# DPDK Getting Started Guide for Linux

本章描述了如何安装及配置DPDK，主要用于DPDK编译运行的快速上手，对于细节并不深入。

# 文档路线图

以下是一份DPDK的文档路线图，学习DPDK可以参照如下文档进行选读。

* Release Note：提供了各个特定发布版本的信息，包括支持特性、限制特定、修订问题、已知问题等等，当然也提供了一些常见问题的解答。
* Getting Started Guide：描述如何安装配置DPDK，用于帮助初学者快速上手
* Programmer Guide：
  + 软件架构及如何使用（使用实例），特别是在Linux环境
  + DPDK的各个组成部分，系统搭建及实例运行
  + 软件的最优实现及开发需要考虑的问题
* API Reference：提供了DPDK接口、数据结构及其他编程结构的详细信息
* Sample Applications User Guide：描述了一组app实例。每章都通过一个实例来展示DPDK的功能及如何编译运行。

# 系统需求

本节描述了编译DPDK的一些需求。

## BIOS Setting Prerequisite on x86

对于大部分的平台，运行DPDK的基础功能，BIOS并不需要做特殊的设置。然而，为了运行额外的HPET（高精度定时器）及电源管理功能，抑或是为了在40G网卡上获得小包处理的高性能，一些BIOS设置修改就是必要的了。这些设置可以参考章节*“开启额外功能”*描述。

## 编译DPDK

### 必选工具

* GNU：make
* coreutils：cmp、sed、grep、arch、etc.
* gcc：
  + 对于i686及x86\_64系统，要求version 4.5.x或者更新的版本
  + 对于ppc\_64或者x86\_32，要求version 4.8.x或者更新的版本
  + 在一些发布版本中，gcc的某些编译选项或链接选项被默认开启了（例如-fstack-protector），这些需要参考你使用的版本的文档或gcc -dumpspecs。
* libc 头文件，常见打包为gcc-multilib（glibc-devel.i686/glibc-devel.i386/glibc-devel.x86\_64/libc-dev for 64 bit compilation on Intel architecture;glibc-devel.ppc64 for 64 bit IBM Power architecture）
* Linux 内核头文件或源码用于编译内核模块
* 在64bit系统中编译32bit时需要额外的安装包：
  + glibc.i686, libgcc.i686, libstdc++.i686, glibc-devel.i686 for Intel i686/x86\_64
  + glibc.ppc64, libgcc.ppc64, libstdc++.ppc64, glibc-devel.ppc64 for IBM ppc\_64

### 可选工具

* Intel C++ Compiler (icc)，为了安装这个，可能需要安装其他的一些库，可以参考ICC安装指南。
* IBM 为PowerLinux提供的增强工具链。这是一组开源的开发工具包和运行时库，用来支持开发者在Linux上运行IBM硬件的额外功能，可以参照IBM官方文档来安装。
* libcap 头文件和库用于编译及使用基于libcap的轮询驱动程序。该驱动默认是不编译的，当然我们也可以通过设置dpdk编译的配置文件选项开启编译CONFIG\_RTE\_LIBRTE\_PMD\_PCAP=y。

## 运行DPDK实例

为了运行DPDK实例，目标机需要一些定制的配置。

### 系统软件

#### 需求

* 内核版本>=2.6.34
  + 内核版本可以通过uname -r命令获取
* glibc>=2.7
  + 版本可以通过ldd –version命令获取
* 内核配置
  + 在Fedora OS或类似的发布版本（如Ubuntu、RedHat）中，系统提供商提供的内核配置可以用于运行大部分的dpdk实例
  + 其他版本系统，dpdk编译运行需要配置如下几点
    - UIO支持
    - HUGETLBFS支持
    - PROC\_PAGE\_MONITOR支持
    - 如果需要HPET，那么HPET和HPET\_MMAP也需要使能

### Use of Hugepages in the Linux Environment

上面的章节讲过Linux内核必须开启HUGETLBFS选项，主要是报文缓存需要用到大量的内存池申请，因此需要HUGEPAGES技术的支持。通过使用大页技术，就可以减少内存页的使用，更少的TLB条目，减少了TLB miss，并且减少了从虚拟内存到物理内存转换的时间消耗，从而提升性能。如果没有大页技术，我们最多只能申请4k大小的页面，较高的TLB miss将导致性能降低，特别是在大量报文处理的设备中。

#### Reserving Hugepages for DPDK use

Hugepages的分配必须在系统boot阶段就完成，或者紧跟在boot之后，防止物理内存被系统分成不连续小段。为了在boot阶段预留hugepages，需要向linux内核传递启动参数，以便启动时能预留出hugepages。

为了得到2MB的页面，只需要传递hugepage选项参数到内核。例如，为了预留1024页2MB大小的页面，需要传递：

hugepages = 1024

其他大小的页，如1G，页面大小需要特殊指定，同样可以通过参数传入到内核中。例如，为了预留4G空间，即1G大小的页面预留4页，需要传递如下参数：

default\_hugepagesz = 1G hugepagesz = 1G hugepages = 4

Note：

Intel架构中，CPU支持的Hugepage页面大小由CPU的flag位决定，如果pse存在，那么2MB大小的页就支持，如果pdpe 1gb存在，那么1G大小的页就支持。在IBM架构中，hugepage大小支持16M和16G两种。

对于64bit的系统，如果CPU支持的话，推荐使用1GB hugepages

对于双插槽的NUMA架构系统， boot阶段预留的hugepages被平均分配给了两个插槽。

更多的内核选项配置及细节可以阅读Linux内核源码中Kernel-parameter.txt文档。

**Alternative**

对于2M大小的hugepages还可以通过向/sys中的文件nr\_hugepages写入hugepages数据需求。例如，在单节点的系统中，命令如下（申请1024个页）

echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr\_hugepages

在NUMA机器中，页面分配需要分别针对不同的节点

echo 1024 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr\_hugepages

echo 1024 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr\_hugepages

对于1G的page，无法在boot之后再分配。

#### Using Hugepages with the DPDK

一旦hugepages内存被预留了，我们就可以通过如下步骤让这些内存在DPDK中使用：

mkdir /mnt/huge

mount -t hugetlbfs nodev /mnt/huge

这个挂在点也可以在启动文件/etc/fstab中加入如下命令来实现：

nodev /mnt/huge hugetlbfs defaults 0 0

对于1GB大小的页，使用如下命令

nodev /mnt/huge\_1GB hugetlbfs pagesize=1GB 0 0

### Xen Domain0 Support in the Linux Enviroment

to be add

# Compiling the DPDK Target form Source

## Install the DPDK and Browse Sources

首先，解压并进入dpdk源码目录

unzip dpdk-version.zip

cd dpdk-version

可以看到，DPDK大致由如下几个目录组成：

* Lib：DPDK库源码
* Drivers：DPDK 轮询模式驱动源码
* App：DPDK应用源码
* Example：DPDK应用实例源码
* Config、tools、scripts、mk：框架相关的makefile、脚本及配置文件

## Install of DPDK Target Environments

DPDK目标文件的格式可以表示如下：

ARCH-MACHINE-EXECENV-TOOLCHAIN

其中：

* ARCH可以是：i686 x86\_64 ppc\_64
* MACHINE可以是：native、ivshmem、power8
* EXECENV可以是：linuxapp、bsdapp
* TOOLCHAIN可以是：gcc、icc

目标文件的编译安装取决于主机系统上的安装包和编译器。DPDK支持的目标文件可以在DPDK/config目录下查询到，前缀defconfig\_标识的无法使用。

当使用ICC，当编译64/32bit时，以下命令必须分别被引用

source /opt/intel/bin/iccvars.sh intel64

source /opt/intel/bin/iccvars.sh ia32

shell脚本跟新了PATH变量。

编译安装目标文件可以通过make命令来实现，在dpdk源码顶级目录使用make install T=<target> 来编译源码。

例如，使用icc编译64bit目标文件，可以使用命令：

make install T=x86\_64-native-linuxapp-icc

如果要使用gcc编译出32bit的目标文件，编译命令为：

make install T=i686-native-linuxapp-gcc

有时候我们可能只要修改配置，但是并不编译，例如配置改变了，我们需要在编译前先修订配置，那么就可以是用如下命令实现：

make config T=x86\_64-native-linuxapp-gcc

所有在使用的内核模块如igb\_uio.ko和kni.ko模块，需要和目标机上跑的模块具有相同的内核配置。如果DPDK不是在目标机上编译出来的，RTE\_KERNELDIR环境变量用于将编译器指向目标机内核版本的一个拷贝

一旦目标环境确定，开发者可以目标环境目录完成剩余的编译和重编译任务。开发者也可以编译DPDK配置的一些改变。

而且，make clean命令可以用于清除原来的编译结果，以便重新编译。

## Browsing the Installed DPDK Environment Target

## Loading Modules to Enable Userspace IO for DPDK

为了运行任何DPDK实例，uio适配模块必须被加载到内核中。很多情况下，linux内核中携带的标准的uio\_pci\_generic模块可以提供uio能力，这个模块可以通过如下命令加载起来：

sudo modprobe uio\_pci\_generic

作为uio\_pci\_generic模块的替代品，DPDK实现了igp\_uio模块，可以通过如下命令加载：

sudo modprobe uio

sudo insmod kmod/igb\_uio.ko

对于部分不支持legacy interrupt的设备，例如 virtual function devices，就需要igb\_uio模块来代替uio\_pci\_generic模块

自从DPDK 1.7版本提供VFIO支持之后，对于是否支持VFIO，UIO可以通过平台配置文件配置。

## Loading VFIO Module

为了运行dpdk实例，且启用VFIO，必须加载vfio-pci模块：

sodu modproce vfio-pci

显然，为了使用VFIO，我们的内核必须支持该功能。VFIO内核模块在Linux kernel version 3.6.0及更新的版本中已经默认提供支持了。

为了使用VFIO，内核和BIOS都必须配置user IO virtualization（例如Intel的VT-d）

当运行DPDK时，为了在普通用户模式下正确操作VFIO，必须时能正确的操作权限，这个可以通过DPDK setup脚本实现。（setup.sh脚本在/dpdk/tool目录下）

## Binding and Unbinding Network Ports to/from the Kernel Modules

从1.4版本之后，DPDK就不再支持将所有的网络接口从原来的内核驱动中解除绑定。相反的，DPDK需要使用的所有端口必须绑定到uio\_pci\_generic、igb\_uio或者vfio-pci中的某个模块中。那些Linux控制下的接口将被DPDK 轮询驱动过滤掉。

默认情况下，dpdk启动时并不会将网卡从Linux内核驱动中解绑定，dpdk app需要用到接口必须从linux内核驱动中解除绑定，然后绑定到uio\_pci\_generic、igb\_uio或是vfio-pci模块中，才能被dpdk app扫描使用

为了绑定网卡打牌uio\_pci\_generic、igb\_uio或vfio-pci模块给dpdk使用，脚本dpdk\_nic\_bind.py被提供来做这个事情。这个脚本可以用于显示当前系统中所有网卡的状态，可以将网卡从不同的内核驱动中绑定或解绑定，包括uio和vfio模块。一下是一些实例用于展示如何使用这个脚本。该脚本完全的解释可以通过—help或者—usage命令呈现。这里我们要注意到，uio或vfio内核模块必须在脚本运行前加载到内核模块中去。

由于VFIO的工作方式，设备能够工作在VFIO下是有一些明确限制的。大部分可以归结为IOMMU group是如何工作的。所有的Virtual Function devices可以使用VFIO，但是物理设备就要求全部绑定到VFIO或者部分绑定到VFIO时其他的不能绑定到任何驱动上。

如果你的设备是挂在PCI-to-PCI bridge下，那么设备所在的bridge将作为IOMMU的一部分。因此，bridge驱动也必须从原来的驱动中解绑定，以便bridge下的VFIO正常工作。

所有用户都有可能操作dpdk\_nic\_bind.py来查看网卡的状态，但是绑定和解绑定需要有root权限

查看网卡的状态

root@suger:/home/suger/tmp/dpdk# ./tools/dpdk\_nic\_bind.py --status

Network devices using DPDK-compatible driver

============================================

<none>

Network devices using kernel driver

===================================

0000:02:01.0 '82545EM Gigabit Ethernet Controller (Copper)' if=eth0 drv=e1000 unused= \*Active\*

0000:02:06.0 '82545EM Gigabit Ethernet Controller (Copper)' if=eth1 drv=e1000 unused= \*Active\*

0000:02:07.0 '82545EM Gigabit Ethernet Controller (Copper)' if=eth2 drv=e1000 unused= \*Active\*

Other network devices

=====================

0000:02:08.0 '82545EM Gigabit Ethernet Controller (Copper)' unused=e1000

root@suger:/home/suger/tmp/dpdk#

将eth1（02:06.0）绑定到uio\_pci\_generic驱动上

/tools/dkdp\_nic\_bind.py –bind=uio\_pci\_generic 02:06.0

或是使用如下命令

/tools/dkdp\_nic\_bind.py –bind=uio\_pci\_generic eth1

# Compiling and Running Sample Applications

待添加

# 调试

## 调试环境

调试环境简要描述：

系统：Linux suger 3.13.0-32-generic #57~precise1-Ubuntu SMP Tue Jul 15 03:51:20 UTC 2014 x86\_64 x86\_64 x86\_64 GNU/Linux

DPDK: 16.04

网卡：

## 快速开始

tar -xvf dpdk-16.04.tar.xz dpdk

cd dpdk

make config T=x86\_64-native-linuxapp-gcc

sed -ri 's,(PMD\_PCAP=).\*,\1y,' build/.config

make

make的时候出现问题：

1. build /driver/pcap时提示pcap.h找不到

因为开了PMD\_PCAP=y，所以会用到pcap库，需要sudu apt-get install libpcap-devel

最好安装一下如下两个软件

sudo apt-get install flex

sudo apt-get install bison

最终编译出的结果在dpdk/build目录中

由于DPDK使用Hugepage技术，因此必须先配置hugepage才能运行

mkdir -p /mnt/huge

mount -t hugetlbfs nodev /mnt/huge

echo 64 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr\_hugepages

配置成功之后，可以通过proc/meminfo查看

root@suger:/home/suger# cat /proc/meminfo

……

AnonHugePages: 26624 kB

HugePages\_Total: 64

HugePages\_Free: 0

HugePages\_Rsvd: 0

HugePages\_Surp: 0

Hugepagesize: 2048 kB

……