```
struct dma_map_ops arm_dma_ops = {
    .alloc          = arm_dma_alloc,
    .free           = arm_dma_free,
    .mmap           = arm_dma_mmap,
    .get_sgtable        = arm_dma_get_sgtable,
    .map_page       = arm_dma_map_page,
    .unmap_page     = arm_dma_unmap_page,
    .map_sg         = arm_dma_map_sg,
    .unmap_sg       = arm_dma_unmap_sg,
    .sync_single_for_cpu    = arm_dma_sync_single_for_cpu,
    .sync_single_for_device = arm_dma_sync_single_for_device,
    .sync_sg_for_cpu    = arm_dma_sync_sg_for_cpu,
    .sync_sg_for_device = arm_dma_sync_sg_for_device,
    .set_dma_mask       = arm_dma_set_mask,
};
```

```
struct dma_map_ops arm_coherent_dma_ops = {
    .alloc          = arm_coherent_dma_alloc,
    .free           = arm_coherent_dma_free,
    .mmap           = arm_dma_mmap,
    .get_sgtable        = arm_dma_get_sgtable,
    .map_page       = arm_coherent_dma_map_page,
    .map_sg         = arm_dma_map_sg,
    .set_dma_mask       = arm_dma_set_mask,
};
```

差异在.alloc, .free 和.map_page callback上！

```
/*
 * Allocate DMA-coherent memory space and return both the kernel remapped
 * virtual and bus address for that space.
 */
void *arm_dma_alloc(struct device *dev, size_t size, dma_addr_t *handle,
            gfp_t gfp, struct dma_attrs *attrs)
{
    pgprot_t prot = __get_dma_pgprot(attrs, PAGE_KERNEL);
    void *memory;

    if (dma_alloc_from_coherent(dev, size, handle, &memory))
        return memory;

    return __dma_alloc(dev, size, handle, gfp, prot, false,
               __builtin_return_address(0));
}

static void *arm_coherent_dma_alloc(struct device *dev, size_t size,
    dma_addr_t *handle, gfp_t gfp, struct dma_attrs *attrs)
{
    pgprot_t prot = __get_dma_pgprot(attrs, PAGE_KERNEL);
    void *memory;

    if (dma_alloc_from_coherent(dev, size, handle, &memory))    ①
        return memory;

    return __dma_alloc(dev, size, handle, gfp, prot, true,
               __builtin_return_address(0));
}
```

这两个functions唯一的区别就是在调用__dma_alloc()是传递的bool is_coherent参数不一样！

```c
static void *__dma_alloc(struct device *dev, size_t size, dma_addr_t *handle,
             gfp_t gfp, pgprot_t prot, bool is_coherent, const void *caller)
{
    u64 mask = get_coherent_dma_mask(dev);
    struct page *page = NULL;
    void *addr;

#ifdef CONFIG_DMA_API_DEBUG
    u64 limit = (mask + 1) & ~mask;
    if (limit && size >= limit) {
        dev_warn(dev, "coherent allocation too big (requested %#x mask %#llx)\n",
            size, mask);
        return NULL;
    }
#endif

    if (!mask)
        return NULL;

    if (mask < 0xffffffffULL)
        gfp |= GFP_DMA;

    /*
     * Following is a work-around (a.k.a. hack) to prevent pages
     * with __GFP_COMP being passed to split_page() which cannot
     * handle them.  The real problem is that this flag probably
     * should be 0 on ARM as it is not supported on this
     * platform; see CONFIG_HUGETLBFS.
     */
    gfp &= ~(__GFP_COMP);

    *handle = DMA_ERROR_CODE;
    size = PAGE_ALIGN(size);

    if (is_coherent || nommu())    ②
        addr = __alloc_simple_buffer(dev, size, gfp, &page);
    else if (!(gfp & __GFP_WAIT))    ③
        addr = __alloc_from_pool(size, &page);
    else if (!dev_get_cma_area(dev))    ④
        addr = __alloc_remap_buffer(dev, size, gfp, prot, &page, caller);
    else
        addr = __alloc_from_contiguous(dev, size, prot, &page, caller);    ⑤

    if (addr)
        *handle = pfn_to_dma(dev, page_to_pfn(page));

    return addr;
}
```

从上面及格函数可以看清Linux中dma memory allocation中的一些让人挠头的关系。

①

dma_alloc_from_coherent()实现在drivers/base/dma-coherent.c中，即如果enable了该文件，则其接管dma allocation的优先级最高。

②

如果没有enable dma-coherent.c implementation,那么如果要求是coherent方式分配dma memory，那么通过

__alloc_simple_buffer()实现。

③

如果申请dma memory时没有设置 `__GFP_WAIT`

```
#define __GFP_WAIT  ((__force gfp_t)___GFP_WAIT)    /* Can wait and reschedule? */
```

即申请dma memory动作时不能引起sleep，比如再interrupt context中必须这样，那么实际上通过dma memory pool完成。

④

dev_get_cma_area()

这里如果kernel没有enable drivers/base/dma-contiguous.c implementation，那么该function return NULL,通过

__alloc_remap_buffer() allocate.

⑤

drivers/base/dma-contiguous.c implementation被enable.

从这儿可看到，如果在

drivers/base/dma-contiguous.c

drivers/base/dma-coherent.c

都enable的情况下，前者的优先级也比后者高！