Linux DPDK 开发指南

文件标识：RK-YH-YF-987

发布版本：V1.0.0

日期：2023-03-25

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

Rockchip Linux平台DPDK开发指南

产品版本

芯片名称	内核版本
RK3588/RK3568/RK3566	Linux 4.19/5.10

读者对象

本文档（本指南）主要适用于以下工程师:

技术支持工程师

软件开发工程师

硬件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	xy	2022-12	初始版本
V0.2.0	xy	2023-01	增加I320/350 PCIe网卡转发数据
V1.0.0	xy	2023-03	增加GMAC开发指导

Linux DPDK 开发指南

1. DPDK简介

DPDK全称Intel Data Plane Development Kit，是Intel提供的数据平面开发工具集，目前支持为ARM 、X86、PowerPC等处理器架构下用户空间高效的数据包处理提供库函数和驱动的支持，它不同于Linux系统以通用性设计为目的，而是专注于网络应用中数据包的高性能处理，目前已经验证可以运行在大多数Linux操作系统上；DPDK使用了BSD License，极大的方便了企业在其基础上来实现自己的协议栈或者应用，有助于更快地部署高性能网络应用程序，并实现比标准基于中断的内核网络堆栈更高效的运算。

2. DPDK工作原理

DPDK 使用多种技术来优化数据包吞吐量，但其工作方式（及其性能的关键）基于绕过内核和轮询模式驱动程序 （PMD）。

- **Kernel Bypass**：在用户空间内创建从 NIC 到应用程序的路径，换句话说，绕过内核。这消除了在用户空间和内核空间之间移动网络帧时的上下文切换。此外，通过消除 NIC 驱动程序、内核网络堆栈和它们引入的性能损失，可以获得进一步的收益。
- **轮询模式驱动程序 （PMD）**：当 CPU 收到帧时，CPU 不会引发 NIC 的中断，而是不断运行 PMD 以轮询 NIC 以获取新数据包。

除此之外，DPDK 还采用其他方法，例如 **CPU affinity、CPU isolation、大页面内存(huge page memory)、缓存行对齐(cache line alignment)和批量操作(bulk operations)**，以实现可能的最佳性能。

3. 平台支持情况

RK平台支持GMAC/PCIe两种接口的网卡，GMAC最多支持接两个千兆网卡；PCIe 支持x1/x2/x4类型的网卡，RK3588最多支持外接6个网卡：4个PCIe + 2个GMAC，各个芯片详细资源如下：

3.1 RK3566

资源	模式	支持芯片互联	支持lane拆分	支持DMA	支持MMU
PCIe Gen2 x 1	RC only	否	否	否	否
GMAC 1000M (**x2**)	RGMII	/	/	/	是

3.2 RK3568

资源	模式	支持芯片互联	支持lane拆分	支持DMA	支持MMU
PCIe Gen2 x 1 lane	RC only	否	否	否	否
PCIe Gen3 x 2 lane	RC/EP	是	1 lane RC+ 1 lane RC	是	否
GMAC 1000M (**x2**)	RGMII	/	/	/	是

3.3 RK3588

资源	模式	支持lane拆分	可用phy	内部DMA
PCIe Gen3 x 4 lane	RC/EP	是	pcie30phy	是
PCIe Gen2 x 1 lane (**x3**)	RC only	否	pcie30phy, combphy1_ps	否
GMAC 1000M (x2)	RGMII	/	/	是

3.4 RK3588S

资源	模式	支持lane拆分	可用phy	内部DMA
PCIe Gen3 x 1 lane	RC only	否	combphy2_psu	否
PCIe Gen2 x 1 lane	RC only	否	combphy0_ps	否
GMAC 1000M **(x2)**	RGMII	/	/	是

4. DPDK内核配置

注意：

- **GMAC方案**：最新SDK默认支持，旧SDK需要打补丁，参考SDK的external/dpdk/gmac/README.txt，根据内核版本打上对应的补丁。
- **PCIE方案**：SDK的external/dpdk/pcie/README.txt。

4.1 DTS

- GMAC方案首先使能 DTS 中 UIO 节点，以 RK3568-evb1 参考：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
index 0cb57e9d8529..c7729258e51d 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
@@ -262,6 +262,14 @@
        status = "okay";
 };

+&gmac_uio0 {
+       status = "okay";
+};
+
+&gmac_uio1 {
+       status = "okay";
+};
+
 /*
  * power-supply should switche to vcc3v3_lcd1_n
  * when mipi panel is connected to dsi1.
```

4.2 内核配置

defconfig打开如下配置：

```
CONFIG_UIO=m
CONFIG_STMMAC_UIO=m
CONFIG_HUGETLBFS=y
```

4.3 编译KO

```
#不同的平台要根据实际情况修改
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 rockchip_linux_defconfig
#打开4.2章节的配置
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 menuconfig
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 rk3568-evb1-ddr4-v10-linux.img -
j8

#烧写 boot.img及push相关的ko
adb push igb_uio.ko  #参考SDK的external/dpdk/pcie/README.txt
adb push drivers/uio/uio.ko
adb push drivers/net/ethernet/stmicro/stmmac/stmmac_uio.ko
```

4.4 DPDK 编译

DPDK 测试使用的开发板跑的系统是 Debian 11，DPDK 版本 21.11，编译参考 http://doc.dpdk.org/guides-21.11/linux_gsg/cross_build_dpdk_for_arm64.html

- PC 交叉编译

```
wget https://developer.arm.com/-/media/Files/downloads/gnu-a/9.2-2019.12/binrel/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz
tar -xvf gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz

export PATH=$PATH:<cross_install_dir>/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu/bin

apt install build-essential
apt install pkg-config-aarch64-linux-gnu
apt-get install python3 python3-pip
apt install meson
apt install ninja-build

#特别注意，修改下面文件里的编译链的名字要跟下载安装的一致,如果不一样要修改成一致的，否则无法编译!
config/arm/arm64_armv8_linux_gcc

meson aarch64-build-gcc --cross-file config/arm/arm64_armv8_linux_gcc
meson --reconfigure aarch64-build-gcc --cross-file config/arm/arm64_armv8_linux_gcc -Dbuildtype=debug -Dplatform=arm64 -Dexamples=l2fwd,l3fwd
ninja -C aarch64-build-gcc
```

- 开发板编译 DPDK（适用Debian系统）

```
apt-get install python3 python3-pip
apt install meson
apt install ninja-build
pip3 install meson ninja
apt-get install libdpkg-perl
apt-get install build-essential
apt-get install python3-pyelftools
apt install libnuma-dev
apt install libpcap-dev

meson setup -Dplatform=generic build
cd build
ninja
ninja install
```

5. 运行 DPDK 程序

5.1 挂载巨页

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

5.2 加载 KO

```
#加载UIO
insmod uio.ko
insmod stmmac_uio.ko    #GMAC
insmod igb_uio.ko       #PCIe

#加载GMAC PMD网卡驱动
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.0
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.1

#加载PCIE PMD网卡驱动（0000:01:00.X要改成实际的）
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.0
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.1
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.2
dpdk/usertools/dpdk-devbind.py -b igb_uio 0000:01:00.3
... ...

#开启性能模式(个别出错信息请忽略)
echo performance | tee $(find /sys/ -name *governor) /dev/null || true

#启动大页支持
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages

#启动测试，  --txpkts=1500可以设置包的大小
dpdk-testpmd -l 5,6,7,8 -n 4 -- -i --nb-cores=1 --forward-mode=flowgen --
txpkts=1500
```

默认开机起来后的网络走的还是原生内核网络，这里我们通过 insmod stmmac_uio.ko 和 rmmod 该 ko 来做到 DPDK 与内核原生网络的切换，即卸载 KO 后，会回到原来的内核网络状态。

- 进入 DPDK 网络控制：

```
//以GMAC为例:
root@linaro-alip:/home/linaro#insmod stmmac_uio.ko
[   41.232761] Generic PHY stmmac-1:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-1:00, irq=POLL)
[   41.236447] dwmac4: Master AXI performs any burst length
[   41.236497] rk_gmac-dwmac fe010000.ethernet eth0: No Safety Features support
found
[   41.236886] rockchip_eth_uio_drv fe010000.uio: Registered uio_eth0 uio
devices, 3 register maps attached
[   41.241453] Generic PHY stmmac-0:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[   41.257084] dwmac4: Master AXI performs any burst length
[   41.257138] rk_gmac-dwmac fe2a0000.ethernet eth1: No Safety Features support
found
[   41.257523] rockchip_eth_uio_drv fe2a0000.uio: Registered uio_eth1 uio
devices, 3 register maps attached
```

- 返回内核网络控制:

```
//以GMAC为例:
root@linaro-alip:/home/linaro# rmmod stmmac_uio
[ 2200.304636] Generic PHY stmmac-0:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 2200.318163] dwmac4: Master AXI performs any burst length
[ 2200.318205] rk_gmac-dwmac fe2a0000.ethernet eth1: No Safety Features support
found
[ 2200.318227] rk_gmac-dwmac fe2a0000.ethernet eth1: IEEE 1588-2008 Advanced
Timestamp supported
[ 2200.318443] rk_gmac-dwmac fe2a0000.ethernet eth1: registered PTP clock
[ 2200.319414] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[ 2200.322971] Generic PHY stmmac-1:00: attached PHY driver [Generic PHY]
(mii_bus:phy_addr=stmmac-1:00, irq=POLL)
[ 2200.324883] dwmac4: Master AXI performs any burst length
[ 2200.324921] rk_gmac-dwmac fe010000.ethernet eth0: No Safety Features support
found
[ 2200.324945] rk_gmac-dwmac fe010000.ethernet eth0: IEEE 1588-2008 Advanced
Timestamp supported
[ 2200.325468] rk_gmac-dwmac fe010000.ethernet eth0: registered PTP clock
[ 2200.325814] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 2205.385406] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

5.3 设置 performance 模式

```
echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
echo performance > /sys/devices/system/cpu/cpufreq/policy4/scaling_governor
echo performance > /sys/devices/system/cpu/cpufreq/policy6/scaling_governor
echo performance > /sys/class/devfreq/dmc/governor
#或
echo performance | tee $(find /sys/ -name *governor) /dev/null || true
```

5.4 运行 testpmd

- 转发模式测试命令

> --vdev=net_stmmac0 --vdev=net_stmmac1 表示指定的虚拟设备，目前是名字是固定的（**PCIe无需指定**）
>
> --main-lcore=0 表示 0 核用作管理，2 和 3 核用于转发
>
> --iova-mode=pa 因为设备不支持 iommu，故 iova-mode 规定为 pa 模式
>
> -- 用于分隔 eal 参数与 testpmd 的参数
>
> -i 表示进入 dpdk-testpmd 命令交互模式

```
#以GMAC为例:
root@linaro-alip:/home/linaro# ./dpdk-testpmd  --iova-mode=pa --vdev=net_stmmac0
--vdev=net_stmmac1 -l 0,2,3 --main-lcore=0 -- -i
EAL: Detected CPU lcores: 4
EAL: Detected NUMA nodes: 1
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
TELEMETRY: No legacy callbacks, legacy socket not created
Interactive-mode selected
Warning: NUMA should be configured manually by using --port-numa-config and --
ring-numa-config parameters along with --numa.
testpmd: create a new mbuf pool <mb_pool_0>: n=163456, size=2176, socket=0
testpmd: preferred mempool ops selected: ring_mp_mc
Configuring Port 0 (socket 0)
stmmac_net: stmmac_eth_link_update()Port (0) link is Up
Port 0: BA:A0:3F:FD:B2:8C
Configuring Port 1 (socket 0)
stmmac_net: stmmac_eth_link_update()Port (1) link is Up
Port 1: B6:A0:3F:FD:B2:8C
Checking link statuses...
stmmac_net: stmmac_eth_link_update()Port (0) link is Up
stmmac_net: stmmac_eth_link_update()Port (1) link is Up
Done
```

- 开启转发模式测试

```
testpmd> start
io packet forwarding - ports=2 - cores=1 - streams=2 - NUMA support enabled, MP
allocation mode: native
Logical Core 2 (socket 0) forwards packets on 2 streams:
  RX P=0/Q=0 (socket 0) -> TX P=1/Q=0 (socket 0) peer=02:00:00:00:00:01
  RX P=1/Q=0 (socket 0) -> TX P=0/Q=0 (socket 0) peer=02:00:00:00:00:00

  io packet forwarding packets/burst=32
  nb forwarding cores=1 - nb forwarding ports=2
  port 0: RX queue number: 1 Tx queue number: 1
    Rx offloads=0x0 Tx offloads=0x0
    RX queue: 0
      RX desc=0 - RX free threshold=0
      RX threshold registers: pthresh=0 hthresh=0  wthresh=0
      RX Offloads=0x0
    TX queue: 0
```

```
    TX desc=0 - TX free threshold=0
    TX threshold registers: pthresh=0 hthresh=0  wthresh=0
    TX offloads=0x0 - TX RS bit threshold=0
port 1: RX queue number: 1 Tx queue number: 1
  Rx offloads=0x0 Tx offloads=0x0
  RX queue: 0
    RX desc=0 - RX free threshold=0
    RX threshold registers: pthresh=0 hthresh=0  wthresh=0
    RX Offloads=0x0
  TX queue: 0
    TX desc=0 - TX free threshold=0
    TX threshold registers: pthresh=0 hthresh=0  wthresh=0
    TX offloads=0x0 - TX RS bit threshold=0
```

- 查看转发数据:

```
testpmd> show port stats all

######################## NIC statistics for port 0  ########################
RX-packets: 5240160    RX-missed: 0          RX-bytes:  314409600
RX-errors: 0
RX-nombuf:  0
TX-packets: 0          TX-errors: 0          TX-bytes:  0

Throughput (since last show)
Rx-pps:            0          Rx-bps:            0
Tx-pps:            0          Tx-bps:            0
############################################################################

######################## NIC statistics for port 1  ########################
RX-packets: 0          RX-missed: 0          RX-bytes:  0
RX-errors: 0
RX-nombuf:  0
TX-packets: 5180224    TX-errors: 0          TX-bytes:  310813440

Throughput (since last show)
Rx-pps:            0          Rx-bps:            0
Tx-pps:            0          Tx-bps:            0
############################################################################
testpmd> show port stats all

######################## NIC statistics for port 0  ########################
RX-packets: 6382336    RX-missed: 0          RX-bytes:  382940160
RX-errors: 0
RX-nombuf:  0
TX-packets: 0          TX-errors: 0          TX-bytes:  0

Throughput (since last show)
Rx-pps:       681809          Rx-bps:    327268480
Tx-pps:            0          Tx-bps:            0
############################################################################

######################## NIC statistics for port 1  ########################
RX-packets: 0          RX-missed: 0          RX-bytes:  0
RX-errors: 0
RX-nombuf:  0
TX-packets: 6322397    TX-errors: 0          TX-bytes:  379343820
```

```
  Throughput (since last show)
  Rx-pps:            0          Rx-bps:            0
  Tx-pps:       681810          Tx-bps:    327269160
  ############################################################################
```

- 其它常用设置：

  --nb-cores 表示指定 dpdk-testpmd 用作转发工作的核的数量

  --rxd 接收队列描述符

  --txd 发送队列描述符

  --rxq 表示指定 dpdk-testpmd 接收队列数，RK3568 队列数为1

  --txq 表示指定 dpdk-testpmd 发送队列数，RK3568 队列数为1

5.5 运行 l2fwd

l2fwd 默认至少要有两个核才能测试转发

```
./dpdk-l2fwd -l 0,2,3  --main-lcore=0 --iova-mode=pa --vdev=net_stmmac0 --
vdev=net_stmmac1 -- -q 1 -p 0x3
```

l2fwd 运行的串口信息每隔10s钟会刷新一次，考虑可能会导致丢包，建议将串口信息重定向到文件。

注：--vdev=net_stmmac0 --vdev=net_stmmac1 表示指定的虚拟设备，目前是名字是固定的（**PCIe无需指定**）。

5.6 运行 l3fwd

```
./dpdk-l3fwd -l 3 -n 1 --iova-mode=pa --vdev=net_stmmac0 --vdev=net_stmmac1 -- -p
0x3 -P --config="(0,0,3),(1,0,3)" --parse-ptype
```

> -p PortMask 参数指定使用的网口掩码;
>
> --vdev=net_stmmac0 --vdev=net_stmmac1 表示指定的虚拟设备，目前是名字是固定的（**PCIe无需指定**）;
>
> -P 参数表示将所有网口设置为混杂模式，以便收到所有数据包;
>
> --config (port,queue,lcore)，[(port,queue,lcore)] 参数用以配置网口、队列、核之间的对应关系，例如，--config (0,0,3) 表示网口 0 的队列 0 由核 3 进行处理;

值得注意的是，上述输出中打印了 l3fwd 的默认路由规则，即

```
LPM: Adding route 198.18.0.0 / 24 (0) [net_stmmac0]
LPM: Adding route 198.18.1.0 / 24 (1) [net_stmmac1]
LPM: Adding route 2001:200:: / 64 (0) [net_stmmac0]
LPM: Adding route 2001:200:0:1:: / 64 (1) [net_stmmac1]
```

也就是说，目的 IP 为 198.18.0.0/24 段的数据包将会通过网口 0 进行转发，目的 IP 为 198.18.1.0 / 24 段的数据包将会通过网口 1 进行转发。上述默认路由规则是在源码中配置的，所以在l3fwd测试的时候，需要设置好测试数据的目标ip和源ip。

6. Pktgen

在板子上跑Pktgen，是基于 DPDK，所以需要先编译和安装好 DPDK，编译参考1.2章节的开发板编译DPDK。

6.1 下载 pktgen-dpdk 源码

```
git clone http://dpdk.org/git/apps/pktgen-dpdk
apt-get install libpcap-dev
apt-get install libnuma-dev
apt install meson
apt install ninja-build pkg-config
```

6.2 DPDK 编译

```
cd build
ninja
ninja install
ldconfig
export PKG_CONFIG_PATH=/usr/local/lib/aarch64-linux-gnu/pkgconfig/
```

6.3 Pktgen 编译

```
cd pktgen-dpdk
meson build
cd build
ninja
```

6.4 运行 Pktgen 程序

```
./build/app/pktgen --iova-mode=pa --vdev=net_stmmac0 -l 6,7 --proc-type auto --
log-level debug -- -P -m 7.0
```

其中，EAL options 参数部分可以参看 DPDK EAL parameters，最重要的一个参数就是 -l 参数，用它来指定使用的核列表，比如：-l 1,2 或者 -l 1-2，表示使用核 1 和核 2。

值得注意的是，pktgen 至少要指定两个核，因为 pktgen 需要一个核与用户进行交互，比如响应测试过程中用户的输入。
pktgen 自有参数部分最重要的是 -m 参数，用它来指定网口与核之间的对应关系，比如：
-m 2.0：表示让核 2 来处理网口 0。值得注意的是，若要指定多个对应关系（使用多个网卡和多个核），则需多次使用 -m 参数。
如果要收包，最好也指定一下 -P 参数，表示让所有网口进入混杂模式，以便接收到所有数据包。

设置数据包格式并开启 Pktgen：

```
set 0 size 64
set 0 src ip 198.18.0.100/24
set 0 dst ip 198.18.1.101
set 0 dst mac ba:a0:3f:fd:b2:8c
start 0
```

7. 性能指标

环境 (单核)	L2转发	L3转发
RK3588 GMAC + 64 Byte	650Mbit/s	530Mbit/s
RK3588 GMAC + 1500 Byte	940Mbit/s	940Mbit/s
RK3588 Intel I210/350 PCIe + 64 Byte	N/A	737Mbit/s
RK3588 Intel I210/350 PCIe + 1500 Byte	940Mbit/s	940Mbit/s
RK3568 GMAC + 64 Byte	600Mbit/s	580Mbit/s
RK3568 GMAC + 1500 Byte	940Mbit/s	940Mbit/s

注意:

1. 对于多核，可以按照85%左右的效率进行转换，比如单核1000Mbps，则8核为1000 x 8 x 0.85= 6.8Gbit/s；
2. N/A表示暂未实际测试数据后续补充，但L2一般都大于L3，它少了一个查找路由表的操作；

8. 网卡支持列表

模块	接口	速率
GMAC+PHY	GMII	1000M
Intel I210	PCIe x1	1000M
Intel I350	PCIe x4	1000M x4
RTL8111H	PCIe x1	1000M