

# Lab 2 - DPDK

---

- 学号: 516030910141
- 姓名: 谢添翼
- 邮箱: lsyhxytj@sjtu.edu.cn

## Part 1

### 1. What' s the purpose of using hugepage?

- 因为数据包需要的缓冲区往往很大。通过使用大页面分配, 性能得以提高, 因为需要的页面更少, 因此转换后占用的TLB条目也更少, 降低了缓存被刷掉的可能, 从而有效提升了TLB命中率从而减少了时间浪费。

### 2. Take examples/helloworld as an example, describe the execution flow of DPDK programs?

1. 初始化基础运行环境, 包括配置初始化, 内存初始化, 终端初始化, 主线程初始化, 建立主从线程通道等等。
2. 多核运行初始化, 并且被初始化的从线程会执行指定函数, 在helloworld中即是执行lcore\_hello函数
3. 主线程等待所有从线程初始化结束后, 执行结束

### 3. Read the codes of examples/skeleton, describe DPDK APIs related to sending and receiving packets.

- `int rte_eth_dev_configure (uint16_t port_id, uint16_t nb_rx_queue, uint16_t nb_tx_queue, const struct rte_eth_conf *eth_conf):` 配置以太网设备
- `int rte_eth_rx_queue_setup (uint16_t port_id, uint16_t rx_queue_id, uint16_t nb_rx_desc, unsigned int socket_id, const struct rte_eth_rxconf *rx_conf, struct rte_mempool *mb_pool):` 为以太网设备设置接收队列
- `int rte_eth_tx_queue_setup (uint16_t port_id, uint16_t tx_queue_id, uint16_t nb_tx_desc, unsigned int socket_id, const struct rte_eth_txconf *tx_conf):` 为以太网设备设置传输队列
- `int rte_eth_dev_start (uint16_t port_id):` 启动该以太网设备
- `void rte_eth_promiscuous_enable (uint16_t port_id) :` 启动混杂模式接收
- `static uint16_t rte_eth_rx_burst (uint16_t port_id, uint16_t queue_id, struct rte_mbuf **rx_pkts, const uint16_t nb_pkts):` 从接收队列中读取数据包
- `static uint16_t rte_eth_tx_burst (uint16_t port_id, uint16_t queue_id, struct rte_mbuf **tx_pkts, uint16_t nb_pkts):` 发送发送队列中的数据包

### 4. Describe the data structure of 'rte\_mbuf

- buf分为三个部分: headroom, data和tailroom。headroom一般用来存放用户自己针对于mbuf的一些描述信息, 一般保留给用户使用。data区域一般指的是地址区间在 `data_off+buf_addr` 到 `data_off+data_len+buf_addr` 即, `data_len` 就是这段数据的长短。tailroom也是为了方便解封报文设置的。

## Part 2

## wireshark拦截发包：

正在捕获 VMware Network Adapter VMnet2

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

应用显示过滤器:  表达式:

| No. | Time     | Source        | Destination  | Protocol | Length | Info               |
|-----|----------|---------------|--------------|----------|--------|--------------------|
| 1   | 0.000000 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 2   | 0.000059 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 3   | 0.000075 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 4   | 0.000088 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 5   | 0.000101 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 6   | 0.000114 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 7   | 0.000140 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 8   | 0.000152 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |
| 9   | 0.000165 | 192.168.80.10 | 192.168.80.6 | UDP      | 114    | 2333 → 2333 Len=64 |

> Frame 5: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0  
 > Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: Vmware\_c0:00:02 (00:50:56:c0:00:02)  
 > Internet Protocol Version 4, Src: 192.168.80.10, Dst: 192.168.80.6  
 > User Datagram Protocol, Src Port: 2333, Dst Port: 2333  
 > Data (64 bytes)

```

0000  00 50 56 c0 00 02 00 00 00 00 00 00 00 08 00 45 00  .PV.....E.
0010  00 5c 00 00 00 00 40 11 59 30 c0 a8 50 0a c0 a8  .\....@.Y0..P...
0020  50 06 09 1d 09 1d 00 48 00 00 00 00 00 04 00 00  P.....H.....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0070  00 00  ..
  
```

VMware Network Adapter VMnet2: <live capture in progress> 分组: 32 · 已显示: 32 (100.0%) 配置: Default

正确性可通过查看包头的信息以及数据内容来判断，首先发送mac地址与代码中填写的相符，源IP地址与目标IP地址均与代码中填写的对应，发送端口与接收端口均为2333，即代码中设置的。且包中数据根据代码逻辑应分别填写了0-31的数字，依次检查这32个包，发现是正确的。