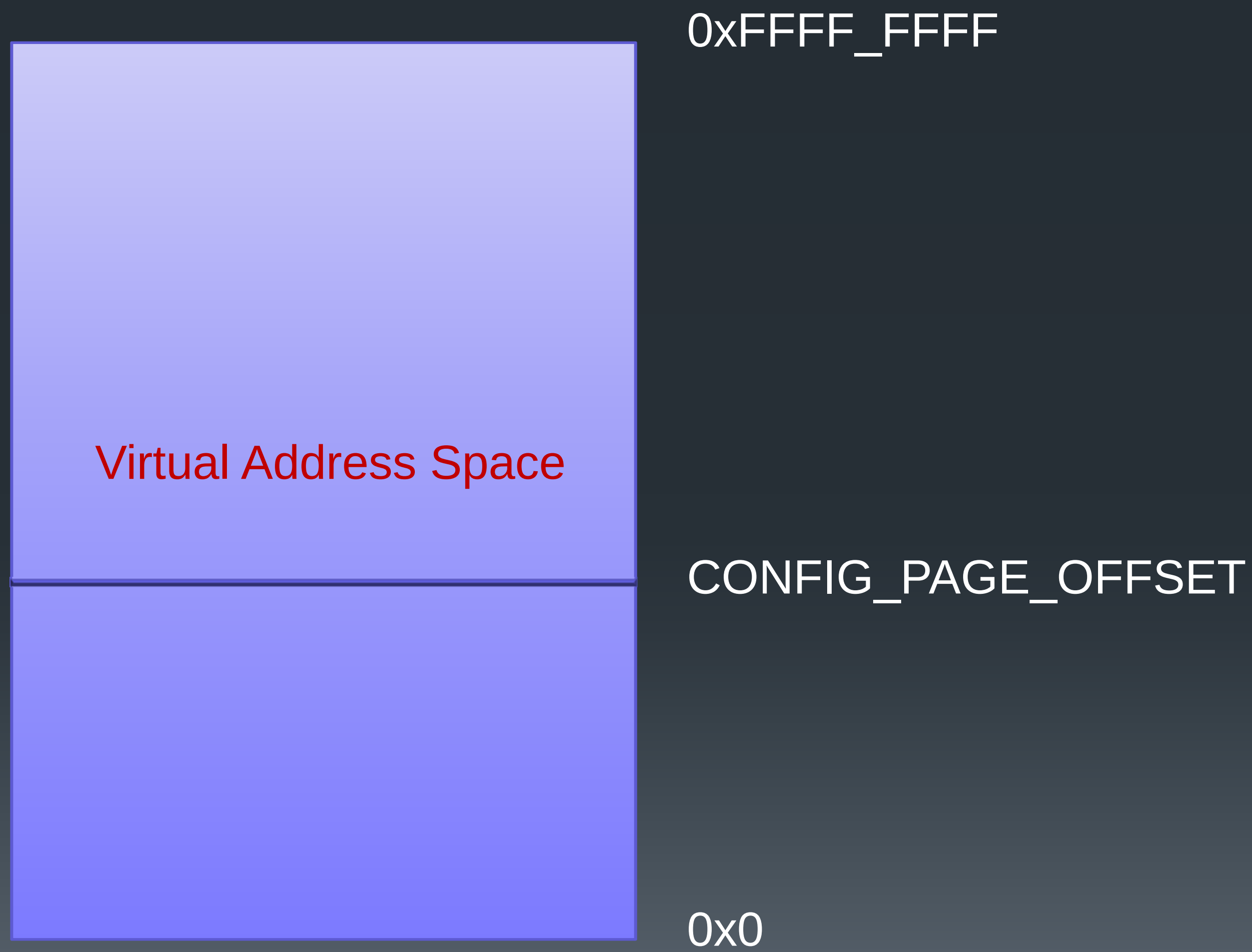
Linus”

http://lwn.net/1999/0128/a/lt-never.html

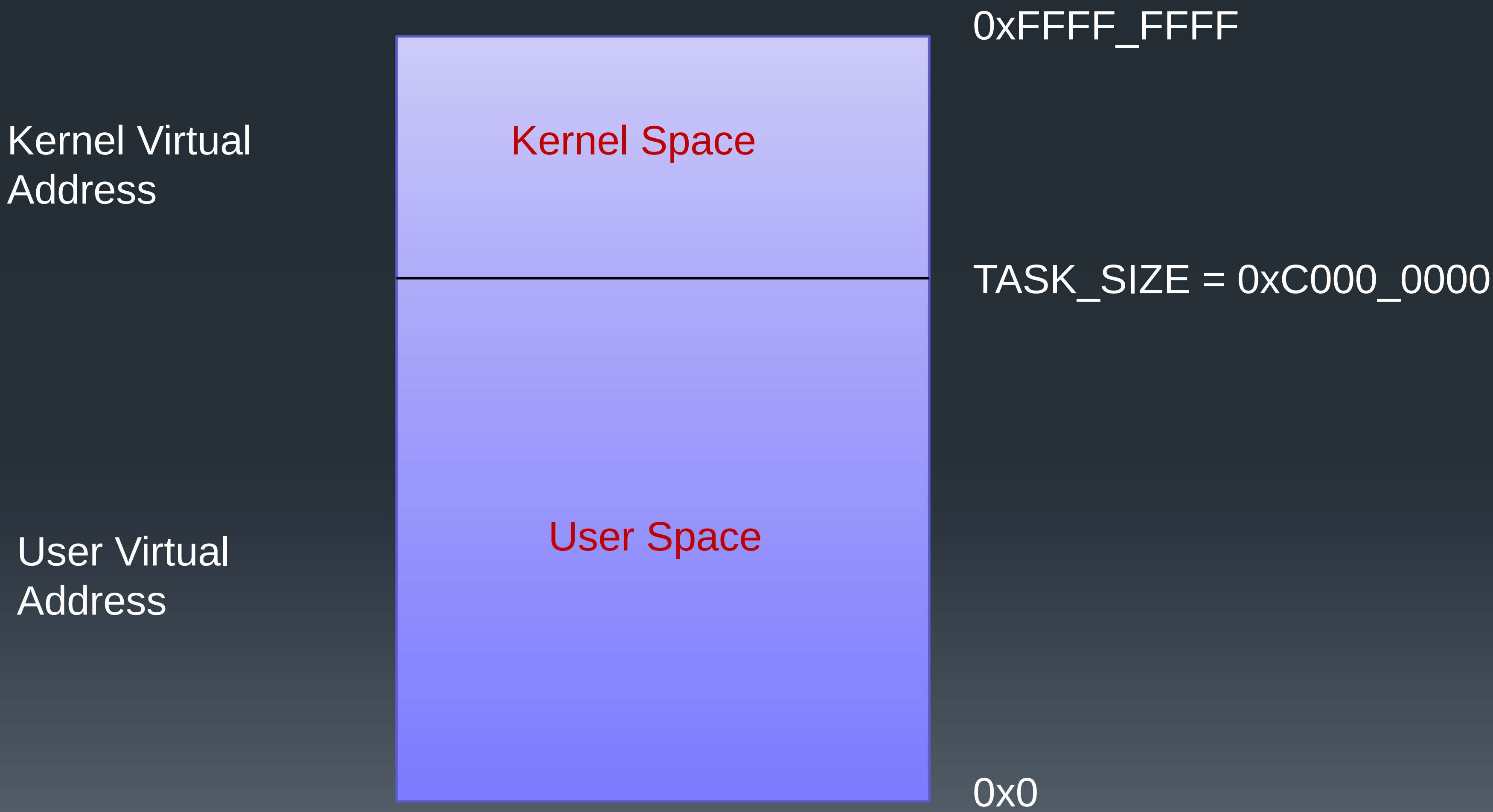
Address Spaces – 32 bit



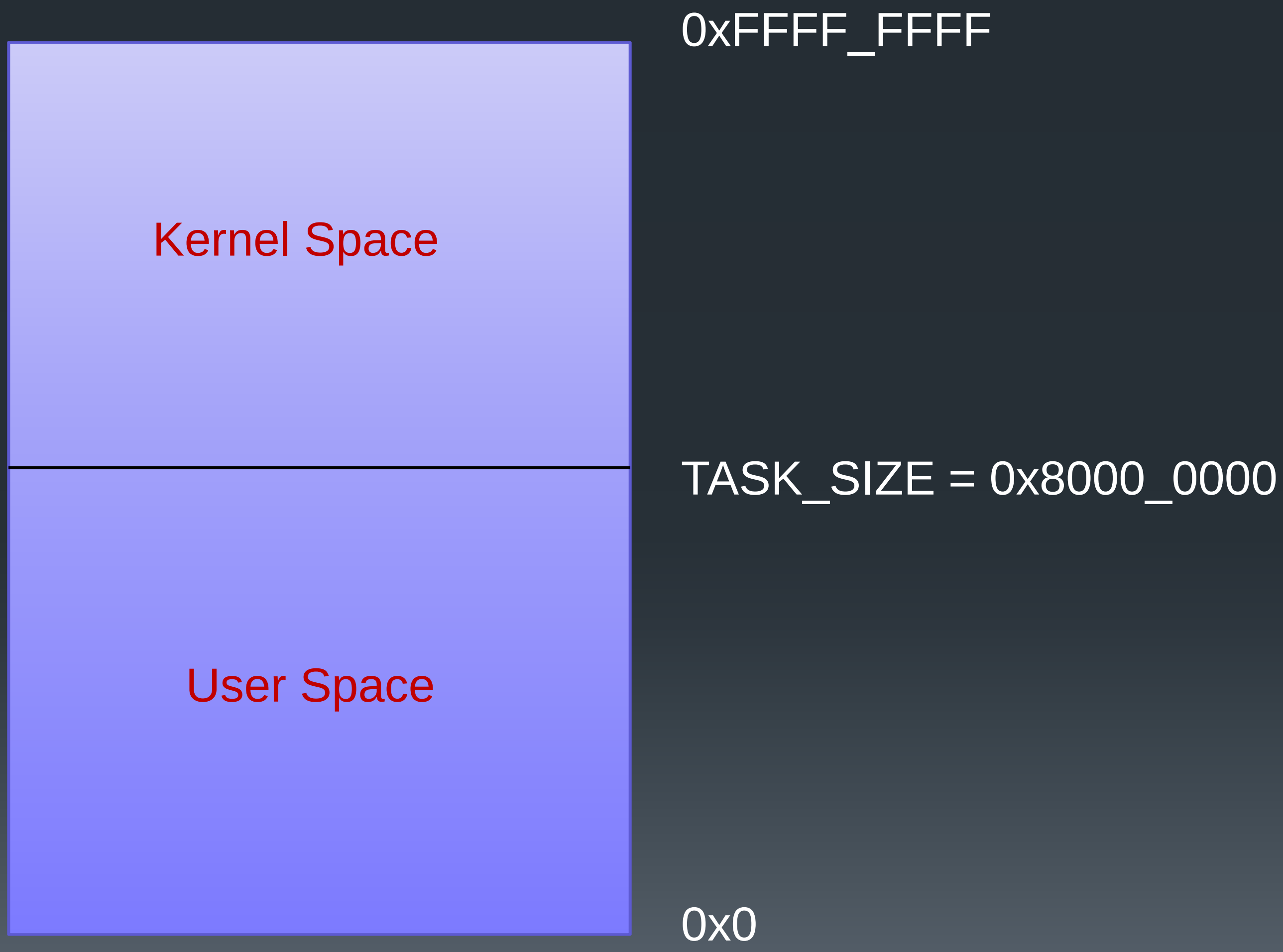
Virtual Address Space – 32 bit CPU



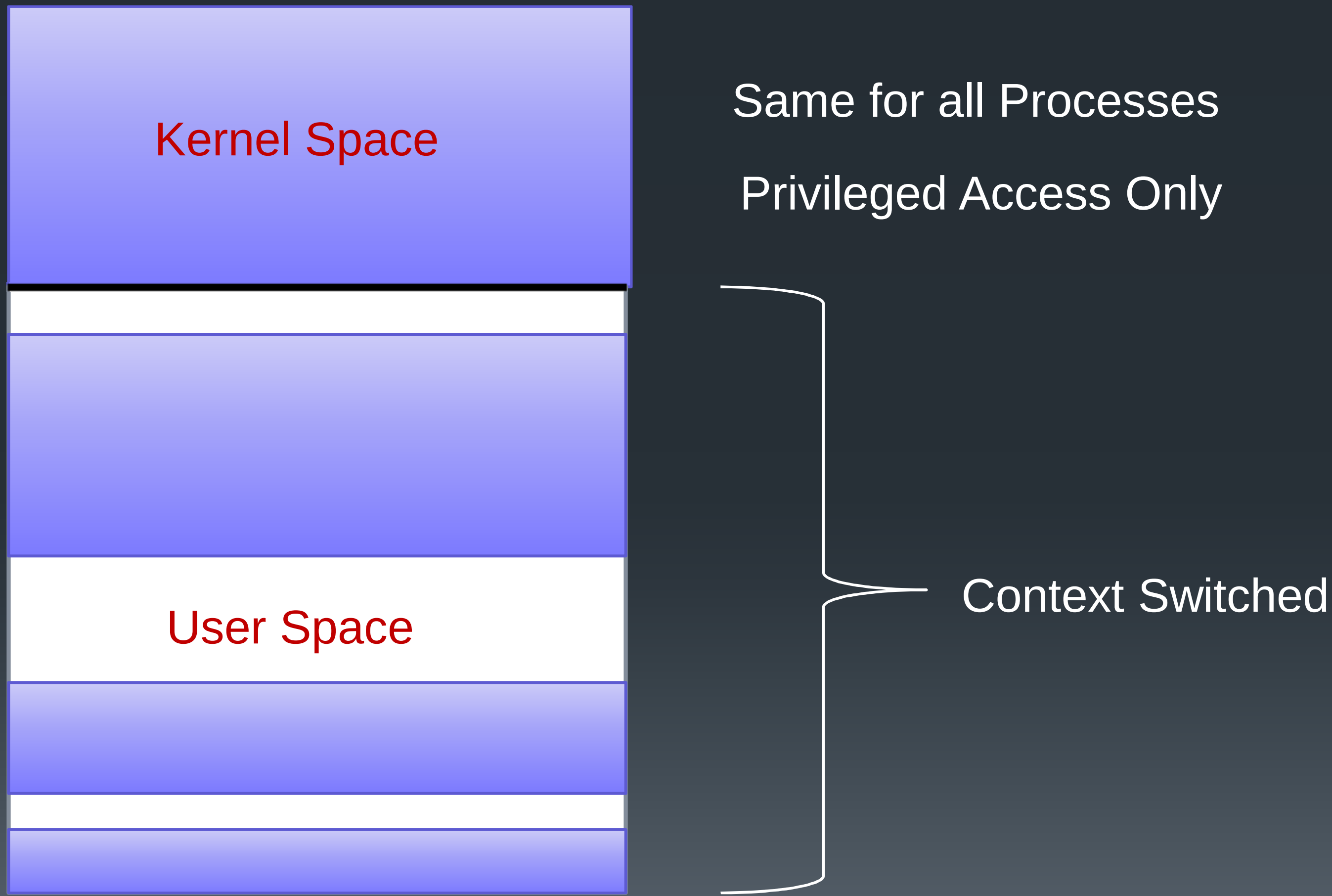
3G/1G


0xFFFF_FFFF
Kernel Virtual Address
Kernel Space
TASK_SIZE = 0xC000_0000
User Virtual Address
User Space
0x0

2G/2G


0xFFFF_FFFF
Kernel Space
TASK_SIZE = 0x8000_0000
User Space
0x0

Reality


Kernel Space
User Space
Same for all Processes
Privileged Access Only
Context Switched

VM_SPLIT 3G

- Presents the kernel with 1G virtual space to map all the memory it can access

- Hey why not use 4G/4G ?
  - Seems like Win-Win
  - Need to flush TLB’s and entire D-Cache and I-Cache (for Virtually Tagged architectures).
  - Costly Context Switch to Kernel Mode
    - E.g. System calls etc

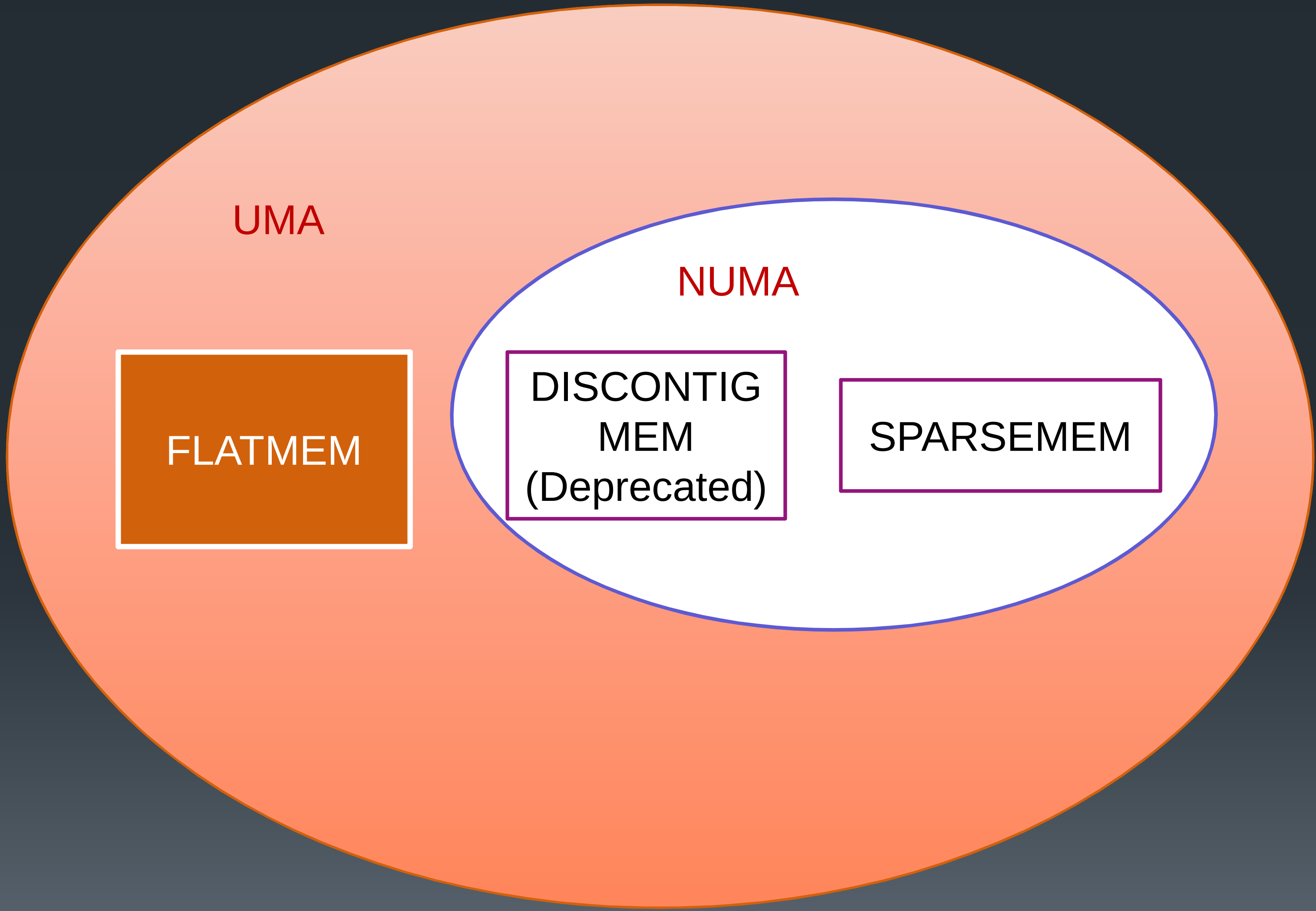
- Is it easy to switch the VM_SPLIT ?
  - Involves User Space as well, not advisable
  - Customers want more memory for apps

Virtual Kernel Memory Layout

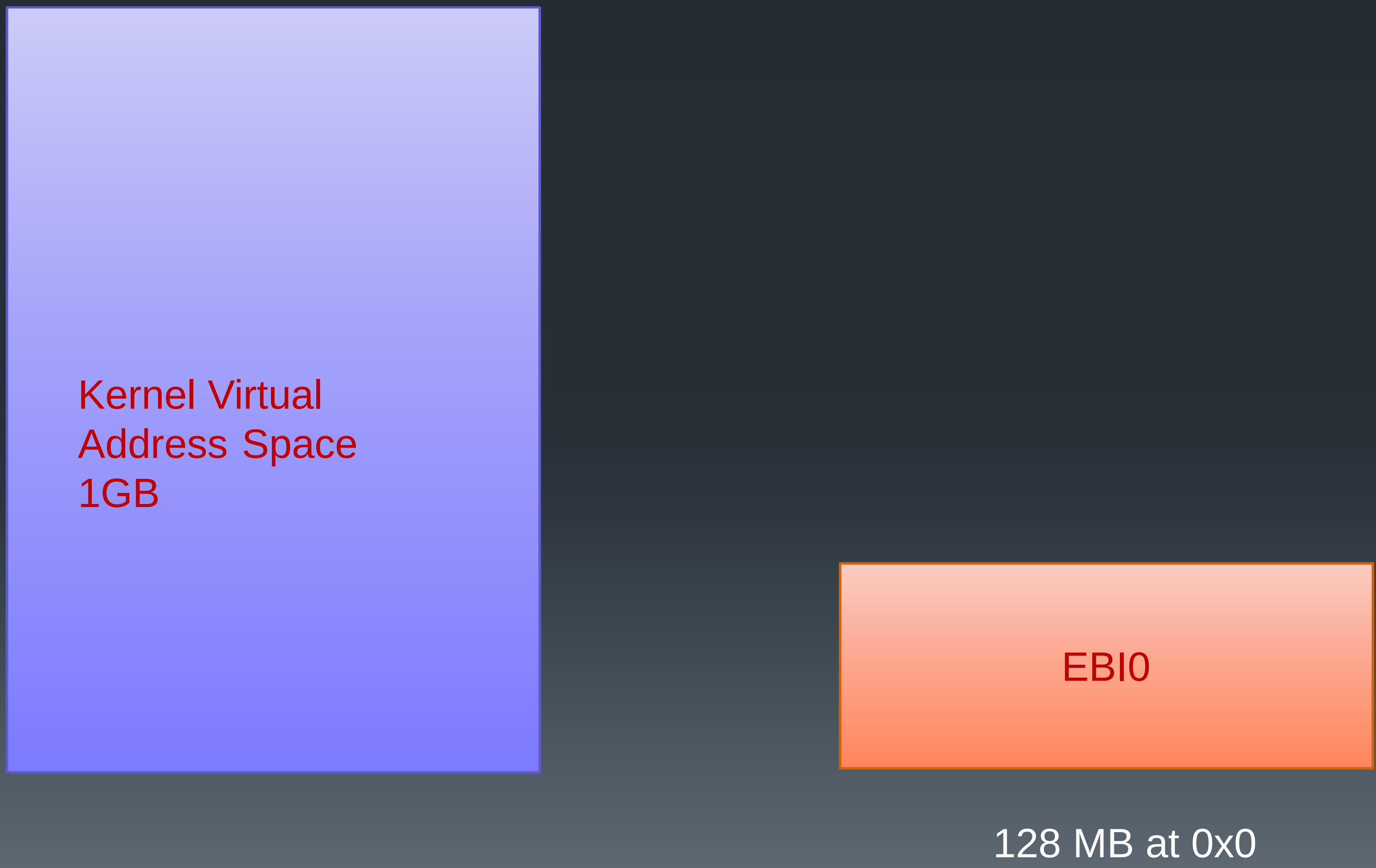
```
<5>[    0.000000] Virtual kernel memory layout:
<5>[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
<5>[    0.000000]     fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
<5>[    0.000000]     DMA     : 0xff000000 - 0xffe00000   (  14 MB)
<5>[    0.000000]     vmalloc : 0xda000000 - 0xfa000000   ( 512 MB)
<5>[    0.000000]     lowmem  : 0xc0000000 - 0xd9f00000   ( 415 MB)
<5>[    0.000000]     pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
<5>[    0.000000]     modules : 0xbf000000 - 0xbfe00000   (  14 MB)
<5>[    0.000000]       .init : 0xc0008000 - 0xc0045000   ( 244 kB)
<5>[    0.000000]       .text : 0xc0100000 - 0xc0a5a1f0   (9577 kB)
<5>[    0.000000]       .data : 0xc0b00000 - 0xc0bc3230   ( 781 kB)
<5>[    0.000000]        .bss : 0xc0bc326c - 0xc0d85a10   (1802 kB)
```

Memory Model vs Memory Zones

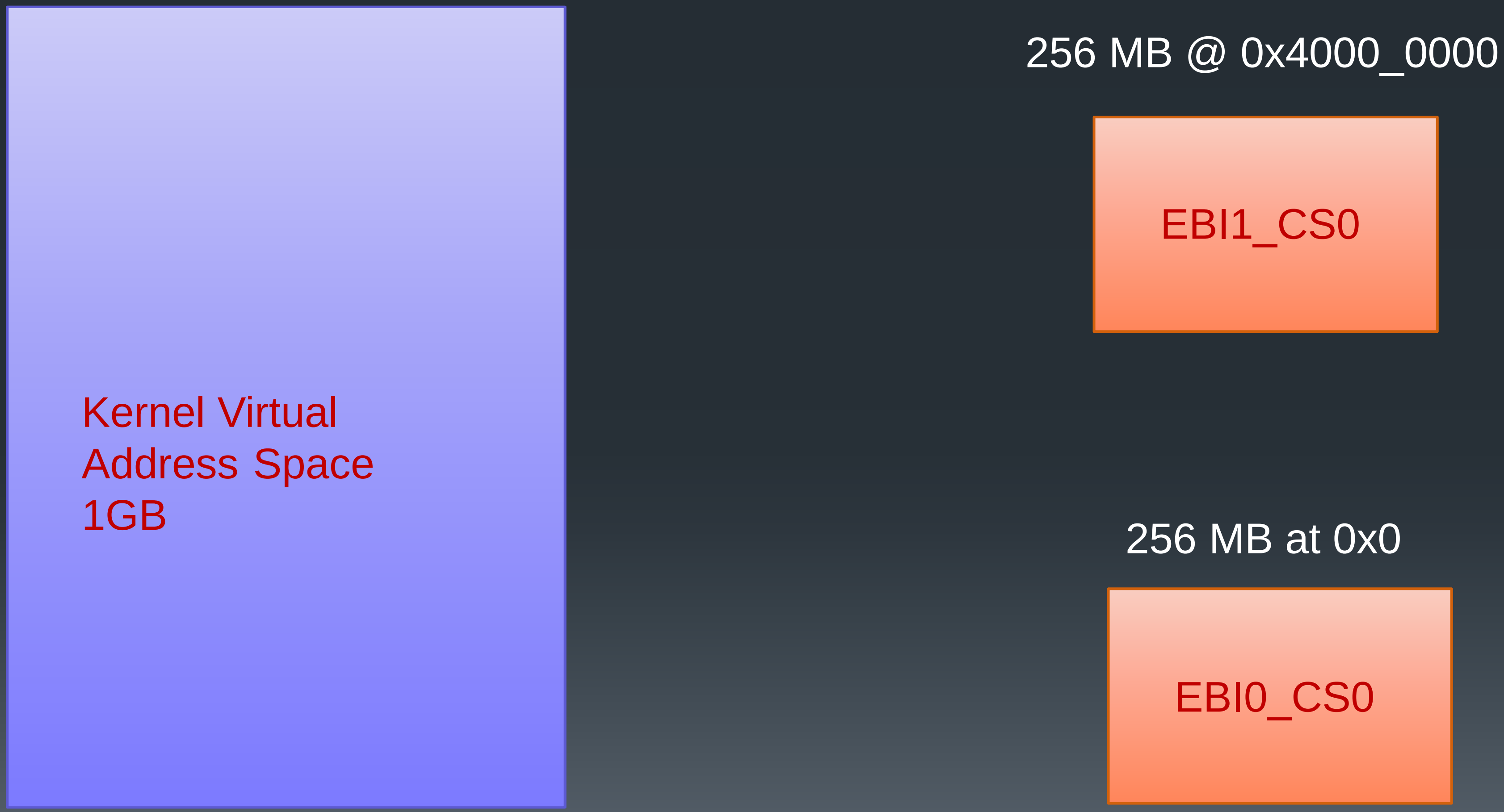
Memory Models



Simplified Memory Map – 7x27



Simplified Memory Map – 7x30



SPARSEMEM

- Divides the physical address space into SPARSEMEM sections
- SECTION_SIZE can be changed and is compile time
- Some of the Sections contain “real memory” some don’t
  - memory_present
- Provides different pfn_to_page() translations for each section
  - Variations exist on how this is done
- Simplified view: Helps to stitch the holes together

MEMORY ZONES

- Divide a NUMA node into multiple zones
  - On UMA, there is just one node

- Division helps kernel to distinguish between “cheap” vs “critical” memory

- ZONE_DMA
- ZONE_NORMAL => lowmem
- ZONE_HIGHMEM => highmem
- ZONE_MOVABLE => Movable pages

- ZONE_MOVABLE is logical

LOWMEM

- Mapped Directly in Kernel Address Space
  - Always Reachable by a Kernel Space Pointer
  - Often termed as a “Kernel Logical Address”

- V->P conversion is simple offset arithmetic

- Why is this needed ?

  - Certain data structures must live in LOWMEM
    - E.g. Linked List pointers cannot be transient
    - E.g. Page tables must reside in lowmem
  - Performance

HIGHMEM

- Increases the usable address space
- Needs to access through temporary mappings, incurs a overhead
- Fairly new to ARM, from 2.6.30 onwards
- Cannot be used all purposes e.g. for DMA
- Typically user space pages
- Exposes bugs in drivers that make assumptions about Virtual addresses

VMALLOC

- Why is it needed ?
  - Require large physical regions which cannot be allocated in a contiguous fashion in a running system because of fragmentation
  - Can use bootmem as a workaround, but never released and cannot be used by modules

- VMALLOC_RESERVE determines the size of the vmalloc region

- 3 stages
  - Obtain a vm_area in the vmalloc range
  - Obtain one page at a time to back the area
  - Set up V->P mappings so that this can be used

Virtual Memory Trade Offs

- Remember, everything has to fit in 1G
- 1G - Device Mappings - Kernel Code and Data = ~ 750 MB
- Kernel will then calculate vmalloc size as
    - VMALLOC_END – VMALLOC_RESERVE
- Then kernel examines to see how much can be directly mapped by lowmem
- Remaining becomes highmem
- lowmem will be reduced if VMALLOC increases and viz.
- More addressable memory means more page tables and page tables live in lowmem, so actual lowmem available is reduced

Memory Management for Special Purposes

Goals

- Multimedia Subsystems need buffers to talk to their cores
- MM Cores either have a SMMU or don't
- MM Drivers need to talk to each other in the pipeline of a use case

Roles

Allocator: Allocate/Free buffers for Clients

- Buffers can have attributes
  - Caching Policy
  - Alignment

Exporter: Share buffers so that consumers can consume it

- Apply policies that govern sharing

Mapper: Map the buffers into appropriate xMMUs

Handle: Need a unique representation to talk across subsystems

Comparison of Internal Solutions

Role	PMEM World	VCM World (Deprecated)	ION World
Allocator	Originally contiguous carve-out Now: mempool	libgenalloc	mempool vmalloc
Exporter	PMEM via mmap	VCM via IOCTL	ION via IOCTL
CPU Mapper	ioremap	ioremap	ioremap
IOMMU Mapper	None	IOMMU Driver	IOMMU Driver
Handle	file pointer	vcm_context	ion_handle
Ref Count	file pointer	vcm_buff	ion_handle and ion_buffer

Comparison with Upstream

Allocator: CMA

Exporter: ION (for Android)

Mapper: SMMU Driver

Handle: dma_buf

Ref Count: file pointer (used by dma_buf)

Thank You