**Linux CentOS下的DPDK安装配置**

编 号：

版本/修订：

编 制 荣涛

审 核

会 签

批 准

**文档修订历史**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本** | **作者** | **版本变化对象** | **变化内容描述** | **审核人** | **批准人** | **修订日期** |
| V1.0 | 荣涛 | 创建 | 创建文档 |  |  | 2020.07.03 |
| V1.0 | 荣涛 | 添加 | 配置 |  |  | 2020.07.06 |

# 引言

## 概述

DPDK的主要目的就是为数据面快速报文处理应用程序提供一个简洁完整的框架。用户可以通过代码来理解其中使用的一些技术，构建自己的应用程序或添加自己的协议栈。

## 缩略词

DPDK：Date Plane Development Kit 数据面开发工具

EAL：Environment Abstract Layer 环境抽象层

RTE：

Pipeline：流水线

NUMA：Non-Uniform Memory Access

## 参考文档

1. 《[Linux平台上DPDK入门指南](https://dpdk-docs.readthedocs.io/en/latest/linux_gsg/index.html)》
2. 《[DPDK编程指南](https://github.com/lockless/dpdk)》
3. 《DPDK-Programmer’s Guide》
4. 《深入浅出DPDK》

# DPDK安装配置

本章节介绍DPDK基于64位X86服务器的GCC编译器的配置，编译安装。

## HugePages配置

### 配置页数

查看HugePage页大小

|  |
| --- |
| [root@localhost ~]# cat /proc/meminfo | grep HugePage -i  HugePages\_Total: 20  HugePages\_Free: 20  HugePages\_Rsvd: 0  HugePages\_Surp: 0  Hugepagesize: 1048576 kB |

这里显示每页HugePage大小为1048576 kB(1G)，共20页，20可用。当没有大页内存可用时，对于单节点系统，加入需要100页，使用

|  |
| --- |
| # echo 100 > /sys/kernel/mm/hugepages/hugepages-1048576kB/nr\_hugepages |

如果是NUMA设备，分别在指定的节点中

|  |
| --- |
| # echo 100 > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr\_hugepages  # echo 100 > /sys/devices/system/node/node1/hugepages/hugepages-1048576kB/nr\_hugepages |

以我的测试环境（10.170.7.166）为例，配置了20页HugePages，结果为

|  |
| --- |
| # cat /sys/kernel/mm/hugepages/hugepages-1048576kB/nr\_hugepages  20  # cat /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr\_hugepages  10  # cat /sys/devices/system/node/node1/hugepages/hugepages-1048576kB/nr\_hugepages  10 |

### 挂载

一旦预留了HugePages内存，为了使DPDK使用该内存，运行

|  |
| --- |
| # mkdir /mnt/huge  # mount –t hugetlbfs nodev /mnt/huge |

查看挂载结果

|  |
| --- |
| # cat /proc/mounts | grep huge  cgroup /sys/fs/cgroup/hugetlb cgroup rw,nosuid,nodev,noexec,relatime,hugetlb 0 0  hugetlbfs /dev/hugepages hugetlbfs rw,seclabel,relatime 0 0 |

## 安装DPDK及源码

### 解压源码包

下载最新的DPDK安装包，进行解压：

|  |
| --- |
| # tar xJf dpdk-20.05.tar.xz |

将解压的文件移动至/opt目录下

|  |
| --- |
| # mv dpdk-20.05 /opt |

### 目标环境安装

利用DPDK的用户工具脚本

|  |
| --- |
| # ./usertools/dpdk-setup.sh  …  [37] x86\_64-native-linuxapp-clang  [38] x86\_64-native-linuxapp-gcc  [39] x86\_64-native-linuxapp-icc  [40] x86\_64-native-linux-clang  [41] x86\_64-native-linux-gcc  [42] x86\_64-native-linux-icc  …  [62] Exit Script  Option: |

键入“x86\_64-native-linuxapp-gcc”对应的“38”，执行结束后会在编译目录生成文件夹“x86\_64-native-linuxapp-gcc”。这时可以设置环境变量

|  |
| --- |
| export RTE\_SDK=/opt/dpdk-20.05  export RTE\_TARGET=x86\_64-native-linuxapp-gcc |

在文件夹“/opt/dpdk-20.05/x86\_64-native-linuxapp-gcc”中存在隐藏配置文件“.config”，在编译过程中未生成kmod文件夹及ko文件，如下“kmod”文件夹

|  |
| --- |
| # pwd  /opt/dpdk-20.05/x86\_64-native-linuxapp-gcc  # ls  app build include kmod lib Makefile |

编辑配置文件

|  |
| --- |
| # vim x86\_64-native-linuxapp-gcc/.config |

搜索UIO得到

|  |
| --- |
| CONFIG\_RTE\_EAL\_IGB\_UIO=n |

将其改为

|  |
| --- |
| CONFIG\_RTE\_EAL\_IGB\_UIO=y |

再次进行配置，即可生成kmod文件夹。

### 建立Linux环境

向内核中插入驱动

再次执行

|  |
| --- |
| # ./usertools/dpdk-setup.sh |

执行UIO的编号，我这里是45

|  |
| --- |
| [45] Insert IGB UIO module  …  Option: 45  Unloading any existing DPDK UIO module  Loading DPDK UIO module  Press enter to continue ... |

执行VFIO的编号，我这里是46

|  |
| --- |
| [46] Insert VFIO module  Option: 46  Unloading any existing VFIO module  Loading VFIO module  chmod /dev/vfio  OK  Press enter to continue ... |

执行KNI的编号，我这里是47

|  |
| --- |
| [47] Insert KNI module  …  Option: 47  Unloading any existing DPDK KNI module  Loading DPDK KNI module  Press enter to continue ... |

当发现KNI在加载ko文件失败时，再次编辑“.config”文件，搜索“KNI”，将

|  |
| --- |
| CONFIG\_RTE\_KNI\_KMOD=n |

改为

|  |
| --- |
| CONFIG\_RTE\_KNI\_KMOD=y |

重新编译

|  |
| --- |
| # cd /opt/dpdk-20.05/x86\_64-native-linuxapp-gcc #根据之前的设定  # make # 漫长的等待  # ls kmod  igb\_uio.ko rte\_kni.ko |

发现已经生成KNI的ko文件。再次执行上述步骤，即可完成KNI的模块加载。

### 绑定网卡

查看网卡信息，

|  |
| --- |
| # usertools/dpdk-devbind.py --status-dev net  Network devices using kernel driver  ===================================  0000:18:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em1 drv=tg3 unused=igb\_uio,vfio-pci \*Active\*  0000:18:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em2 drv=tg3 unused=igb\_uio,vfio-pci  0000:19:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em3 drv=tg3 unused=igb\_uio,vfio-pci  0000:19:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em4 drv=tg3 unused=igb\_uio,vfio-pci  Other Network devices  =====================  0000:3b:00.0 'Ethernet 10G 2P X520 Adapter 154d' unused=ixgbe,igb\_uio,vfio-pci  0000:3b:00.1 'Ethernet 10G 2P X520 Adapter 154d' unused=ixgbe,igb\_uio,vfio-pci |

或者在总线中查找

|  |
| --- |
| # lspci | grep net -i  18:00.0 Ethernet controller: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe  18:00.1 Ethernet controller: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe  19:00.0 Ethernet controller: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe  19:00.1 Ethernet controller: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe  3b:00.0 Ethernet controller: Intel Corporation Ethernet 10G 2P X520 Adapter (rev 01)  3b:00.1 Ethernet controller: Intel Corporation Ethernet 10G 2P X520 Adapter (rev 01) |

绑定网卡

|  |
| --- |
| # usertools/dpdk-devbind.py --bind=igb\_uio 0000:3b:00.0 0000:3b:00.1 |

当正确执行上述指令后，应该没有异常输出。

再次查看网卡状态

|  |
| --- |
| # usertools/dpdk-devbind.py --status-dev net  Network devices using DPDK-compatible driver  ============================================  0000:3b:00.0 'Ethernet 10G 2P X520 Adapter 154d' drv=igb\_uio unused=ixgbe,vfio-pci  0000:3b:00.1 'Ethernet 10G 2P X520 Adapter 154d' drv=igb\_uio unused=ixgbe,vfio-pci  Network devices using kernel driver  ===================================  0000:18:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em1 drv=tg3 unused=igb\_uio,vfio-pci \*Active\*  0000:18:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em2 drv=tg3 unused=igb\_uio,vfio-pci  0000:19:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em3 drv=tg3 unused=igb\_uio,vfio-pci  0000:19:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em4 drv=tg3 unused=igb\_uio,vfio-pci |

说明网卡绑定成功。此时用“ifconfig”指令是看不到网卡信息的。但是用“lspci”仍旧可查。

# DPDK测试

## HelloWorld

在源码包中，文件夹“examples”中给出了丰富的实例程序，以“helloworld”程序为例，

首先进行编译

|  |
| --- |
| # pwd  /opt/dpdk-20.05/examples/helloworld  # make  CC main.o  LD helloworld  INSTALL-APP helloworld  INSTALL-MAP helloworld.map |

首先查看大页内存使用情况

|  |
| --- |
| # cat /proc/meminfo | grep HugePage -i  HugePages\_Total: 20  HugePages\_Free: 20  HugePages\_Rsvd: 0  HugePages\_Surp: 0  Hugepagesize: 1048576 kB |

此时大页内存均为被申请。运行程序

|  |
| --- |
| # ./build/helloworld  EAL: Detected 40 lcore(s)  EAL: Detected 2 NUMA nodes  EAL: Multi-process socket /var/run/dpdk/rte/mp\_socket  EAL: Selected IOVA mode 'PA'  EAL: Probing VFIO support...  EAL: VFIO support initialized  EAL: Probe PCI driver: net\_ixgbe (8086:154d) device: 0000:3b:00.0 (socket 0)  EAL: Probe PCI driver: net\_ixgbe (8086:154d) device: 0000:3b:00.1 (socket 0)  EAL: No legacy callbacks, legacy socket not created  hello from core 11  hello from core 12  hello from core 13  hello from core 14  hello from core 15  hello from core 16  hello from core 17  hello from core 18  hello from core 19  hello from core 20  hello from core 21  hello from core 22  hello from core 23  hello from core 24  hello from core 25  hello from core 26  hello from core 27  hello from core 28  hello from core 29  hello from core 30  hello from core 31  hello from core 32  hello from core 33  hello from core 34  hello from core 35  hello from core 36  hello from core 37  hello from core 38  hello from core 39  hello from core 0 |

再次查看大页内存使用情况

|  |
| --- |
| # cat /proc/meminfo | grep HugePage -i  HugePages\_Total: 20  HugePages\_Free: 19  HugePages\_Rsvd: 0  HugePages\_Surp: 0  Hugepagesize: 1048576 kB |

发现有一页内存被占用，此时该如何回收被占用的内存呢，通过在网上各种查，才疏学浅，没找到有效的方法，但仔细看了一下DPDK源代码，其中一个参数“--huge-unlink”参数吸引眼球，对于这个参数的解释为

|  |
| --- |
| Unlink hugepage files after init |

此参数在初始化大页内存后会主动删除，这样，可以达到释放的效果。

我们传入这个参数，再次运行示例程序，并查看大页内存使用情况

|  |
| --- |
| # ./build/helloworld --huge-unlink  EAL: Detected 40 lcore(s)  EAL: Detected 2 NUMA nodes  EAL: Multi-process socket /var/run/dpdk/rte/mp\_socket  EAL: Selected IOVA mode 'PA'  EAL: Probing VFIO support...  EAL: VFIO support initialized  EAL: Probe PCI driver: net\_ixgbe (8086:154d) device: 0000:3b:00.0 (socket 0)  EAL: Probe PCI driver: net\_ixgbe (8086:154d) device: 0000:3b:00.1 (socket 0)  EAL: No legacy callbacks, legacy socket not created  hello from core 11  hello from core 12  hello from core 13  hello from core 14  hello from core 15  hello from core 16  hello from core 17  hello from core 18  hello from core 19  hello from core 20  hello from core 21  hello from core 22  hello from core 23  hello from core 24  hello from core 25  hello from core 26  hello from core 27  hello from core 28  hello from core 29  hello from core 30  hello from core 31  hello from core 32  hello from core 33  hello from core 34  hello from core 35  hello from core 36  hello from core 37  hello from core 38  hello from core 39  hello from core 0  # cat /proc/meminfo | grep HugePage -i  HugePages\_Total: 20  HugePages\_Free: 20  HugePages\_Rsvd: 0  HugePages\_Surp: 0  Hugepagesize: 1048576 kB |

可见内存被成功释放。

## 三层转发l3fwd