## 2简介

Dpdk为数据包处理程序提供了一个简单、完整和高效的底层框架。

Dpdk在生成一个eal库，该库与平台、编译器、系统体系架构（32/64位）等因素相关。用户可以将eal库链接进来，生成自己的应用。除了eal库，还有一些支持特定功能的库，如hash、Longest Prefix Match、rings等库，也都可以在应用中使用。

Dpdk的收发包使用的是轮询模式，而不是中断模式（原因是中断处理过程开销比较大）；当dpdk的进程运行时，dpdk进程会一直占用cpu直到运行结束（run-to-completion model）。

Dpdk除了支持run-to-completion model外，还支持多核之间pipeline的模式（通过数据包和消息的核间通信），以充分提高cpu core的计算效率。

### 2.1开发环境

Dpdk的安装环境：

Linux、编译工具、make utility、各种基础库

创建dpdk应用程序时，需要指定两个环境变量:

RTE\_SDK 指定dpdk头文件和库的存放路径

RTE\_TARGET指定dpdk库的编译环境、操作系统体系架构、编译器和内存体系等。

### 2.2环境抽象层（environment abstraction layer）

环境抽象层提供了一些列的基础组建的接口供上层应用程序调用：

1. dpdk的加载和启动
2. 支持多线程和多进程
3. Cpu亲和性和cpu核绑定
4. 内存的分配和回收
5. 原子操作
6. 时间相关操作
7. Pci总线访问
8. 跟踪和调试
9. Cpu特征识别（cpu feature identification）

10、中断处理

11、alarm operations

详见文档Environment Abstraction Layer。

### 2.3重要组件

核心组件的层次图：



#### 2.3.1 内存管理(librte\_malloc)

Librte\_malloc提供内存分配的API，内存分配的API从hugepages中分配内存。从hugepages中分配内存的好处是在分配大片内存的时候避免频繁地出现TLB miss。

#### 2.3.2 ring Manager(librte\_ring)

Ring数据结构提供一套支持多生产者多消费者的无锁队列，能够高效地支持批量操作。Ring作为基础组件为librte\_mempool库使用，同时，也可以作为支持多核间的通信以及单核上不同模块之间通信的机制。

#### 2.3.3 内存池管理(librte\_mempool)

Librte\_mempool用于内存对象池的分配和管理，对象池的名字可以作为对象池的标识，对象池使用ring来存放空闲的对象。Librte\_mempool还提供了一些其他的特性，比如说per-core object cache和内存补齐，保证对象在内存中是对齐存放，方便对象分散存放到多个内存通道中（RAM channels内存通道的作用是什么？）

#### 2.3.4 网络数据包缓存管理(librte\_mbuf)

Librte\_mbuf提供了机制创建和销毁buffers，给dpdk应用程序存放message buffers，message buffers在程序启动的时候创建，存放在mempool中。

Librte\_mbuf提供了API来分配/释放 mbuf，操作control message buffers(ctrlmbuf)和packet buffers(pktmbuf)，其中ctrlmbuf用于存放一般消息数据，pktmbuf用于存放网络数据包。

#### 2.3.5 时间管理（librte\_timer）

Librte\_timer库提供程序异步执行的机制，EAL提供了一套精确的计时、定时机制，用来支持函数周期性的调用或者瞬发调用。

### 2.4 网卡轮询模式

Dpdk提供轮询的机制来实现1GE，10GE，40GE以及虚拟网卡的收发报文，不提供中断方式来做报文收发。

### 2.5包转发算法支持

Dpdk提供了librte\_hash库和librte\_lpm(Longest Prefix Match)库用来支持相关的包转发算法的实现。

### 2.6 librte\_net

Librte\_net提供了一系列的协议定义和宏定义，用来支持上层的解码。