**DPDK上下行速率加速现存问题总结**

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 |  |  | 2020.08.25 |
|  |  |  |  |  |  |  |

# 简介

5G基站软件架构如下图，其中用户面CUUP与核心网之间，以及CUUP和DU之间均需要进行加速，当前协议组测得速率大概为900 Mbps的环境为：采用linux内核协议栈、UDP。



Figure 1 5G基站软件架构

## 缩略词

DPDK： Date Plane Development Kit，英特尔开源

VPP： Vector Packet Processing，思科开源，支持DPDK

F-Stack：FreeBSD+DPDK，腾讯开源

# 上下行速率提升

## DPDK介绍

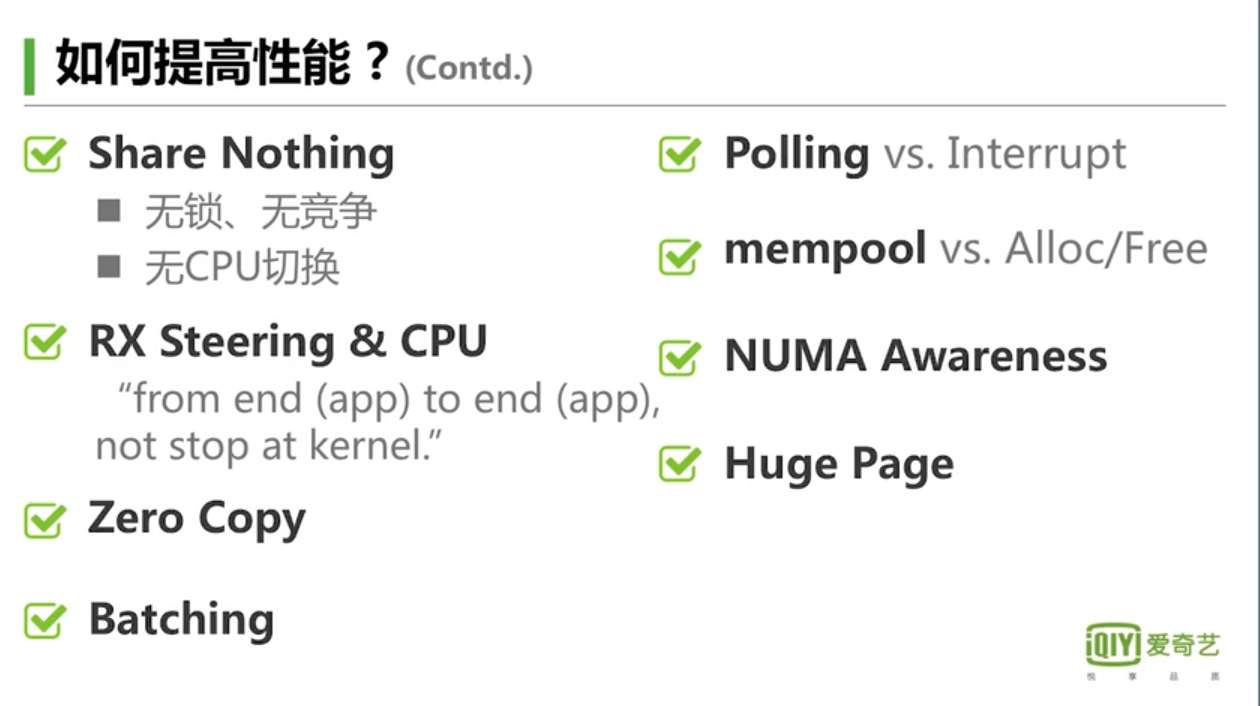


Figure 2 如何提高性能？

DPDK是一种kernel Bypass方案，略过内核协议栈，打破了中断、线程调度、内存拷贝、资源竞争等多种限制。

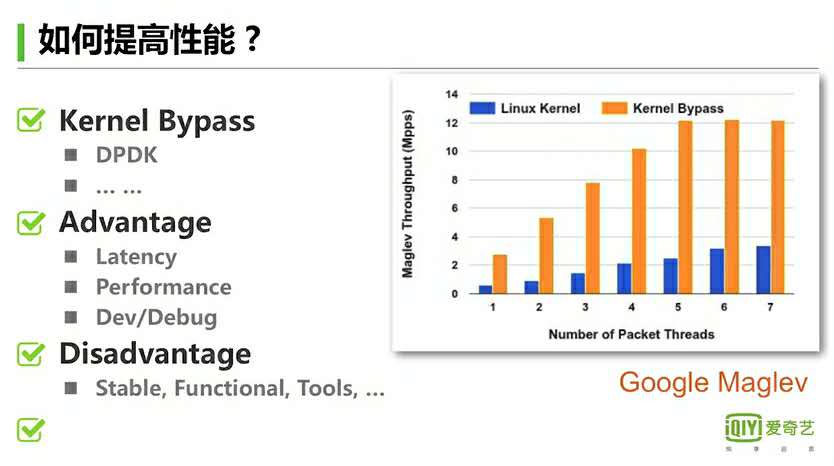
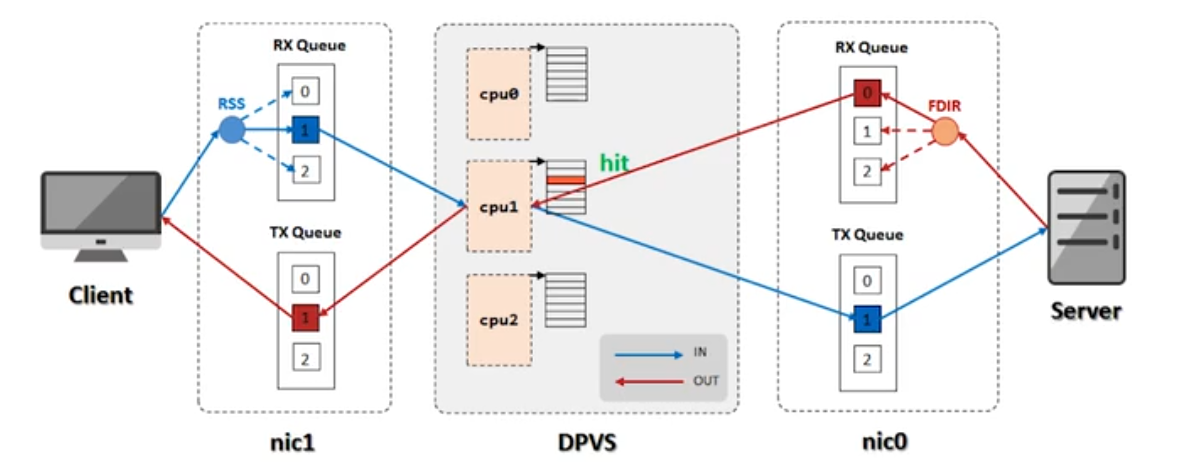


Figure 3 Kernel Bypass



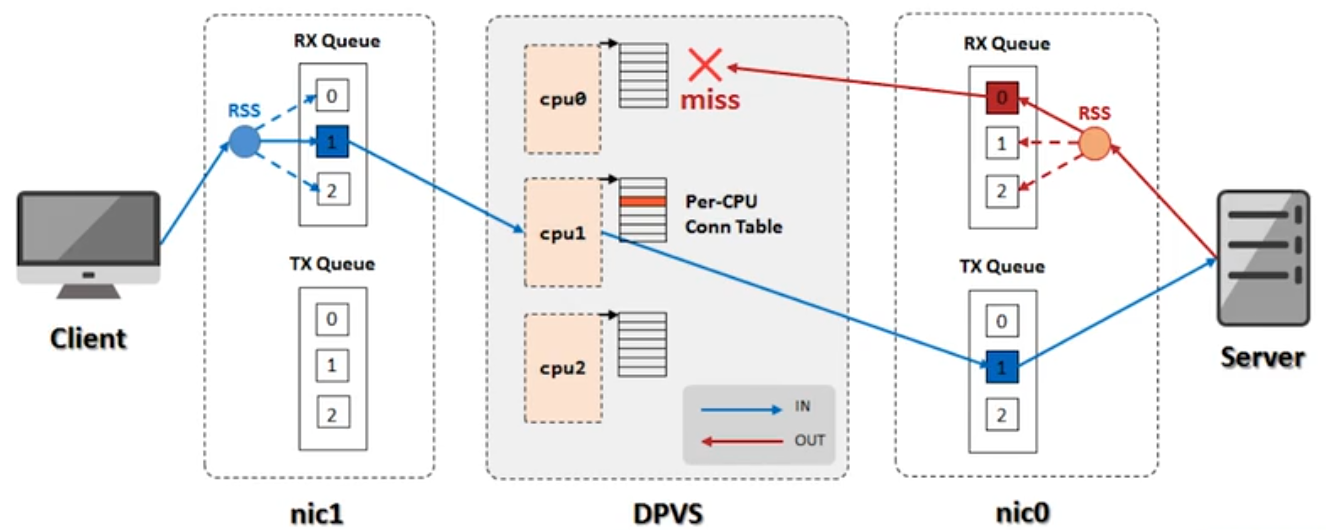


Figure 4 Package转发

DPDK的lcore可认为是一个pthread\_create线程，这个线程与cpu core绑定，并独占网卡，进行RX、TX队列的轮询。整体理解，DPDK接管的网卡作为加速通道，绑核的线程轮询环形队列进行收发包。

## F-Stack问题

下图为部署F-Stack的整体架构，F-Stack通过DPDK绑定10G intel网卡，f-stack 应用程序中提供异步的回调函数进行数据的异步收发。



Figure 5 F-Stack协议栈接管10G网口架构

下图为5G产品的现有软件架构，基本上为单进程内多线程，其中涉及到多线程的UDP收发，这在使用f-stack时遇到了线程间交互的问题（下图紫色线标注）。



Figure 6 用户线程发包

因为f-stack的异步接口中是异步监听fd的，所以初始涉及想采用pipe管道进行异步发包的实现，收包采用回调函数。



Figure 7 利用管道实现多线程异步fstack发包

实现上图架构的F-Stack程序，通过实际的速率测试结果来看，速率并不能满足要求，甚至远低于采用内核协议栈的收发速率，具体原因分析：

1. 多线程应用导致线程调度开销巨大；
2. 多线程间异步发包通道限制（基于fd的管道read、write速率限制）；

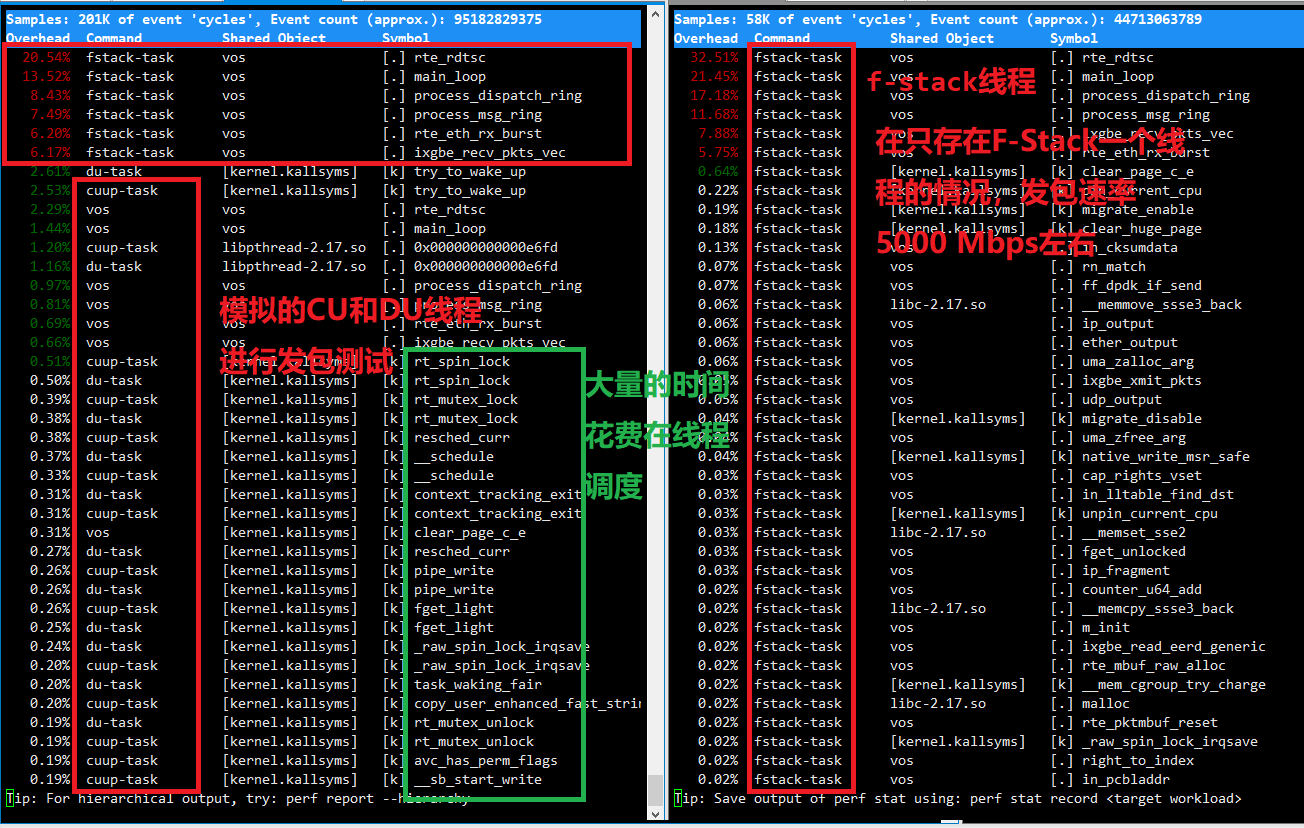


Figure 8 F-Stack多线程面临的线程调度开销

接下来，打算实现上层（协议栈>L4层）的类似于异步rte\_ring机制的轮询接口，通过rte\_mempool的内存支持。这里需要讨论



Figure 9 实现一套基于rte\_ring + mempool的异步机制

关于基于F-Stack的多线程应用的开发，腾讯官方“F-Stack交流群”微信群中“chris”给出了如下评论，值得在此处作为技术性参考，并作为上图方案的可行性依据，进行探讨。

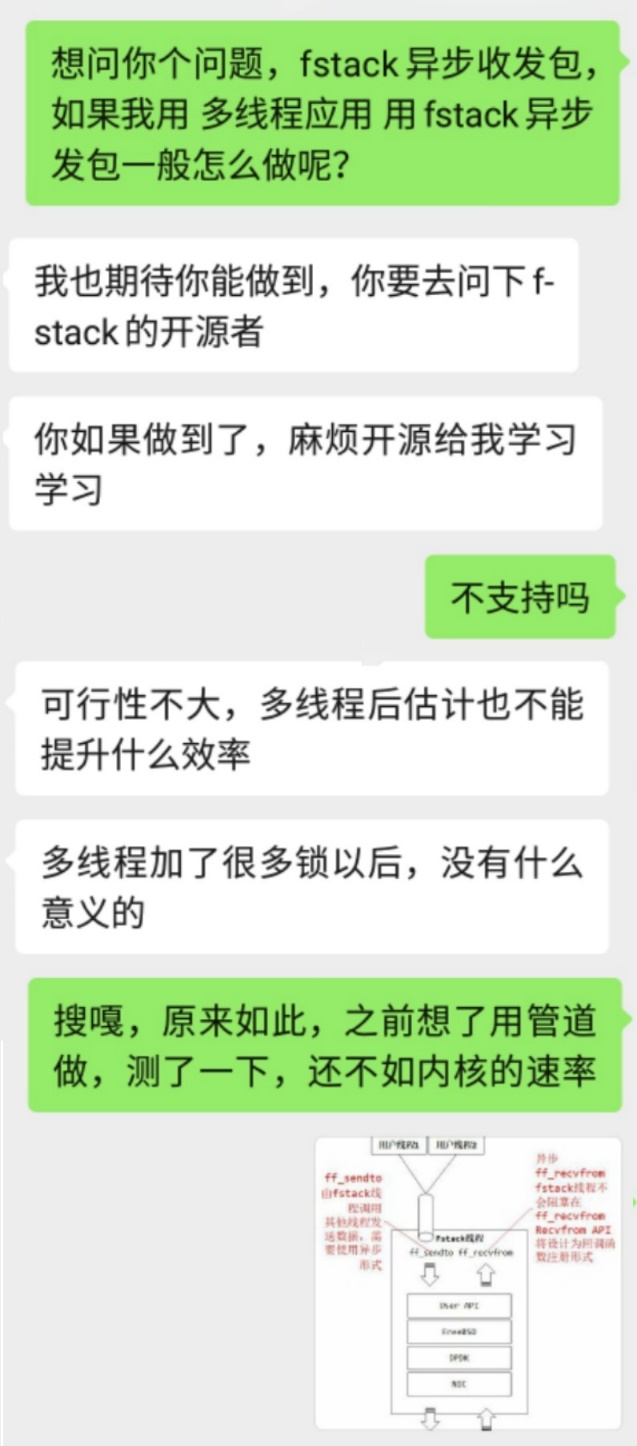


Figure 10 资深人员对多线程F-Stack实现的建议

## VPP问题

目前正在调研中，安装过程遇到依赖问题。

其官方给出的应用场景中，VPP分为数据面和控制面，其中控制面采用netconf\yang模型，为当前我们5G产品与RRU的交互方式。

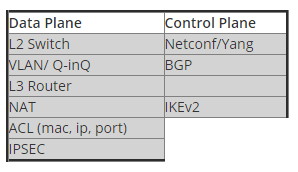


Figure 11 Figure 8 VPP应用场景：通用CPE（<https://fd.io/overview/fdiousecases/>）

## \*\*其他解决方案

### Linux内核调优

重新编译内核，调研内核配置参数“CONFIG\_XXX”是否由关于协议栈线程调度、中断处理、内存拷贝相关配置，减小线程调度、中断处理、内存拷贝开销，提高性能。

### 应用程序优化



Figure 12 应用程序的优化：消息队列

#### 内存开销

1. 减少内存拷贝、赋值；
2. 采用内存池，较小malloc、free开销；

#### 线程调度

1. 线程绑核与独占；
2. 减小线程间通信；
3. 线程间不要共享内存池（锁）；
4. SCHED\_XXX：SCHED\_FIFO，SCHED\_RR选项；

#### 资源竞争

1. 尽量避免使用锁，能用读写锁就别用互斥锁；
2. 减小临界区大小，合理使用CAS；

### Kernel Bypass

1. 采用基于DPDK的用户态协议栈；

### \*\*光口-光口

由于北京服务器环境的限制，不满足“光口-光口”的测试，之前在南京出差阶段，在南京搭建了“光口-光口”的测试环境，采用iperf进行UDP灌包测试，基本上可以满足“4个用户每个用户1.5 Gbps共6 Gbps的速率”的要求。这里需要进行系统化的测试。

提示：

1. iperf和iperf3测得的速率不一样；
2. iperf测的速率如下图：平均8.3 Gbps
3. iperf3测得的速率大概为iperf的一半：平均4.5 Gbps（网卡已开启双工）



Figure 13 光口-光口 iperf速率测试