

Distributed System Lab2 Report

516030910293 姚子航

Part1

Q1: What's the purpose of using hugepage?

A: Using huge page reduce the number of page table entry because now an entry can map 4MB memory. Then the hit rate of TLB may increase significantly, which accelerates memory access.

Q2: Take examples/helloworld as an example, describe the execution flow of DPDK programs?

1. Initialize basic running environment using function "rte_eal_init"
2. Initialize multi-core running using function "rte_eal_remote_launch"
Then lcore_hello will be called on every slave logical core.
3. Call lcore_hello on master logical core
4. Wait all logical cores to end up executing using function "rte_eal_mp_wait_lcore"

Q3: Read the codes of examples/skeleton, describe DPDK APIs related to sending and receiving packets.

A:

```
{  
Create mempool API: struct rte_mempool *rte_pktmbuf_pool_create(const char *name,  
unsigned n, unsigned cache_size, uint16_t priv_size, uint16_t data_room_size,  
int socket_id)
```

This function create a memory pool for storing rte_mbuf.
}

```
{  
Receive API: static inline uint16_t rte_eth_rx_burst(uint8_t port_id, uint16_t  
queue_id, struct rte_mbuf **rx_pkts, const uint16_t nb_pkts)
```

Send API: static inline uint16_t rte_eth_tx_burst(uint8_t port_id, uint16_t
queue_id, struct rte_mbuf **tx_pkts, uint16_t nb_pkts)

1st parameter: port
2nd parameter: queue identifier
3rd parameter: packet buffer
4th parameter: the number of packets we hope to send or receive

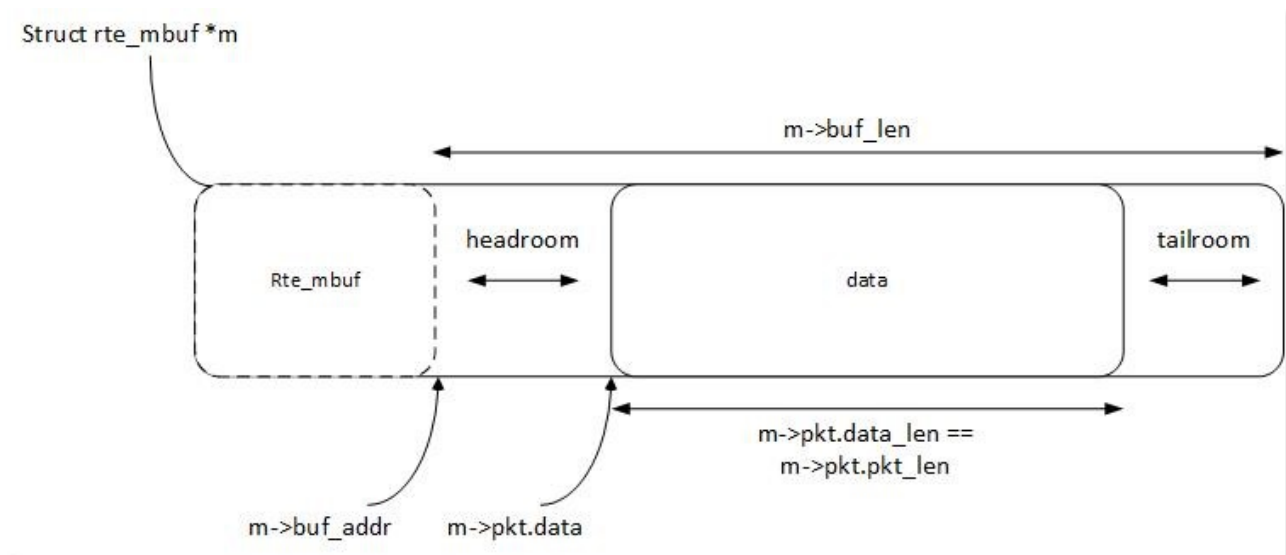
return value: the number packets we actually send or receive.
}

```
{  
Allocate API: struct rte_mbuf *rte_pktmbuf_alloc(struct rte_mempool *mp);
```

This function apply a rte_mbuf from memory pool.
}

Q4: Describe the data structure of 'rte_mbuf'.

A: "rte_mbuf" structure describes a packet mbuf, which mainly contains a data buffer. When we use rte_pktmbuf_alloc function to allocate a new rte_buf, dpdk will reserve some headroom. If we want to fill data in it, we could use rte_pktmbuf_prepend or rte_pktmbuf_append to increase the size of headroom or tailroom. The following image gives us a more intuitive description.



Part 2

Action:

- 1.Install wireshark in host OS
- 2.Make and start program “send_udp”
- 3.Open wireshark and select to catch packets on vmnet2(the virtual NIC I just add)
- 4.Detect many packets sent by dpdk.

Screenshot:

A screenshot of an Ubuntu desktop environment. The background is a purple and blue geometric pattern. On the left side, there is a vertical dock containing several application icons: Dash, Home Folder, Files, Firefox, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, LibreOffice Draw, Amazon, Settings, and a terminal icon. At the top of the screen, a status bar shows "Ubuntu Desktop" on the left and system information (signal strength, Wi-Fi, battery at 90%, time 9:11 AM, and a settings gear icon) on the right. A terminal window titled "allen@ubuntu: ~/dppk-stable-16.11.8/examples/dppk-lab" is open in the center-right. It displays the output of running a program called "send_udp". The output includes detection of 1 local core(s), no free hugepages, VFIO support probing, PCI device information, probe driver details, port number 1, MAC address, and a loop of sending packets. The prompt returns to the shell after the ^C interrupt.

```
allen@ubuntu:~/dppk-stable-16.11.8/examples/dppk-lab$ sudo ./build/send_udp
EAL: Detected 1 lcore(s)
EAL: No free hugepages reported in hugepages-1048576kB
EAL: Probing VFIO support...
EAL: PCI device 0000:02:01.0 on NUMA socket -1
EAL:   probe driver: 8086:100f net_e1000_em
port num:1
Port 0 MAC: 00 0c 29 e5 32 13

Core 0 forwarding packets. [Ctrl+C to quit]
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
Send packet:!!!!
^C
allen@ubuntu:~/dppk-stable-16.11.8/examples/dppk-lab$
```

