7.Mbuf库

Mbuf库用于从mempool中分配内存块（mbufs）给dpdk应用程序使用。

rte\_mbuf结构体用于存放数据包或者通用控制信息（通用控制信息由CTRL\_MBUF\_FLAG标识）。Rte\_mbuf还支持扩展存放其他类型的数据。Rte\_mbuf头部设计的尽量小，目前占用两个cache line，其中存放了最常用的字段信息。

7.1 Packet buffers的设计

有两种方式存放packet数据：

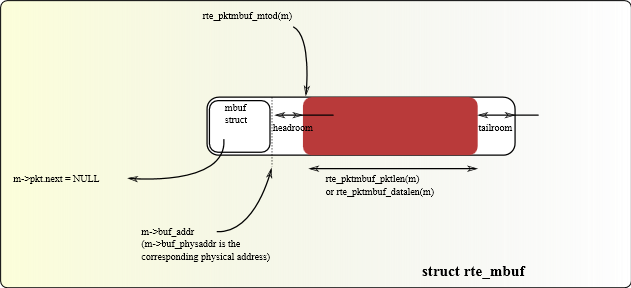
1. 将元数据信息与数据包数据存放在同一块固定大小的内存中
2. 将元数据信息与数据包数据分开存放

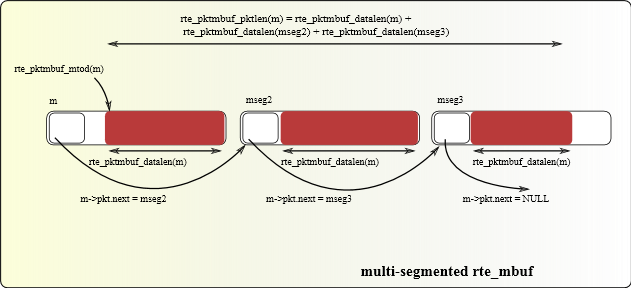
第一种方式的优势在于在分配内存时每个数据包只需要做一次分配的操作。第二种方式将元数据信息与数据包数据分开存放，会更加灵活一些。

Dpdk使用第一种方式，其中元数据包括控制信息，如消息类型、长度、数据段的偏移以及下一个mbuf的指针（当数据需要多个mbuf存放时）

Message buffers可以通过buffer chaining的方式存放巨型帧。

在一个新分配的mbuf中，数据存放在buffer的起始地址偏移RTE\_PKTMBUF\_HEADROOM（默认是128个字节，可以根据实际需要自行调整大小）个字节之后。Message buffers用于存放如控制信息、数据包、事件等信息，mbuf中的buffer 指针也能够指向其他mbuf的数据区或是其他的数据结构。





Mbuf库中有一整套完整的访问函数来操作数据包数据。

7.2 mbuf的存放

Mubf 从Mempool中分配，尽管每个pool中的mbuf允许离散存放在物理内存中以充分利用内存通道来加速访问，但每个mbu都有标识自己所属mempool的字段。当调用rte\_ctrlmbuf\_free和rte\_pktmbuf\_free来释放mbuf时，被释放的mbuf会自动归还到自己的pool中。

7.3 构造

在使用rte\_mempool\_create()函数创建mempool时，ret\_mempool\_create通过回调函数的方式调用rte\_pktmbuf\_init()或rte\_ctrlmbuf\_init()来初始化pool中所有mbuf的相关相关（如mbuf type，origin pool，buffer start address等）。

7.4 分配和释放mbufs

分配mbuf时需要指定从哪个mempool中分配，一个新分配的mbuf，默认有一个segment，数据负载长度为0，数据偏移字段初始化成RTE\_PKTMBUF\_HEADROOM（headroom的长度）。

释放mbuf会将mbuf释放到其所属的mempool，释放后负载数据不会清零，元数据信息也不需要重新初始化。

当一个数据包占用多个mbuf，释放时会将所有的mbuf归还给mempool。

7.5mbuf操作函数

Mbuf 库提供了相关的函数来操作mbuf中的数据：

1. 获取数据长度
2. 获取数据的起始地址
3. 在数据头插入前置数据
4. 在数据尾追加数据
5. 在数据头删除指定大小的数据
6. 在数据末尾删除指定大小的数据

7.6 元数据信息

Mbuf的一些元数据信息有网卡驱动来填充，如vlan、rss的哈希结果、硬件checksum的标志等。Mbuf还存放了input port以及mbufs segment的编号。在buffer chaining中，只有第一个mbuf会包含这些元素据信息。

7.7 直接和间接buf

直接buf指正常的mbuf头和数据存储区在一起的mbuf，间接buf指mbuf元数据中的buffer指针和数据偏移指向另一个mbuf的数据存储区。间接buf用于数据包需要拷贝和分片重组的场景。

Rte\_pktmbuf\_attach()函数用于将一个buffer间接指向一个直接buffer上。当一个间接buffer 在attach到一个直接buffer中时，直接buffer的引用计数会加1；当间接buffer从直接buffer中detach时，直接buffer的引用计数减1。当一个buffer的引用计数为0时，该buffer会释放到mempool中。

使用间接buffer时要注意：1）间接buffer不能attach到另一个间接buffer上。2）当一个buffer需要变成间接buffer时，其引用计数必须等于1，意思就是这个buffer不能已经被其他的间接buffer attach上了。3）已经attach的间接buffer不能直接attach到其他的直接buffer上（除非先dettach，再attach）.

尽管attach/detach操作可以直接调用rte\_pktmbuf\_attach()和rte\_pktmbuf\_detach()函数，不过更为推荐的做法是调用更高层封装的rte\_pktmbuf\_clone()函数，该函数在跨段的数据拷贝中更加安全。

间接buffer实际上也占用了mempool中的内存，使用例子请参考IPV4 Multicast sample application。

7.8 Debug

在debug模式下（CONFIG\_RTE\_MBUF\_DEBUG 开启），mbuf 库中的所有函数会在操作前执行完整性检查（如buffery异常、类型错误等）。

7.9 使用场景

所有的网络应用都是用到了mbufs.