

**Q1:**

一个2M的大页相比4K的页，可以在一条TLB映射更多的虚拟地址，减少TLBmiss，较少访存的时间。  
缺点是使用的连续内存较大，较为浪费物理内存。

**Q2:**

首先主线程调用rte\_eal\_init初始化。主线程然后调用RTE\_LCORE\_FOREACH\_WORKER宏，遍历每一个逻辑核，调用rte\_eal\_remote\_launch在这个核上开启从线程，执行lcore\_hello。然后主线程自己也执行lcore\_hello。然后主线程调用rte\_eal\_mp\_wait\_lcore等待从线程运行完毕，最后返回。

**Q3:**

## 1. rte\_eth\_tx\_burst(port\_id, queue\_id, tx\_pkts, nb\_pkts)批量发送包

port\_id 为发包的接口号，queue\_id 为发送队列的id。tx\_pkts 为一个指向 struct rte\_mbuf的指针数组，用于存放接收包的内容，nb\_pkts为tx\_pkts向量的长度。返回值为实际发送包的数量。该函数会释放已经发送的mbuf的内存。

## 2. rte\_eth\_tx\_burst 和tx\_burst相似，用来接收包。

**Q4:**

rte\_mbuf是用于放网络包的缓存。

它通过mem\_pool中获得，大小总数等参数在初始化mem\_pool时设置。

rte\_mbuf头部有两个缓存行大小的元数据，用来保存mbuf的数据，包括源池，next指针等

然后是headroom长的空白，然后是数据头指针，数据尾指针指向的初始位置，后面全是tailroom。

需要放置网络包时，需要通过append,alloc,trim, adj等函数改变rte\_buf的位置，确认成功后才能复制数据。

超过一个mbuf长度的数据可以写在另一个mbuf中，并用chain函数链接起来，这会设置元数据中的next。可以通过pkt\_len获取被链接起来的总数据大小

rte\_eth\_tx\_burst 会自动free发送成功的mbuf，其他情况需要通过free函数手动释放。

用gen\_data产生测试数据，为连续的"hi dpdk!"，最后一个为"\0"

|    |              |         |         |      |   |  |
|----|--------------|---------|---------|------|---|--|
| 1  | 0.000000000  | 1.1.1.1 | 1.1.1.1 | UDP  | 235 80 → 80 Len=185                                     |  |
| 2  | 26.467535052 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=0, ID=0  |  |
| 3  | 26.467550920 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=1496, I  |  |
| 4  | 26.467553575 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=2992, I  |  |
| 5  | 26.467556936 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=4488, I  |  |
| 6  | 26.467559642 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=5984, I  |  |
| 7  | 26.467562351 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=7480, I  |  |
| 8  | 26.467565040 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=8976, I  |  |
| 9  | 26.467567819 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=10472, I |  |
| 10 | 26.467570384 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=11968, I |  |
| 11 | 26.467573686 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=13464, I |  |
| 12 | 26.467576339 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=14960, I |  |
| 13 | 26.467580127 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=16456, I |  |
| 14 | 26.467581913 | 1.1.1.1 | 1.1.1.1 | IPv4 | 1530 Fragmented IP protocol (proto=UDP 17, off=17952, I |  |
| 15 | 26.467585914 | 1.1.1.1 | 1.1.1.1 | UDP  | 579 80 → 80 Len=19985                                   |  |

```

▶ Frame 1: 235 bytes on wire (1880 bits), 235 bytes captured (1880 bits) on interface vmnet1, id 0
▶ Ethernet II, Src: VMware_8e:ae:f9 (00:0c:29:8e:ae:f9), Dst: VMware_8e:ae:f9 (00:0c:29:8e:ae:f9)
▶ Internet Protocol Version 4, Src: 1.1.1.1, Dst: 1.1.1.1
▶ User Datagram Protocol, Src Port: 80, Dst Port: 80
▼ Data (185 bytes)
  Data: 6869206470646b216869206470646b216869206470646b21...
  [Length: 185]

```

```

0000 00 0c 29 8e ae f9 00 0c 29 8e ae f9 08 00 45 00  ..).... ).....E.
0010 00 dd 00 6f 00 00 40 11 9e 75 01 01 01 01 01  ..o..@.u.....
0020 01 01 00 50 00 50 00 c1 8f 9a 68 69 20 64 70 64  ...P.P..hi dpd
0030 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0040 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0050 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0060 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0070 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0080 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
0090 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
00a0 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
00b0 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
00c0 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
00d0 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64  k!hi dpd k!hi dpd
00e0 6b 21 68 69 20 64 70 64 6b 21 00  k!hi dpd k!

```

```
▶ Frame 15: 579 bytes on wire (4632 bits), 579 bytes captured (4632 bits) on interface vmmnet1, id
▶ Ethernet II, Src: VMware_8e:ae:f9 (00:0c:29:8e:ae:f9), Dst: VMware_8e:ae:f9 (00:0c:29:8e:ae:f9)
▶ Internet Protocol Version 4, Src: 1.1.1.1, Dst: 1.1.1.1
▶ User Datagram Protocol, Src Port: 80, Dst Port: 80
▼ Data (19985 bytes)
  Data: 6869206470646b216869206470646b216869206470646b21...
  [Length: 19985]
```

| Offset | Hex   | ASCII             |
|--------|---|-------------------|
| 0000   | 00 0c 29 8e ae f9 00 0c 29 8e ae f9 08 00 45 00 | ..).....).....E.  |
| 0010   | 02 35 00 6f 09 7f 40 11 c7 6a 01 01 01 01 01 01 | .5.o..@. .j.....  |
| 0020   | 01 01 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | ..hi dpd k!hi dpd |
| 0030   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0040   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0050   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0060   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0070   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0080   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0090   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00a0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00b0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00c0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00d0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00e0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 00f0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0100   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0110   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0120   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0130   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0140   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0150   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0160   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0170   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0180   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0190   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01a0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01b0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01c0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01d0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01e0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 01f0   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0200   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0210   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0220   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0230   | 6b 21 68 69 20 64 70 64 6b 21 68 69 20 64 70 64 | k!hi dpd k!hi dpd |
| 0240   | 6b 21 00  | k!..              |

Frame (579 bytes) Reassembled IPv4 (19993 bytes)

包的长度小于一个MTU，data中为测试数据，外部是一个udp包头，一个ipv4包头和一个ether包头。ipv4的checksum为ip包头的checksum，udp的checksum为ip的地址加udp包头加数据的checksum。只用了一个mbuf，因为它比MTU大

对于较长的数据，ip层会fragmentation，ip里的offset和flag用于标记顺序

可以目测结果是正确的。