DPDK

背景

1.传统的收发报文方式都必须采用硬中断来唤醒进程，每次硬中断大约消耗100微秒，这还不算因为修正上下文所带来的Cache Miss。

2.数据必须从内核态内存空间拷贝到用户态内存空间，在用户态进行数据处理

3.收发包都有系统调用的开销。

- 轮询模式驱动DPDK
  - 概念：PMD也指用户空间的特定的驱动程序接口的API组成，用于对设备和它关联的队列进行设置。
  - 特点：去除中断，避免内核态和用户态内存拷贝，减少系统开销，从而提升I/O吞吐能力
- Ring Queue
  - 针对单个或多个数据包生产者，单个数据包消费者的出入队列提供无锁机制，有效减少系统开销
- HUGEPAGE
  - 如果有多个核可能需要同时访问一个网卡，把DPDK中分为每个核准备一个单独的接收队列/发送队列，这样避免了竞争，也避免了cache一致性问题。
  - 利用内存大页HUGEPAGE降低TLB(Translation Lookaside Buffer) miss，利用内存多通道交错访问提高内存访问有效带宽,降低访问开销

4.内核工作在多核上，为何全局一数，即使采用Lock Free，也避免不了锁总线，内存屏障带来的性能损耗。

概念

数据平面开发套件(DPDK [1] ,Data Plane Development Kit)是由Intel(英特尔)公司开发，主要基于Linux系统运行，用于快速数据包处理的函数库与驱动集合，可以收入提高数据处理性能和吞吐量，提高数据平面应用程序的工作效率。

主要目的

在数据平面应用中为快速的数据包处理提供一个简单而完善的架构。在理解此工具集之后，开发人员可以以此为基础进行新的原型设计，或用单纯为己用。

工作环境

DPDK的环境抽象层向应用与函数库隐藏了底层环境的细节，因而能扩展到任何处理器上使用

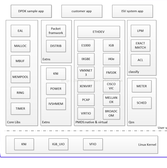
- 操作系统
  - linux
  - FreeBSD

DPDK的高性能代码实现

- 轮询模式
- 内存管理
  - 多级内存分配
  - 内存对齐
  - 内存通道
  - 内存池
- 多核编程
  - 多核亲和性
  - 多核调度
  - 多核同步
  - 无锁数据结构
- 硬件卸载
  - 网卡流量分类
  - 校验和计算
  - 加密解密
- 性能优化
  - 缓存优化
  - 指令级并行
  - 内存预取
- 网络协议栈
  - 轻量级协议栈
  - 自定义协议支持
- 虚拟化支持
  - 虚拟设备驱动
  - 与虚拟机通信
- 调试与监控
  - 性能统计
  - 错误检测
- 安全性
  - 内存保护
  - 数据加密
- 可扩展性
  - 插件架构
  - 模块化设计
- 兼容性
  - 跨平台支持
  - 硬件抽象层
- 文档与社区
  - 详细文档
  - 活跃社区支持

框架

- 核心库Core Libs
  - 环境抽象层(EAL, Environment Abstraction Layer)
    - 功能：主要负责对计算机底层资源（如硬件和内存空间）的访问，并对提供给用户的接口实施了实现细节的封装。其初期的侧重点是了如何分配这些资源（PCI设备，计时器，控制台等
    - EAL提供的典型服务
      - 加载和初始化DPDK：DPDK及其应用程序会被链接为单一应用，因此需要通过某种方式进行加载DPDK。
      - CPU功能识别：在运行时确定CPU是否支持特定功能。确定当前CPU是否支持编译产生的二进制指令集。
      - 核关联/分配过程：EAL提供了将执行单元分配给特定核并创建执行实例的机制。
      - 预留系统内存：EAL为应用不同的内存区域提供便利，例如用于设置交互的物理内存区域。
      - 抽象PCI地址：EAL提供了访问PCI地址空间的接口。
      - 中断处理：内核空中断派发到用户态的接口。
      - 跟踪和调试功能：日志、堆栈转储等。
      - 实现的功能：libc中所没有提供的自定义时间原子计数器。
      - 时钟功能：用于设置/管理在特定时间运行的回调函数接口。
  - data manager
    - Malloc Library：提供分配任意大小内存的API。
    - Ring Library：提供对一种特殊的缓冲队列的管理，具有无进先出、定长、无锁、并发入/出队等特性。
      - 优点：工作更快、实现简单、适用于块数据的入/出队。
    - Mempool Library：内存池用于分配固定大小的对象。在DPDK中内存池用户空区分，并用环形存储来使用的对象
    - Mbuf Library：用于分配网络数据包处理的数据结构。使用上述的内存池来存储。
- extis(Packet manager)
  - Packet Framework
    - 1.为建造复杂的数据包处理管道提供标准方法
    - 2.为所即的流量操作块提供在抽象软件和硬件加速达两种实现间切换的能力
    - 3.在流水线和多核间转移数据分析中方案
    - 4.提供逻辑上类似于Open Flow（SDN核心技术）的架构。
  - Packet Distributor Library
    - 用于流量的动态负载均衡问题。使用时，使用中的逻辑核心会被考虑为两种角色：其一是一个分配器核心，负责负载均衡或数据包分配；其二是一群工人核心，负责从分配器接收数据包并对它们进行处理。
    - Mbufs in → Distributor core → Worker 1 … Worker 8
    - Figure 3.1 : Packet Distribution mode in operation
    - 缺点：定长、环多导致内存占用增加
- PMD Library：提供全用户态驱动，以便通过轮询/无线程模式管理网络端口，支持各种本地和虚拟网卡。
- Classify Library：支持精确匹配、最长匹配和通配符匹配。提供常用包处理的查表操作。
- Qos Library：提供网络服务质量相关组件，限速和调度。
- KNI(Kernel NIC Interface)：为了让数据包进入内核协议栈，通过创建虚拟设备用于收发报文以达到利用内核本身已实现协议的目的
- UIO(user-space drivers)
  - 作用：作为高并发大流量网络开发框架的DPDK，能够避免内核中断频率和大量数据拷贝动力法，在用户空间能够直接和硬件进行交互。
  - 原理：将硬件端口映射到用户空间的kernel bypass方案
  - 实现
    - 1.开发运行在内核的UIO模块，以为硬件端口进行内核处理
    - 2.read阻塞中断，通过/dev/uio读取中断
    - 3.通过mmap和内核共享内存
  - Kernelspace：internal kernel functions (non-stable API) → UIO Framework → Driver
  - Interface：sysfs → /dev/uioX
  - Userspace：Application → Driver → read() / mmap()
  - 驱动的作用
    - 内核态部署
      - 处理硬件产生的中断以及存取硬件的内存区域数据。
      - 分配和注册设备需要的资源和注册uio设备
      - 使能设备
      - 申请资源
      - 读取并记录配置信息
      - 注册uio设备
      - 必须"在内核空间实现的小部分中断应答函数
    - 用户态部署
      - 获取中断事件(read/poll)
      - 处理中断(读写数据)
    - igb_uio分析
      - 注册一个pci设备，在DPDK工具/dpdk_nic_bind.py绑定NIC的时候这个驱动会probe到这个设备，进行相关配置，之后会注册一个UIO设备，probe函数会将公共设备的资源比如PCI设备BAR空间的物理地址、大小等信息记录下来传给用户态。



DPDK架构

- 优点
  - 性能高
  - 用户态开发一目程序就清，不受子导致内核完蛋，带来更高的健壮性
- 缺点
  - 无网络协议栈
  - 开发困难，周期长

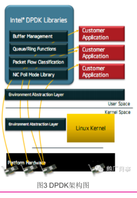


图3 DPDK架构图