БУДУЩЕЕ
В НАШИХ
РУКАХ

Как мы AMD GPU на ПЛИС с RISC-V Linux запускали

Мирошниченко Сергей,
2024.11.30

Затравка

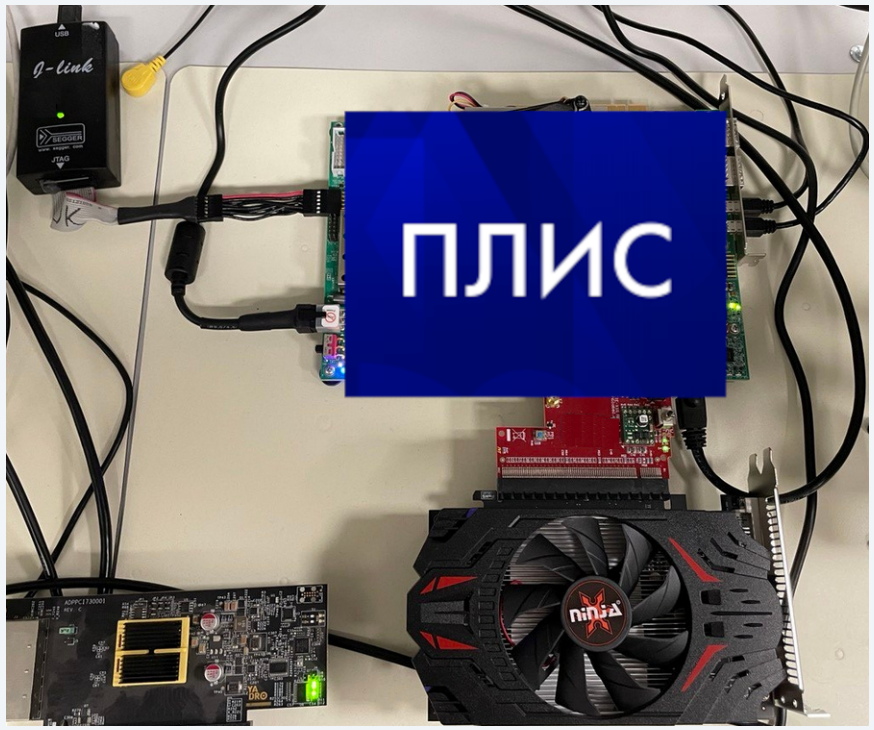
Хотелось всего и сразу:

- Проверить наши PCIe на ПЛИС
- Radeon вообще сможет в OpenGL под RISC-V Linux?
- А другие устройства?
- Стандарты открыты, драйверы тоже

Минуточку!

- BAR'ы влезли с третьего раза
- IP-блок сломался о прерывания
- Все указатели испорчены
- Код нужно отучить от 32-битности
- Мимоходом обнаружился баг в MMU

PCIe в жизни человека



Сетевые карты
Требования Enterprise: RDMA, Infiniband

Накопители
Корзины скоростных NVMe

Прочие адаптеры
SAS, iSCSI и пр.

Общие регистры

AMD GPU

drivers/gpu/drm/amd/amdgpu/amdgpu_device.c

```
int amdgpu_device_health_check()
{
  pci_read_config_dword(pdev, PCI_COMMAND, &status);
  if (PCI_POSSIBLE_ERROR(status)) {
    dev_err(dev, "device lost from bus!");
    ret = -ENODEV;
  }
```

NVMe

drivers/nvme/host/pci.c

```
  pci_read_config_word(dev,
                       PCI_STATUS, &pci_status);
```

Краткая справка

Регистры MMIO

AMD GPU

drivers/gpu/drm/amd/amdgpu/amdgpu_device.c

```
int amdgpu_device_health_check()
{
  pci_read_config_dword(pdev, PCI_COMMAND, &status);
  if (PCI_POSSIBLE_ERROR(status)) {
    dev_err(dev, "device lost from bus!");
    ret = -ENODEV;
  }
```

```
int gfx_v8_0_ring_test_ring(adev)
{
  WREG32(mmSCRATCH_REG0, 0xCAFEDEAD);
```

NVMe

drivers/nvme/host/pci.c

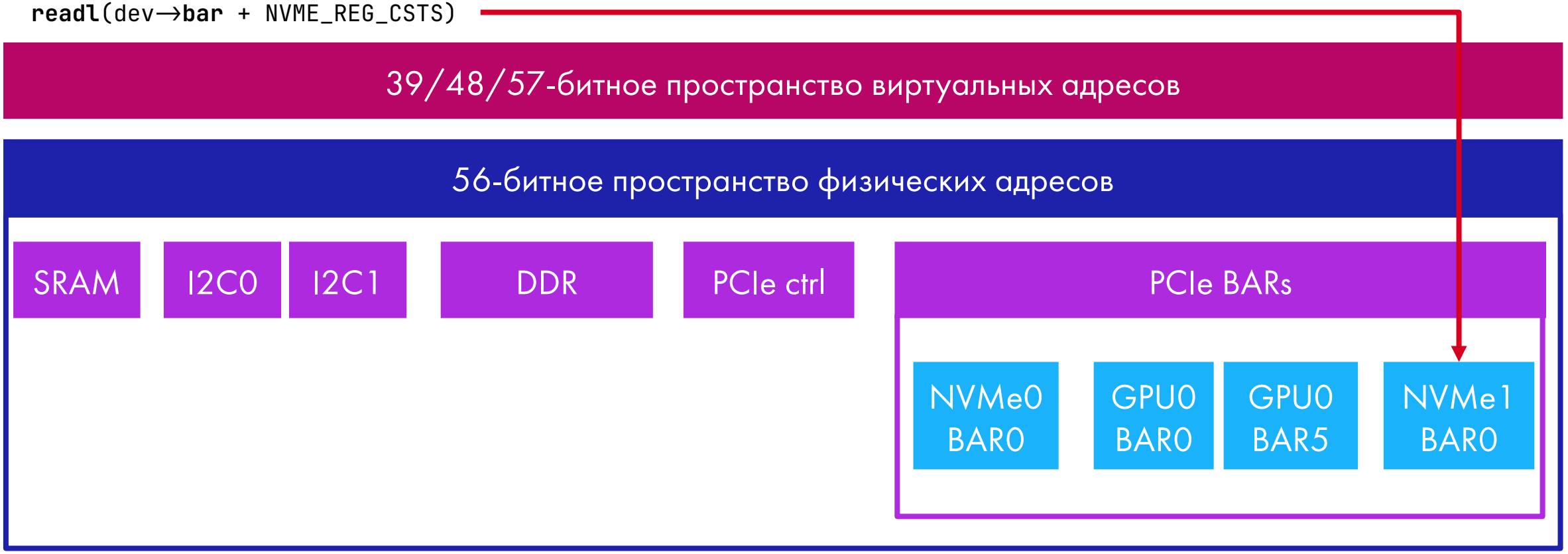
```
  pci_read_config_word(dev,
                       PCI_STATUS, &pci_status);
```

```
nvme_pci_enable(dev)
{
  if (readl(dev->bar + NVME_REG_CSTS) == -1) {
    result = -ENODEV;
    goto disable;
  }
```

Краткая справка

BAR

```
readl(dev→bar + NVME_REG_CSTS)
```




```
dev→bar = ioremap(pci_resource_start(pdev, 0), size);
dev→bar_mapped_size = pci_resource_len(pdev, 0)
```

Типы BAR'ов и карта памяти

Три BAR'а для AMD GPU:

- VRAM: 256МиБ+
  - Prefetchable, 64-bit
- MMIO (регистры): 256КиБ
  - Non-prefetchable
  - 32-bit only
- Doorbells: 2МиБ
  - Prefetchable, 64-bit

Но:

```
# cat /proc/iomem
...

 40000000-40ffffff : 4000000.pcie config
 41000000-4fffffff : pcie@4000000 # 240MiB
   41000000-410fffff : PCI Bus 0000:01

...

400080000-47efffffff : System RAM  # DDR > 4GiB
  400202000-4041b3927 : Kernel image
    400202000-40095aa73 : Kernel code
    403a00000-403dfffff : Kernel rodata
    404000000-4041487ff : Kernel data
    404149000-4041b3927 : Kernel bss

...
```

Типы BAR'ов и карта памяти (вторая попытка)

Три BAR'а для AMD GPU:

- VRAM: 256МиБ+
  - Prefetchable, 64-bit
- MMIO (регистры): 256КиБ
  - Non-prefetchable
  - 32-bit only
- Doorbells: 2МиБ
  - Prefetchable, 64-bit

Но:

```
# cat /proc/iomem
...
```

```
400080000-47effffff : System RAM  # DDR > 4GiB
  400202000-4041b3927 : Kernel image
    400202000-40095aa73 : Kernel code
    403a00000-403dfffff : Kernel rodata
    404000000-4041487ff : Kernel data
    404149000-4041b3927 : Kernel bss

...

2001000000-2fffffffff : pcie@4000000  # > 4GiB
  2080000000-21fffffffff : PCI Bus 0000:01
```

Типы BAR'ов и карта памяти (третья попытка)

Три BAR'а для AMD GPU:

- VRAM: 256МиБ+
  - Prefetchable, 64-bit
- MMIO (регистры): 256КиБ
  - Non-prefetchable
  - 32-bit only
- Doorbells: 2МиБ
  - Prefetchable, 64-bit

Добавили второй AXI Slave:

```
# cat /proc/iomem
...

41000000-4fffffff : pcie@4000000
  41000000-410fffff : PCI Bus 0000:01
    41000000-4103ffff : 0000:01:00.0
    41040000-4105ffff : 0000:01:00.0
    41060000-41063fff : 0000:01:00.1
  41100000-4110ffff : 0000:00:00.0

2001000000-2fffffffff : pcie@4000000
  2080000000-21ffffffff : PCI Bus 0000:01
    2080000000-20801fffff : 0000:01:00.0
    2100000000-21ffffffff : 0000:01:00.0


# lspci -vvvs 01:00.0
...
 Region 0: Memory at 2100000000 (64-bit, prefetchable) [size=4G]
 Region 2: Memory at 2080000000 (64-bit, prefetchable) [size=2M]
 Region 5: Memory at 41000000 (32-bit, non-prefetchable) [size=256K]
 Expansion ROM at 41040000 [disabled] [size=128K]
```

Краткая справка

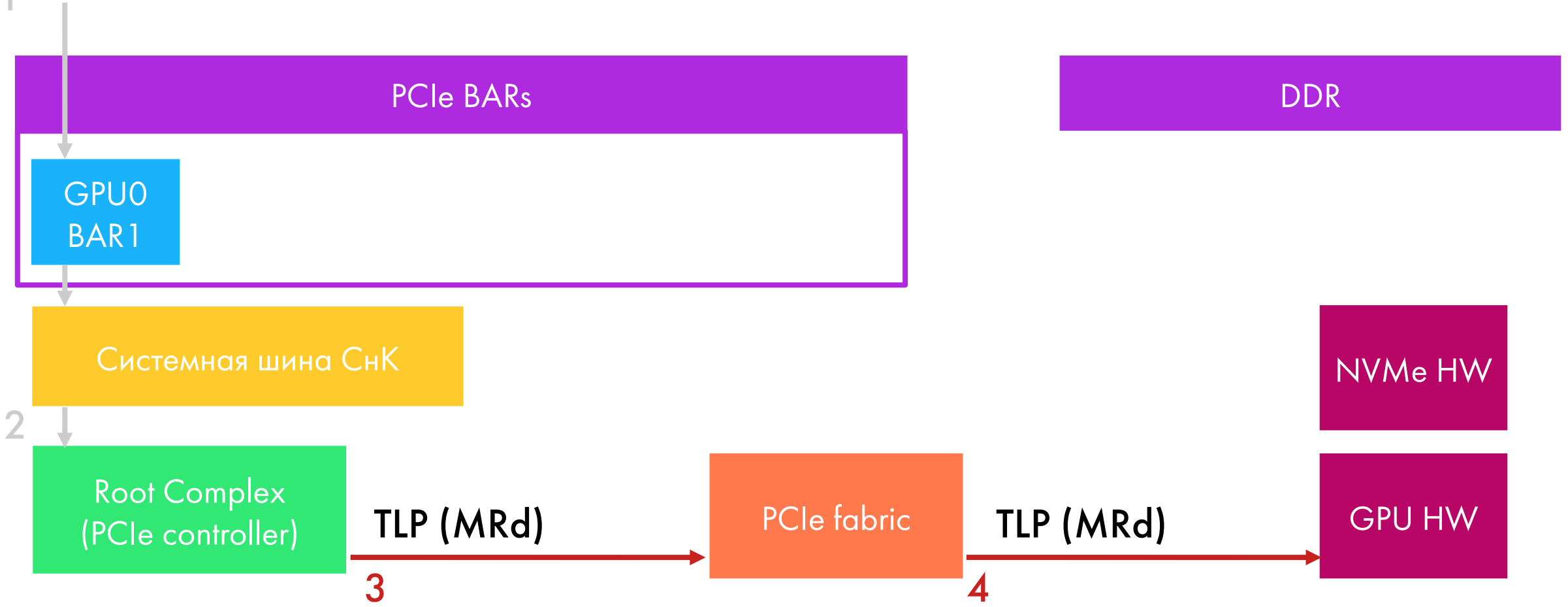
TLP

```
tmp = RREG32_SOC15(GC, 0, mmCP_RB_DOORBELL_CONTROL);
```

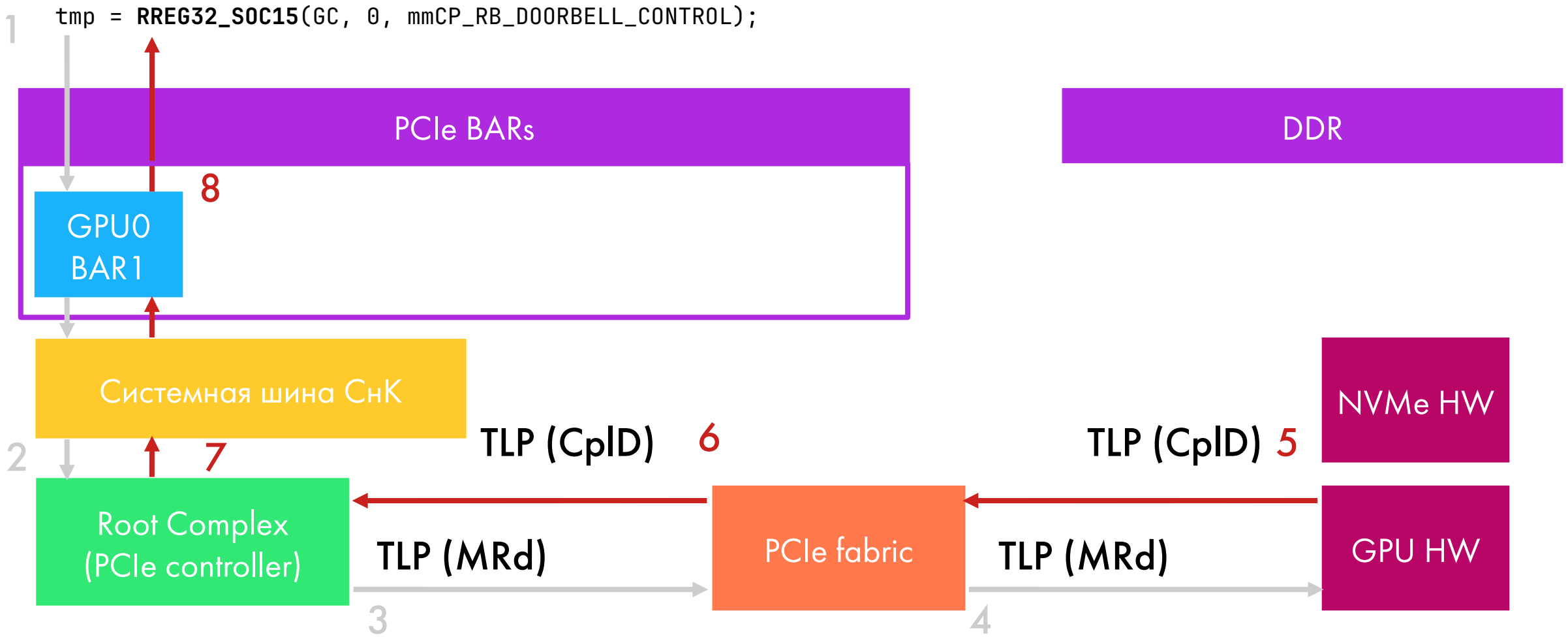
Краткая справка

TLP

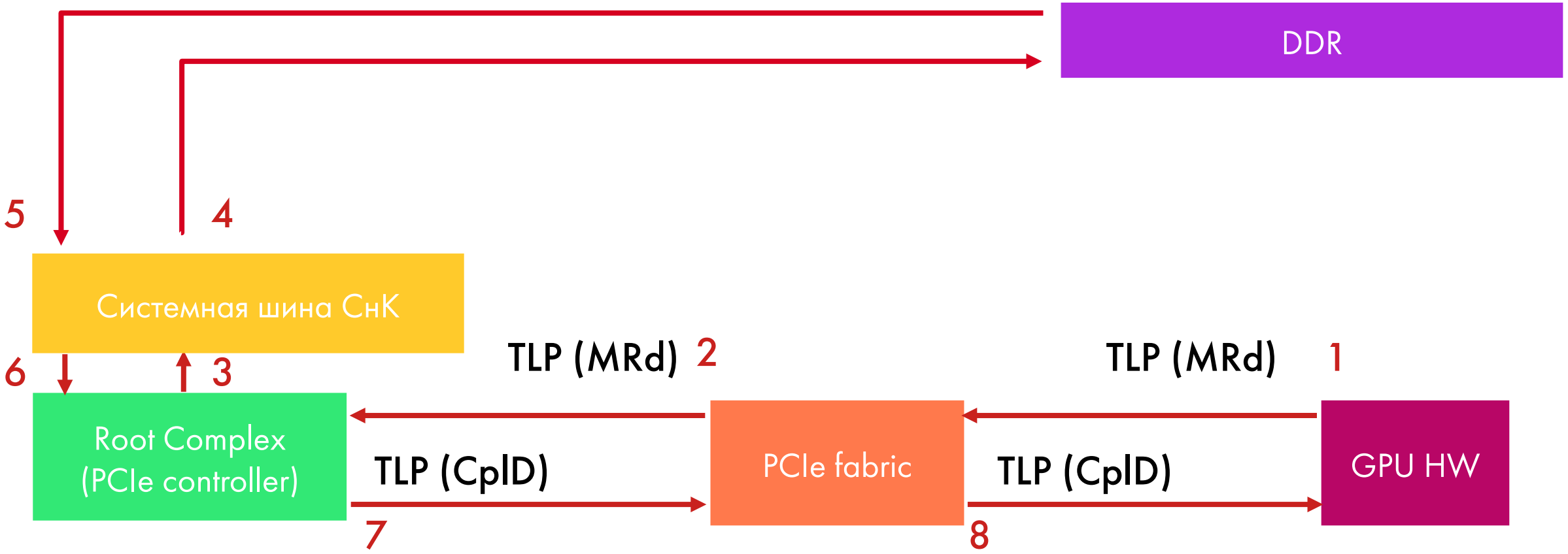
```
tmp = RREG32_SOC15(GC, 0, mmCP_RB_DOORBELL_CONTROL);
```



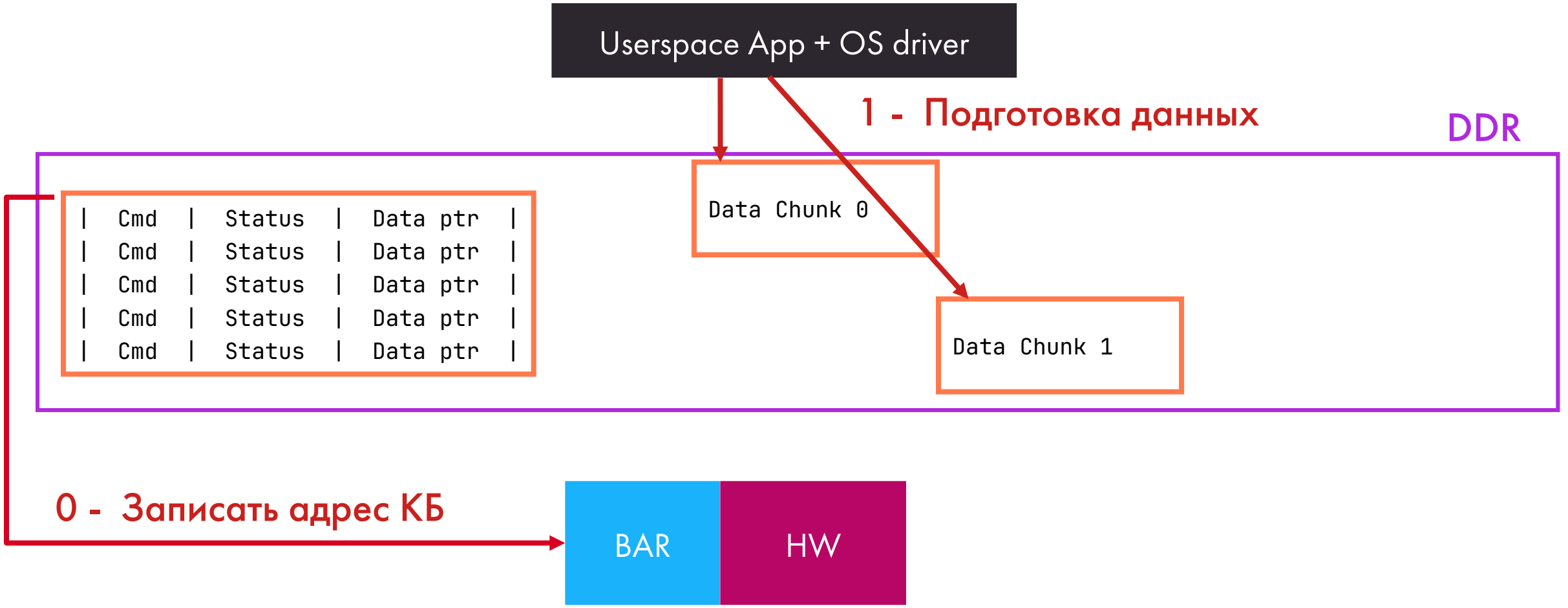
TLP


1
tmp = RREG32_SOC15(GC, 0, mmCP_RB_DOORBELL_CONTROL);
PCIe BARs
DDR
8
GPU0
BAR1
Системная шина СнК
NVMe HW
2
7
TLP (CplD)
6
TLP (CplD)
5
Root Complex
(PCIe controller)
PCIe fabric
GPU HW
TLP (MRd)
TLP (MRd)
3
4

DMA

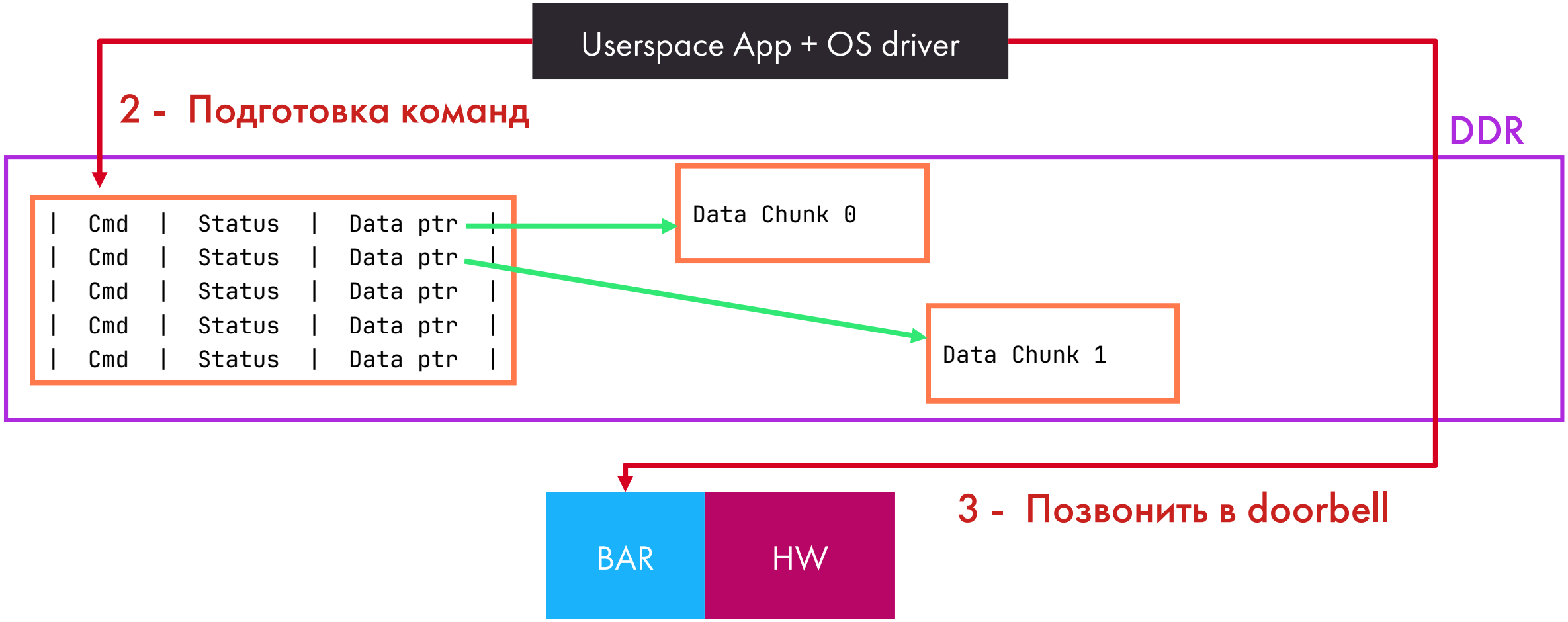

DDR
5
4
Системная шина CнK
6
3
TLP (MRd) 2
TLP (MRd) 1
Root Complex
(PCIe controller)
TLP (CplD)
PCIe fabric
TLP (CplD)
GPU HW
7
8

Circular buffers 0/2

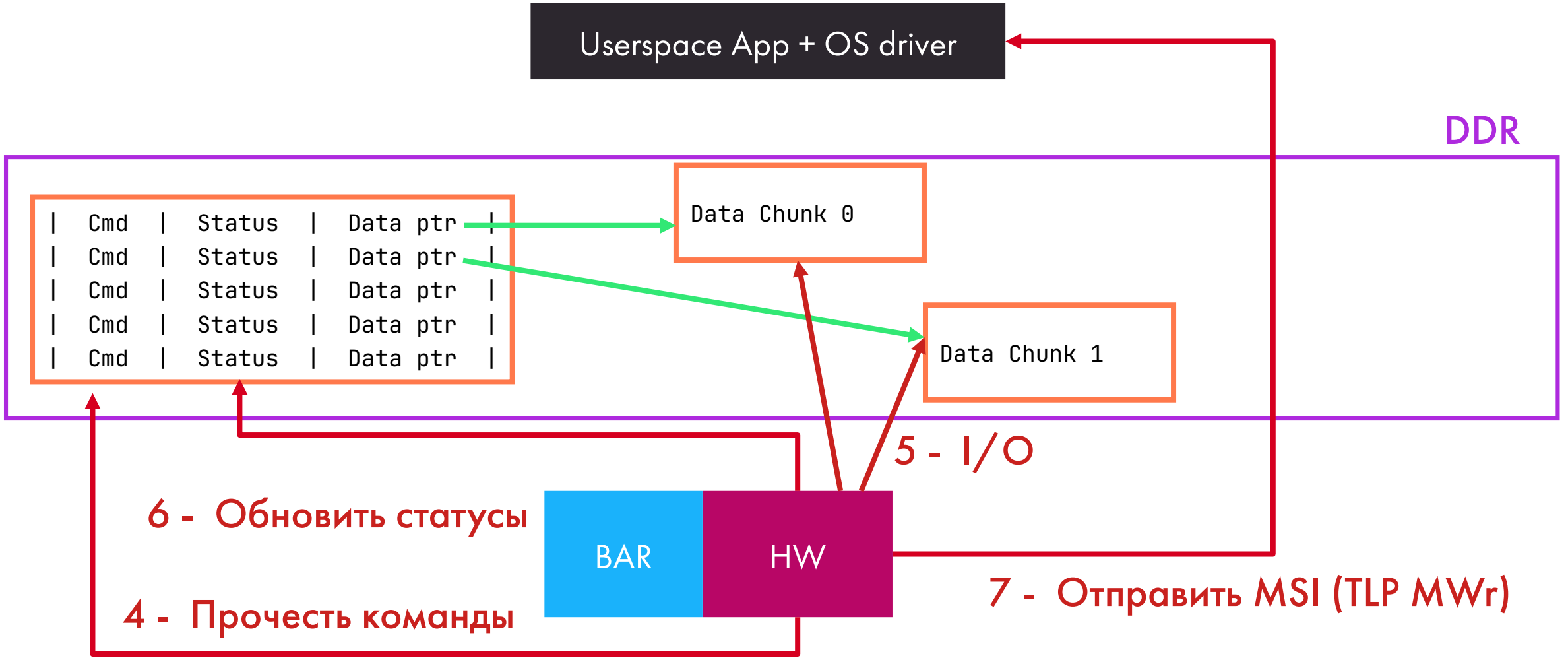


Краткая справка

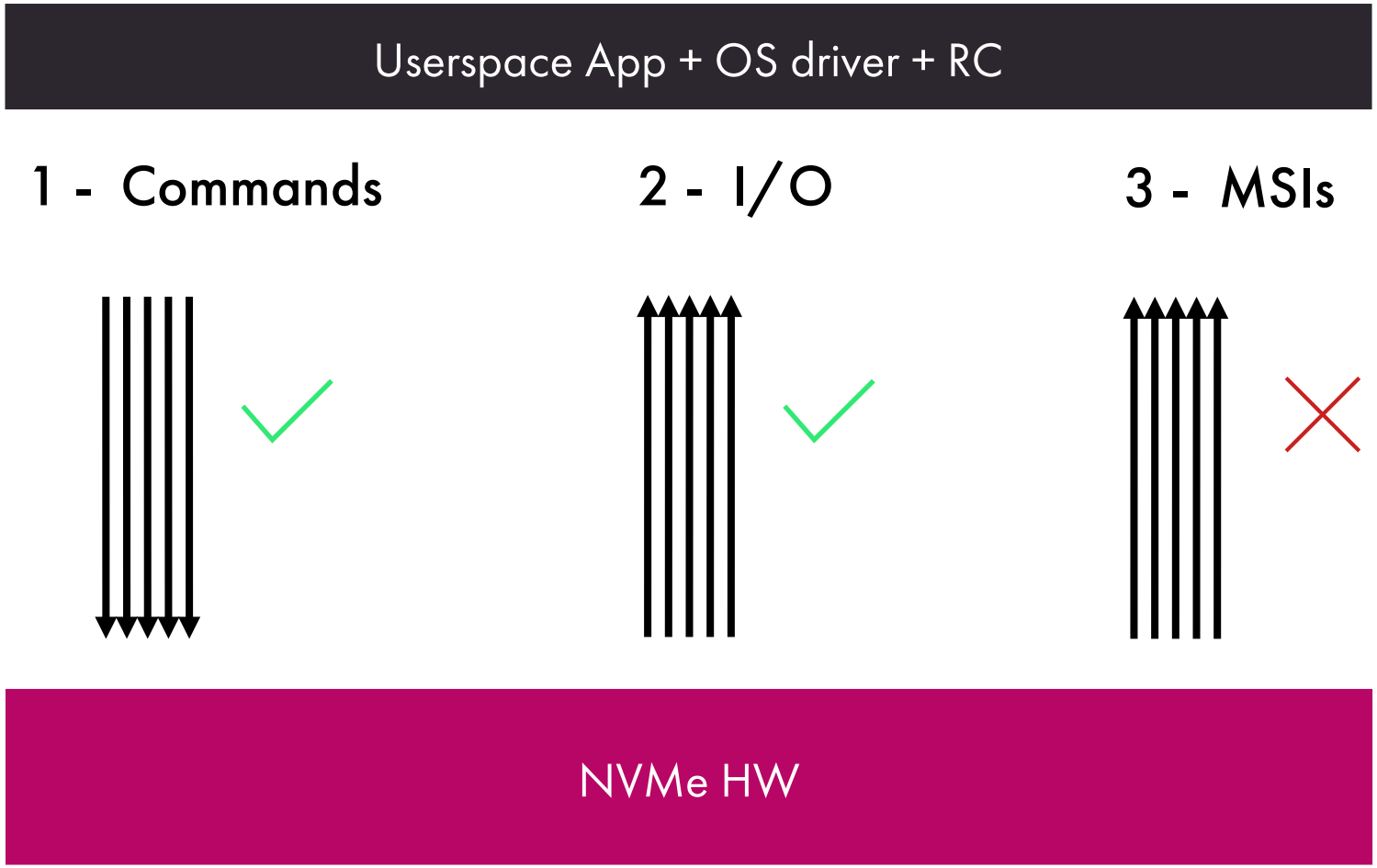
Circular buffers 1/2



Circular buffers 2/2



MSI overflow

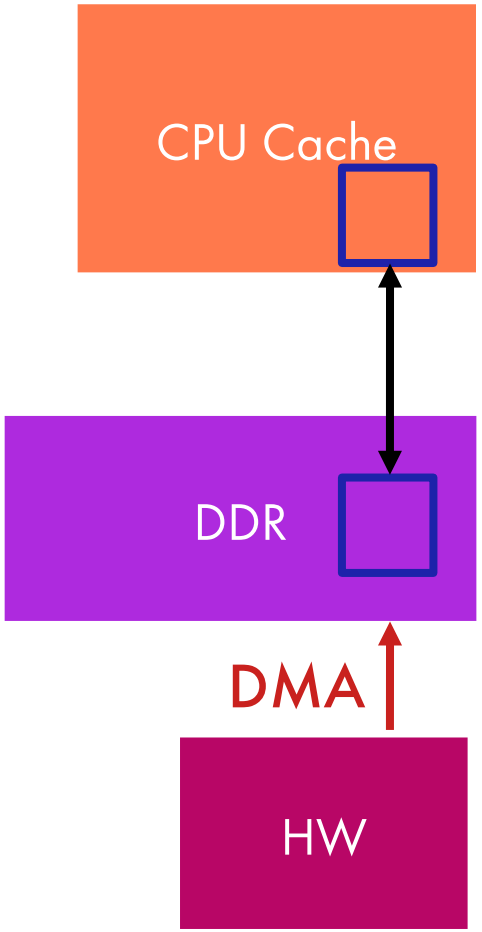


У RC переполнился FIFO прерываний!

Решения:

- Сократить глубину очереди (`nvme.io_queue_depth: 1024 → 8`)
- Оптимизировать обработку прерываний
- Добавить обработку ошибок

Память для rings и когерентность



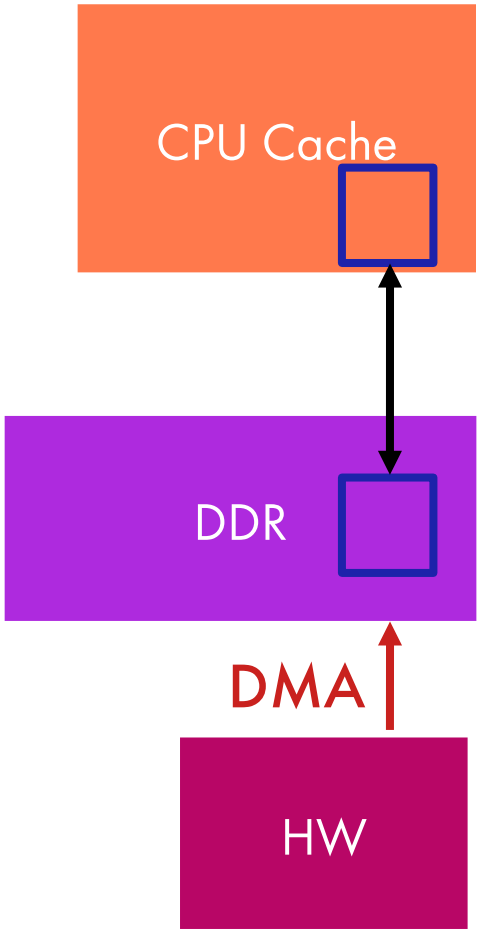
https://lkml.org/lkml/2018/5/18/979

```
dma_map_single(, dir)
dma_sync_single_for_cpu(, dir)
dma_sync_single_for_device(, dir)
dma_unmap_single(, dir)
```

	map	**for_cpu**	**for_device**	**unmap**
TO_DEV	writeback	none	writeback	none
TO_CPU	invalidate	invalidate*	invalidate	invalidate*
BIDIR	writeback	invalidate	writeback	invalidate

При наличии когерентного порта эти функции вырождаются в no-op

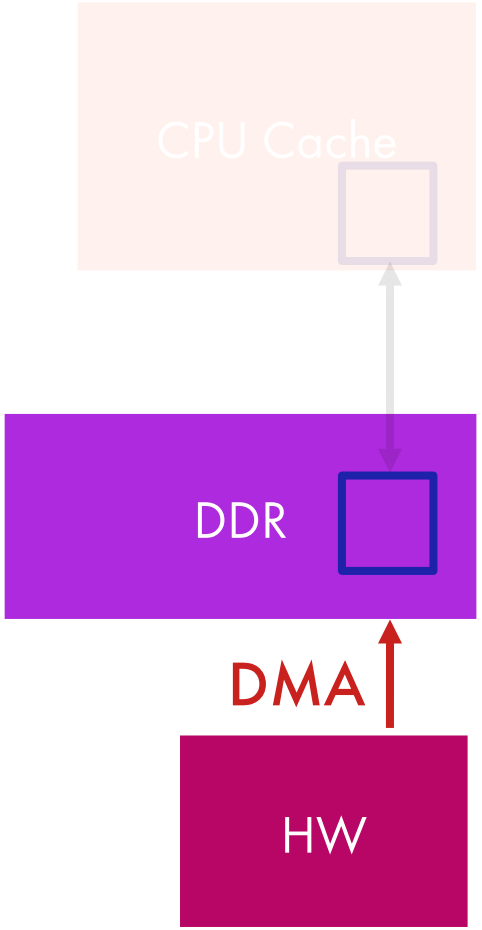
Память для rings и DMA API



linux/drivers/gpu/drm/ttm/ttm_pool.c

```
static struct page *ttm_pool_alloc_page(..., unsigned int order)
{
  vaddr = dma_alloc_attrs((1ULL << order) * PAGE_SIZE,
                          &dma->addr, gfp_flags, attr);
```

Нет кеша, нет проблем



linux/drivers/gpu/drm/ttm/ttm_pool.c

```
static struct page *ttm_pool_alloc_page(..., unsigned int order)
{
  vaddr = dma_alloc_attrs((1ULL << order) * PAGE_SIZE,
                          &dma->addr, gfp_flags, attr);
```

Device Tree

```
reserved-memory {
  linux,dma@0 {
    compatible = "shared-dma-pool";
    reg = <UPPER32(BASE) LOWER32(BASE)
           UPPER32(SIZE) LOWER32(SIZE)>;
    linux,dma-default;
    no-map;
  }
}
```

```
opensbi-domains {
  compatible = "opensbi,domain,config";

  dma_mem: dma_mem {
    compatible = "opensbi,domain,memregion";
    base = <UPPER32(BASE) LOWER32(BASE)>;
    order = <LOG2_DMA_SIZE>;
    mmio;
  };
```

Rings и негодные адреса

linux/drivers/gpu/drm/ttm/ttm_pool.c

```
static struct page *ttm_pool_alloc_page(..., unsigned int order)
{
  vaddr = dma_alloc_attrs((1ULL << order) * PAGE_SIZE,
                          &dma->addr, gfp_flags, attr);

  /* TODO: This is an illegal abuse of the DMA API, but we need to rework
   * TTM page fault handling and extend the DMA API to clean this up.
   */
   struct page *p;

   if (is_vmalloc_addr(vaddr))
     p = vmalloc_to_page(vaddr);
   else
     p = virt_to_page(vaddr);
```

https://lore.kernel.org/linux-mm/20190614134726.3827-13-hch@lst.de/

Subject: [PATCH 12/16] staging/comedi: mark as broken

comedi_buf.c abuse the DMA API in gravely broken ways, as it assumes it can call virt_to_page on the result, and the just remap it as uncached using vmap. Disable the driver until this API abuse has been fixed.

Signed-off-by: Christoph Hellwig <hch@lst.de>

CMA вместо DMA

```
diff --git a/kernel/dma/contiguous.c b/kernel/dma/contiguous.c
index 055da410ac71..4d1f636d165a 100644
--- a/kernel/dma/contiguous.c
+++ b/kernel/dma/contiguous.c
@@ -360,7 +360,7 @@ struct page *dma_alloc_contiguous(dev, size, gfp)
  if (dev->cma_area)
      return cma_alloc_aligned(dev->cma_area, size, gfp);
  if (size <= PAGE_SIZE)
-     return NULL;
+     size = PAGE_SIZE;
```

If it's stupid but it works, it isn't stupid.

Device Tree

```
reserved-memory {
  linux,dma@0 {
    compatible = "shared-dma-pool";
    reg = <UPPER32(BASE) LOWER32(BASE)
           UPPER32(SIZE) LOWER32(SIZE)>;
    linux,cma-default;
    reusable;
  };
}
```

DEBUG_PAGEALLOC

Пока разбирал CMA, включил отладку, перестало загружаться!

Нужно всегда держать наготове свежее ядро

```
commit 311cd2f6e25380cff0abc2884dc6a3d33bc9b5c3
Author: Alexandre Ghiti <alexghiti@rivosinc.com>
Date:   Wed Nov 8 08:59:30 2023 +0100

    riscv: Fix set_memory_XX() and set_direct_map_XX() by splitting huge linear mappings

    When STRICT_KERNEL_RWX is set, any change of permissions on any kernel
    mapping (vmalloc/modules/kernel text...etc) should be applied on its
    linear mapping alias. The problem is that the riscv kernel uses huge
    mappings for the linear mapping and walk_page_range_novma() does not
    split those huge mappings.

    So this patchset implements such split in order to apply fine-grained
    permissions on the linear mapping.
```

GFP_DMA32

Да:

```
--- a/drivers/gpu/drm/ttm/ttm_device.c
+++ b/drivers/gpu/drm/ttm/ttm_device.c
@@ -96,7 +96,7 @@ static int ttm_global_init(void)
        ttm_pool_mgr_init(num_pages);
        ttm_tt_mgr_init(num_pages, num_dma32);

-       glob->dummy_read_page = alloc_page(__GFP_ZERO | GFP_DMA32 |
+       glob->dummy_read_page = alloc_page(__GFP_ZERO |
                                           __GFP_NOWARN);
```

Но:

```
# cat /proc/iomem
...
400080000-47effffff : System RAM  # DDR > 4GiB
  400202000-4041b3927 : Kernel image
    400202000-40095aa73 : Kernel code
    403a00000-403dfffff : Kernel rodata
    404000000-4041487ff : Kernel data
    404149000-4041b3927 : Kernel bss
...
```

Другие последствия не-32-битной памяти

linux/drivers/gpu/drm/ttm/ttm_device.c

```
 * @use_dma_alloc: If coherent DMA allocation API should be used.
 * @use_dma32: If we should use GFP_DMA32 for device memory allocations.

int ttm_device_init(..., bool use_dma_alloc, bool use_dma32)
```

drivers/gpu/drm/amd/amdgpu/amdgpu_ttm.c

```
@@ -1829,8 +1829,8 @@ int amdgpu_ttm_init(struct amdgpu_device *adev)
        r = ttm_device_init(
-                           adev->need_swiotlb,
-                           dma_addressing_limited(adev->dev));
+                           true,
+                           false);
```

А как это обошли в x86?

AMD EPYC 7313

```
% sudo cat /proc/iomem
...                 # 32-bit addresses
000a0000-000fffff : Reserved
  000a0000-000bffff : PCI Bus 0000:40
  000c0000-000dffff : PCI Bus 0000:00
  000f0000-000fffff : System ROM
00100000-2fffffff : System RAM     # 768MiB
...
b0000000-b3cfffff : PCI Bus 0000:40
  b0000000-b01fffff : PCI Bus 0000:46
  b2000000-b30fffff : PCI Bus 0000:44
    b2000000-b30fffff : PCI Bus 0000:45
      b2000000-b2ffffff : 0000:45:00.0
      b3000000-b301ffff : 0000:45:00.0
...
```

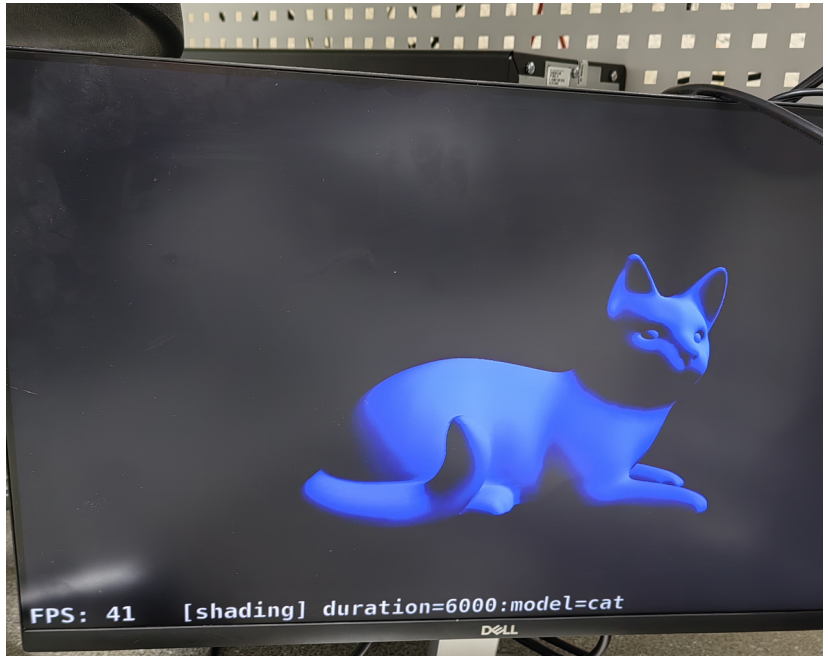
```
...         # 64-bit addresses
100000000-404fdfffff : System RAM       # 253GiB
404fe00000-404fffffff : Reserved
4050000000-804f2fffff : System RAM      # 255GiB
  49f3000000-49f43fffff : Kernel code
  49f4400000-49f52f0fff : Kernel rodata
  49f5400000-49f5836cbf : Kernel data
  49f62d9000-49f67fffff : Kernel bss
...
20080200000-280801fffff : PCI Bus 0000:40
  20080200000-200803fffff : PCI Bus 0000:41
...
```

Баг в TLB

Вот и всё!

Успешный запуск

```
# glmark2-es2-drm -b "shading:model=cat:duration=6000" --annotate
=======================================================
    glmark2 2023.01
=======================================================
    OpenGL Information
    GL_VENDOR:      AMD
    GL_RENDERER:    AMD Radeon RX 560 Series (radeonsi, polaris11, LLVM 15.0.3, DRM 3.57, 6.9.1)
    GL_VERSION:     OpenGL ES 3.2 Mesa 24.0.9
    Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
    Surface Size:   1920x1080 fullscreen
=======================================================
```



Выводы

Итого:

- Заведенный в первый 4ГиБ участок RAM избавит от ряда проблем
- Для PCIe BARs требуется два региона:
  - Один в первых 4ГиБ (MMIO)
  - Второй огромный (память)
- RC должен справляться с потоком MSI
- Часть драйверов проверены только на x86, обеспечить кросс-платформенность предстоит самим
- Удобно держать под рукой несколько релизов ядра Linux
- Любая экзотическая конфигурация может оказаться полезной