DPDK\_L3FWD性能测试

## 一、测试环境：

|  |  |
| --- | --- |
| DPDK\_L3FWD | CPU ：Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz |
| Mem : 187G(Hugepage Mem:128) |
| OS : CentOS Linux release 7.6.1810 (Core) |
| Sprient | SPT-C50 |

10G

DPDK\_L3FWD

Sprient

## 二、测试说明：

本实验的目的是测试DPDK\_L3FWD程序性能的网卡扩展能力与核心扩展能力。

DPDK\_L3FWD程序绑定1张到4张网卡，每张网卡对应1个、2个或4个pipeline程序（本实验中每个pipeline程序独占一个FE核，故也可以说每张网卡绑定1个、2个或4个核）。pipeline程序的工作流程为：查找路由表后，将数据包从源端口转发回去（因此需要设计数据包地址与路由表）。因此，每个网卡的数据包反射的速率极限，表征了DPDK\_L3FWD程序的数据包处理效率。

物理环境中，将思博伦机器的4个10G口与DPDK\_L3FWD程序的四个10G网卡相连，4个口以线速打IPV6包（包大小为98byte），实验中观察每个思博伦口的收包速率。

## 三、测试过程：

### 编译准备l3fwd运行环境

1. checkout DPDK程序代码。这里我们使用seasw1.0中的dpdk文件夹，dpdk版本为18.11

2. 编译dpdk

cd dpdk

source dpdk.rc

make clean

make install T=x86\_64-native-linuxapp-gcc -j64

3. 进入l3fwd环境，编译

cd dpdk/examples/l3fwd/

make install

4. 此时会在l3fwd/build下生成可执行文件l3fwd

### 编辑l3fwd路由表与思博伦仪器发包配置

为了达到思博伦1号port发出的包，到达服务器0号port，经过l3fwd程序处理后，再从0号port发回来的效果（后续端口以此类推）（思博伦仪器UI网卡从1开始数，dpdk程序配置中的网卡从0开始数），我们需要编辑l3fwd路由表与思博伦仪器发包配置。

#### 编辑l3fwd路由表

l3fwd程序使用的路由表由目录下的l3fwd\_lpm.c配置，故修改路由表后需冲洗编译l3fwd程序。

cd l3fwd

make clean

make install

本实验中，将/home/majd/seasw1.0/dpdk/examples/l3fwd/l3fwd\_lpm.c的第42行由

|  |
| --- |
| static struct ipv6\_l3fwd\_lpm\_route ipv6\_l3fwd\_lpm\_route\_array[] = {  {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 0},  {{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 1},  {{3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 2},  {{4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 3},  {{5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 4},  {{6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 5},  {{7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 6},  {{8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 7},  }; |

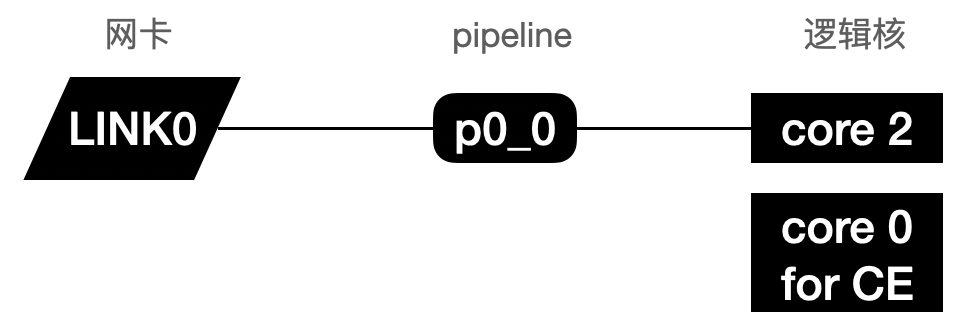
修改（删减）为

|  |
| --- |
| static struct ipv6\_l3fwd\_lpm\_route ipv6\_l3fwd\_lpm\_route\_array[] = {  {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 0},  {{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 1},  {{3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 2},  {{4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, 48, 3}  }; |

#### 编辑思博伦仪器发包配置

根据路由表和我们的设计目标，故设置思博伦port1发出的包的源IPv6地址为1000::1、port2发出的包的源IPv6地址为2000::1、port3发出的包的源IPv6地址为1000::1、port4发出的包的源IPv6地址为4000::1。即控制地址的前两个字节。

### 单网卡、单核心（1core\_1port）



DPDK\_L3FWD程序绑定网卡LINK0，pipeline程序运行在FE核上，负责网卡LINK0的数据包处理。运行测试用到的命令：

|  |
| --- |
| ./build/l3fwd -l 2 -- -P -L -p 0x1 --config="(0,0,2)" |

其中，

-l 2 指定dpdk程序调用的逻辑核，这里仅使用2号逻辑核。这里采用的是列举逻辑核号的方式。也可以采用-c 0x2的掩码表示方式

-- 分割EAL options与l3fwd options

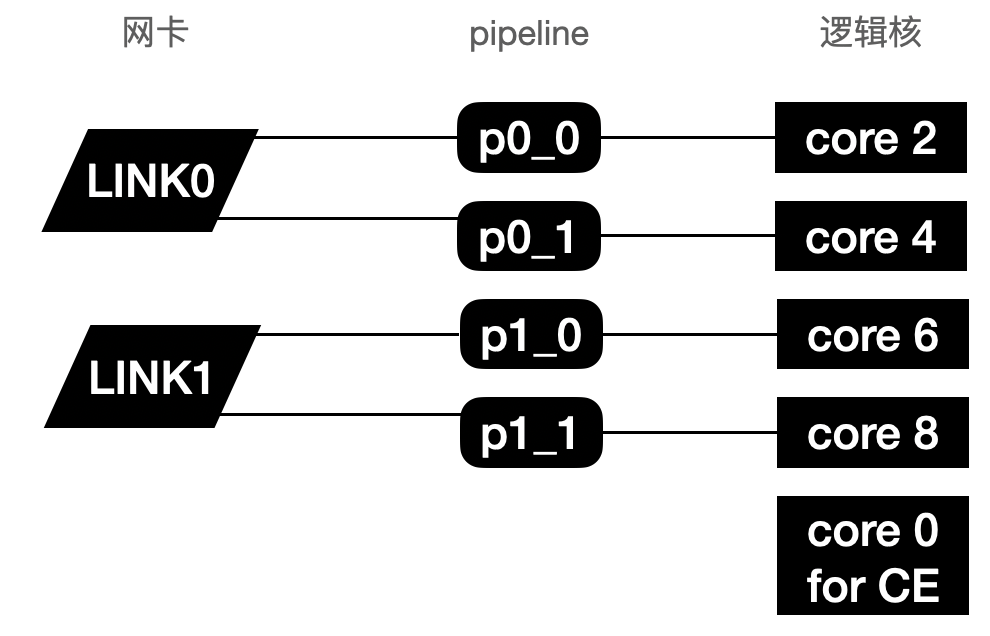
-P 使dpdk网卡处于混杂模式，即所有目的mac地址的数据包都会收包交给dpdk进行处理。如果不加此选项，则只有目的mac地址与网卡相匹配的数据包才会进入到dpdk程序进行处理

-L 指示程序采用“最大长度匹配”的路由表进行查表转发。

-p 0x1 指示程序调用网卡port，使用掩码表示。这里仅使用0号网卡。

--config="(0,0,2)" 指示dpdk程序的网卡、序列、逻辑核分配，这里表达的是（0号网卡，0号序列，2号逻辑核）。其采用(port,queue,lcore)的格式，列举之间用逗号隔开，整体用双引号括起。

### 双网卡、四核心（4core\_2port）



DPDK\_L3FWD程序绑定网卡LINK0、LINK1，pipeline程序p0\_0、p0\_1负责LINK0网卡的流量处理，pipeline程序p1\_0、p1\_1负责LINK1网卡的流量处理。p0\_0、p0\_1、p1\_0、p1\_1各独占一个FE核运行。由网卡到多个pipeline的流量输入由RSS分流队列实现。运行测试用到的命令：

|  |
| --- |
| ./build/l3fwd -l 2,4,6,8 -- -P -L -p 0x3 --config="(0,0,2),(0,1,4),(1,0,6),(1,1,8)" |

其中，

-l 2,4,6,8 指定dpdk程序调用的逻辑核，这里使用2,4,6,8号逻辑核。这里采用的是列举逻辑核号的方式。也可以采用-c 0x154的掩码表示方式

-p 0x3 指示程序调用网卡port，使用掩码表示。这里使用0、1号网卡。

--config="(0,0,2),(0,1,4),(1,0,6),(1,1,8)" 指示dpdk程序的网卡、序列、逻辑核分配。其采用(port,queue,lcore)的格式，列举之间用逗号隔开，整体用双引号括起。因为rss分流的存在，所以对于服务器的0号网卡，有2号逻辑核的(0,0,2)的队列0与4号逻辑核的(0,1,4)的队列1之分。

## 四、测试结果

以此类推，我们进行了DPDK\_L3FWD程序绑定1张到4张网卡，每张网卡对应1个、2个或4个逻辑核的多次实验。实验结果如下：

dpdk=18.11

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FE逻辑核个数(FE lcore) | 绑定网卡数(port) | 思博伦网口发包线速率Tx L1 Rate(Gbps) | Port 1 Rx L1 Rate(Gbps) | Port 2 Rx L1 Rate | Port 3 Rx L1 Rate | Port 4 Rx L1 Rate |
| 1 | 1 | 9.83 | 9.83（线速） |  |  |  |
| 2 | 1 | 9.83（线速） |  |  |  |
| 4 | 1 | 9.83（线速） |  |  |  |
| 1+1 | 2 | 9.83（线速） | 9.83（线速） |  |  |
| 2+2 | 2 | 9.83（线速） | 9.83（线速） |  |  |
| 4+4 | 2 | 9.83（线速） | 9.83（线速） |  |  |
| 1+1+1 | 3 | 9.83（线速） | 9.83（线速） | 9.83（线速） |  |
| 2+2+2 | 3 | 9.83（线速） | 9.83（线速） | 9.83（线速） |  |
| 4+4+4 | 3 | 9.83（线速） | 9.83（线速） | 9.83（线速） |  |
| 1+1+1+1 | 4 | 7.83 | 7.83 | 7.83 | 7.83 |
| 2+2+2+2 | 4 | 7.82 | 7.82 | 7.82 | 7.82 |
| 4+4+4+4 | 4 | 7.84 | 7.84 | 7.84 | 7.84 |

dpdk版本使用19.11时，测试结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FE逻辑核个数(FE lcore) | 绑定网卡数(port) | 思博伦网口发包线速率Tx L1 Rate(Gbps) | Port 1 Rx L1 Rate(Gbps) | Port 2 Rx L1 Rate | Port 3 Rx L1 Rate | Port 4 Rx L1 Rate |
| 1+1+1+1 | 4 | 9.83 | 8.55 | 8.55 | 8.55 | 8.55 |
| 2+2+2+2 | 4 | 8.53 | 8.53 | 8.53 | 8.53 |
| 4+4+4+4 | 4 | 8 | 8 | 8 | 8 |

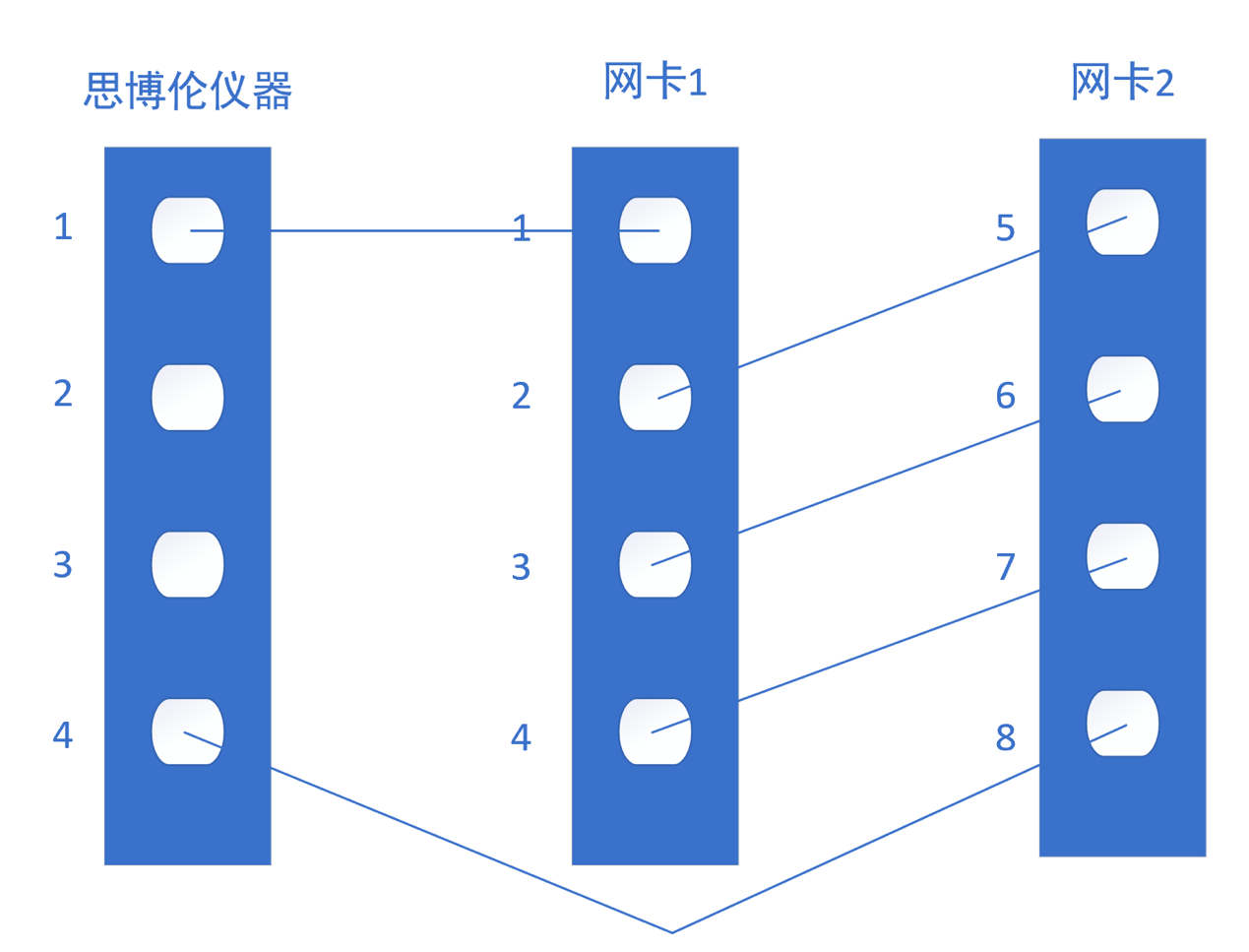
dpdk版本使用19.11，且使用两张网卡，每张网卡接两根线时，端口转发方式为“反射”模式（同以上实验），测试结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| FE逻辑核个数(FE lcore) | 绑定网卡数(port) | 思博伦网口发包线速率Tx L1 Rate(Gbps) | Port 1 Rx L1 Rate(Gbps) | Port 2 Rx L1 Rate | Port 3 Rx L1 Rate | Port 4 Rx L1 Rate |
| 1+1+1+1 | 4 | 9.83 | 9.83（线速） | 9.83（线速） | 9.83（线速） | 9.83（线速） |
| 2+2+2+2 | 4 | 9.83（线速） | 9.83（线速） | 9.83（线速） | 9.83（线速） |
| 4+4+4+4 | 4 | 9.83（线速） | 9.83（线速） | 9.83（线速） | 9.83（线速） |

这说明，在4core\_4port的测试环境下，限制程序达到40Gbps吞吐的主要是网卡的PCIe带宽，当使用两张网卡各接两个口时，dpdk\_l3fwd可以达到40Gbps的吞吐。

双网卡八端口的80G吞吐测试（使用dpdk\_l2fwd）：

将思博伦仪器、网卡1、网卡2按如下方式连接：



思博伦仪器的1口、4口各打10Gbps的流量。dpdk\_l2fwd程序的转发逻辑为：port1收到的包转发给port2，port2收到的包转发给port1，以此类推，port3port4，port5port6，port7port8也各为一组。这样便可以实现dpdk\_l2fwd程序处理80Gbps吞吐的测试效果，通过观察思博伦1口和4口的收包速率即可。

## 五、附录

本文件举例中所用的代码和配置文件。

### 服务器cpu隔离与大页内存分配情况

|  |
| --- |
| [root@seasw ~]# cat /proc/cmdline  BOOT\_IMAGE=/vmlinuz-3.10.0-957.el7.x86\_64 root=UUID=f073d92a-9a31-45db-8651-8e16cb9e2426 ro crashkernel=auto rhgb quiet  default\_hugepagesz=1G hugepagesz=1G hugepages=128  isolcpus=1,2,3,4,6,8,10,12,14,16,18,20,22,24,26,28,30 nohz\_full=1,2,3,4,6,8,10,12,14,16,18,20,22,24,26,28,30 rcu\_nocbs=1,2,3,4,6,8,10,12,14,16,18,20,22,24,26,28,30 |

### 各测试例的DPDK\_L3FWD执行命令

1core\_1prot

./build/l3fwd -l 2 -- -P -L -p 0x1 --config="(0,0,2)"

2core\_1port

./build/l3fwd -l 2,4 -- -P -L -p 0x1 --config="(0,0,2),(0,1,4)"

4core\_1port

./build/l3fwd -l 2,4,6,8 -- -P -L -p 0x1 --config="(0,0,2),(0,1,4),(0,2,6),(0,3,8)”

2core\_2port

./build/l3fwd -l 2,4 -- -P -L -p 0x3 --config=“(0,0,2),(1,0,4)”

4core\_2port

./build/l3fwd -l 2,4,6,8 -- -P -L -p 0x3 --config=“(0,0,2),(0,1,4),(1,0,6),(1,1,8)”

8core\_2port

./build/l3fwd -l 2,4,6,8,10,12,14,16 -- -P -L -p 0x3 --config=“(0,0,2),(0,1,4),(0,2,6),(0,3,8),(1,0,10),(1,1,12),(1,2,14),(1,3,16)”

3core\_3port

./build/l3fwd -l 2,4,6 -- -P -L -p 0x7 --config="(0,0,2),(1,0,4),(2,0,6)"

6core\_3port

./build/l3fwd -l 2,4,6,8,10,12 -- -P -L -p 0x7 --config=“(0,0,2),(0,1,4),(1,0,6),(1,1,8),(2,0,10),(2,1,12)”

12core\_3port

./build/l3fwd -l 2,4,6,8,10,12,14,16,18,20,22,24 -- -P -L -p 0x7 --config=“(0,0,2),(0,1,4),(0,2,6),(0,3,8),(1,0,10),(1,1,12),(1,2,14),(1,3,16),(2,0,18),(2,1,20),(2,2,22),(2,3,24)”

4core\_4port

./build/l3fwd -l 2,4,6,8 -- -P -L -p 0xf --config="(0,0,2),(1,0,4),(2,0,6),(3,0,8)"

8core\_4port

./build/l3fwd -l 2,4,6,8,10,12,14,16 -- -P -L -p 0xf --config="(0,0,2),(0,1,4),(1,0,6),(1,1,8),(2,0,10),(2,1,12),(3,0,14),(3,1,16)"

16core\_4port

./build/l3fwd -c 0x155555554 -- -P -L -p 0xf --config="(0,0,2),(0,1,4),(0,2,6),(0,3,8),(1,0,10),(1,1,12),(1,2,14),(1,3,16),(2,0,18),(2,1,20),(2,2,22),(2,3,24),(3,0,26),(3,1,28),(3,2,30),(3,3,32)"

./build/l3fwd -c 0x55555556 -- -P -L -p 0xf --config="(0,0,2),(0,1,4),(0,2,6),(0,3,8),(1,0,10),(1,1,12),(1,2,14),(1,3,16),(2,0,18),(2,1,20),(2,2,22),(2,3,24),(3,0,26),(3,1,28),(3,2,30),(3,3,1)"

注：16core\_4port测试中的FE核跨numa。