```
    reserved-memory {
        #address-cells = <0x2>;
        #size-cells = <0x2>;
        ranges;

        r4@6000000 {
            reg = <0x0 0x6000000 0x0 0x400000>;
            no-map;
        };

        linux,cma {
            compatible = "shared-dma-pool";
            reusable;
            size = <0x0 0x10000>;
            alignment = <0x0 0x2000>;
            linux,cma-default;
        };
    };
```

`reserved-memory` node分两部分

1. r4 memory(96M to 100M)
2. dma pool

---

r4 memory(96M to 100M)

arch/arm/kernel/setup.c/setup_arch()
=> arch/arm/mm/init.c/arm_memblock_init()
==> drivers/of/fdt.c/early_init_fdt_scan_reserved_mem()

```
/**
 * early_init_fdt_scan_reserved_mem() - create reserved memory regions
 *
 * This function grabs memory from early allocator for device exclusive use
 * defined in device tree structures. It should be called by arch specific c
ode
 * once the early allocator (i.e. memblock) has been fully activated.
 */
void __init early_init_fdt_scan_reserved_mem(void)
{
    int n;
    u64 base, size;

    if (!initial_boot_params)
        return;

    /* Reserve the dtb region */
    early_init_dt_reserve_memory_arch(__pa(initial_boot_params),
                      fdt_totalsize(initial_boot_params),
                      0);

    /* Process header /memreserve/ fields */
    for (n = 0; ; n++) {
        fdt_get_mem_rsv(initial_boot_params, n, &base, &size);
        if (!size)
            break;
        early_init_dt_reserve_memory_arch(base, size, 0);
    }

    of_scan_flat_dt(__fdt_scan_reserved_mem, NULL);
    fdt_init_reserved_mem();
}
```

对dtb中`reserved-memory` node的处理在__fdt_scan_reserved_mem()中。

```
/**
 * fdt_scan_reserved_mem() - scan a single FDT node for reserved memory
 */
static int __init __fdt_scan_reserved_mem(unsigned long node, const char *uname,
                                          int depth, void *data)
{
    static int found;
    const char *status;
    int err;

    if (!found && depth == 1 && strcmp(uname, "reserved-memory") == 0) {
        if (__reserved_mem_check_root(node) != 0) {                    ①
            pr_err("Reserved memory: unsupported node format, ignoring\n");
            /* break scan */
            return 1;
        }
        found = 1;
        /* scan next node */
        return 0;
    } else if (!found) {
        /* scan next node */
        return 0;
    } else if (found && depth < 2) {
        /* scanning of /reserved-memory has been finished */
        return 1;
    }

    status = of_get_flat_dt_prop(node, "status", NULL);            ②
    if (status && strcmp(status, "okay") != 0 && strcmp(status, "ok") != 0)
        return 0;

    err = __reserved_mem_reserve_reg(node, uname);                 ③
    if (err == -ENOENT && of_get_flat_dt_prop(node, "size", NULL))
        fdt_reserved_mem_save_node(node, uname, 0, 0);

    /* scan next node */
    return 0;
}
```

①

```
/**
 * __reserved_mem_check_root() - check if #size-cells, #address-cells provided
 * in /reserved-memory matches the values supported by the current implementation,
 * also check if ranges property has been provided
 */
static int __init __reserved_mem_check_root(unsigned long node)
{
    const __be32 *prop;

    prop = of_get_flat_dt_prop(node, "#size-cells", NULL);
    if (!prop || be32_to_cpup(prop) != dt_root_size_cells)
        return -EINVAL;

    prop = of_get_flat_dt_prop(node, "#address-cells", NULL);
    if (!prop || be32_to_cpup(prop) != dt_root_addr_cells)
        return -EINVAL;

    prop = of_get_flat_dt_prop(node, "ranges", NULL);
    if (!prop)
return -EINVAL;
    return 0;
}
```

```
 #address-cells = <0x2>;
 #size-cells = <0x2>;
 ranges;
```

`reserved-memory` node检查上面 3 个property

②

check status property，是否disable该node

③

```c
/**
 * res_mem_reserve_reg() - reserve all memory described in 'reg' property
 */
static int __init __reserved_mem_reserve_reg(unsigned long node,
                         const char *uname)
{
    int t_len = (dt_root_addr_cells + dt_root_size_cells) * sizeof(__be32);
    phys_addr_t base, size;
    int len;
    const __be32 *prop;
    int nomap, first = 1;

    prop = of_get_flat_dt_prop(node, "reg", &len);        (A)
    if (!prop)
        return -ENOENT;

    if (len && len % t_len != 0) {
        pr_err("Reserved memory: invalid reg property in '%s', skipping node
.\n",
               uname);
        return -EINVAL;
    }

    nomap = of_get_flat_dt_prop(node, "no-map", NULL) != NULL;    (B)


    while (len >= t_len) {
        base = dt_mem_next_cell(dt_root_addr_cells, &prop);        (C)
        size = dt_mem_next_cell(dt_root_size_cells, &prop);        (D)

        if (size &&
            early_init_dt_reserve_memory_arch(base, size, nomap) == 0)    (E
)
            pr_debug("Reserved memory: reserved region for node '%s': base %
pa, size %ld MiB\n",
                uname, &base, (unsigned long)size / SZ_1M);
        else
            pr_info("Reserved memory: failed to reserve memory for node '%s'
: base %pa, size %ld MiB\n",
                uname, &base, (unsigned long)size / SZ_1M);

        len -= t_len;
        if (first) {
            fdt_reserved_mem_save_node(node, uname, base, size);
            first = 0;
        }
    }
    return 0;
}
```

(A)

prop –> reg = <0x0 0x6000000 0x0 0x400000>;

(B)

> no-map;

nomap = 1

(C)

> reg = <0x0 0x6000000 0x0 0x400000>;

base = 0x6000000 (96M边界)

(D)

> reg = <0x0 0x6000000 0x0 0x400000>;

size = 0x400000　(4M)

(E)

early_init_dt_reserve_memory_arch(0x6000000, 0x400000, 1)

```
int __init __weak early_init_dt_reserve_memory_arch(phys_addr_t base,
                    phys_addr_t size, bool nomap)
{
    if (nomap)
    return memblock_remove(base, size);
    return memblock_reserve(base, size);
}
```

由于nomap = 1,所以memblock_remove(0x6000000, 0x400000)

nomap = 1，表示这块space不要map，即从整个物理地址中remove。

---

dma pool

```
        linux,cma {
            compatible = "shared-dma-pool";
            reusable;
            size = <0x0 0x10000>;
            alignment = <0x0 0x2000>;
            linux,cma-default;
        };
```

> compatible = "shared-dma-pool";
> 有两个driver handle "shared-dma-pool"
>
> 1. drivers/base/dma-contiguous.c
> 2. drivers/base/dma-coherent.c
>
> in drivers/base/Makefile

```
obj-$(CONFIG_DMA_CMA) += dma-contiguous.o
obj-y           += power/
obj-$(CONFIG_HAS_DMA)  += dma-mapping.o
obj-$(CONFIG_HAVE_GENERIC_DMA_COHERENT) += dma-coherent.o
```

in .config

> CONFIG_HAVE_GENERIC_DMA_COHERENT=y
> CONFIG_DMA_CMA并没有enable，所以 `drivers/base/dma-coherent.c` handle the device tree node。

Note:
要在dtb中支持"shared-dma-pool" node必须enable `CONFIG_OF_RESERVED_MEM` .
in dma-coherent.c

> RESERVEDMEM_OF_DECLARE(dma, "shared-dma-pool", rmem_dma_setup);

RESERVEDMEM_OF_DECLARE()定义了一个struct of_device_id，该structure最终将成为 `__reservedmem_of_table` array的entry。

in vmlinux.lds

> __reservedmem_of_table = .; *(__reservedmem_of_table)*
> (__reservedmem_of_table_end)
> 该array的head由symbol __reservedmem_of_table指向。

in drivers/of/of_reserved_mem.c

```
/**
 * fdt_init_reserved_mem - allocate and init all saved reserved memory regio
ns
 */
void __init fdt_init_reserved_mem(void)
{
    int i;
    for (i = 0; i < reserved_mem_count; i++) {
        struct reserved_mem *rmem = &reserved_mem[i];
        unsigned long node = rmem->fdt_node;
        int len;
        const __be32 *prop;
        int err = 0;

        prop = of_get_flat_dt_prop(node, "phandle", &len);
        if (!prop)
            prop = of_get_flat_dt_prop(node, "linux,phandle", &len);
        if (prop)
            rmem->phandle = of_read_number(prop, len/4);

        if (rmem->size == 0)
            err = __reserved_mem_alloc_size(node, rmem->name,
                        &rmem->base, &rmem->size);
        if (err == 0)
            __reserved_mem_init_node(rmem);
    }
}
```

```
/**
 * res_mem_init_node() - call region specific reserved memory init code
 */
static int __init __reserved_mem_init_node(struct reserved_mem *rmem)
{
    extern const struct of_device_id __reservedmem_of_table[];
    const struct of_device_id *i;

    for (i = __reservedmem_of_table; i < &__rmem_of_table_sentinel; i++) {  ①

        reservedmem_of_init_fn initfn = i->data;
        const char *compat = i->compatible;

        if (!of_flat_dt_is_compatible(rmem->fdt_node, compat))              ②
            continue;

        if (initfn(rmem) == 0) {                                            ③
            pr_info("Reserved memory: initialized node %s, compatible id %s\n",
                rmem->name, compat);
            return 0;
        }
    }
    return -ENOENT;
}
```

①
处理__reservedmem_of_table[]中的成员
②
linux,cma node中的"shared-dma-pool"与dma-coherent.c中的

> RESERVEDMEM_OF_DECLARE(dma, "shared-dma-pool", rmem_dma_setup);
> match。
> ③
> initfn(rmem)

即运行dma-coherent.c/rmem_dma_setup()

```
static int __init rmem_dma_setup(struct reserved_mem *rmem)
{
    unsigned long node = rmem->fdt_node;

    if (of_get_flat_dt_prop(node, "reusable", NULL))        (A)
        return -EINVAL;

#ifdef CONFIG_ARM
    if (!of_get_flat_dt_prop(node, "no-map", NULL)) {        (B)
        pr_err("Reserved memory: regions without no-map are not yet supported\n");
        return -EINVAL;
    }
#endif

    rmem->ops = &rmem_dma_ops;                  (C)

    pr_info("Reserved memory: created DMA memory pool at %pa, size %ld MiB\n",
        &rmem->base, (unsigned long)rmem->size / SZ_1M);
    return 0;
}
```

(A)

```
        linux,cma {
            compatible = "shared-dma-pool";
            reusable;
            size = <0x0 0x10000>;
            alignment = <0x0 0x2000>;
            linux,cma-default;
        };
```

> reusable是必须的

(B)

不能有 `no-map` property

(C)

> rmem->ops = &rmem_dma_ops;
> 使用这里指定的callback。