## 3、环境抽象层environment abstraction layer

Environment abstraction layer用于访问底层的资源，如硬件、内存空间等。环境抽象层提供统一的操作接口，屏蔽了底层环境相关的实现细节，并在初始化例程中决定了如何分配底层的资源（如内存空间、pci设备、定时器等）

EAL提供的组件包括：

1. dpdk加载和运行：加载dpdk运行环境
2. cpu亲和性和cpu核绑定：EAL提供一套将执行单元分配到特定核上运行的机制
3. 系统内存保留：EAL设置保留各种内存区域，例如用户设备交互的物理内存空间
4. PCI地址访问：EAL提供访问PCI地址空间的接口
5. 跟踪和调试功能：提供logs，dump\_stack，panic等等机制
6. 公用基础函数：自旋锁、读写锁、原子操作等
7. Cpu特性：决定运行时cpu的某些特性是否支持（如intel AVX指令集等）。判断当前正在编译的二进制使用到的cpu特性当前cpu是否支持。
8. 中断处理：注册中断处理的回调函数
9. 定时器：设置定时器的回调函数

### 3.1 EAL

DPDK应用程序是一个Linux用户态进程，使用pthread库。设备的pci信息和地址空间的访问时通过/sys内核接口和igb\_uio模块。Uio：用户态驱动。这部分内存通过mmap映射到应用程序中。

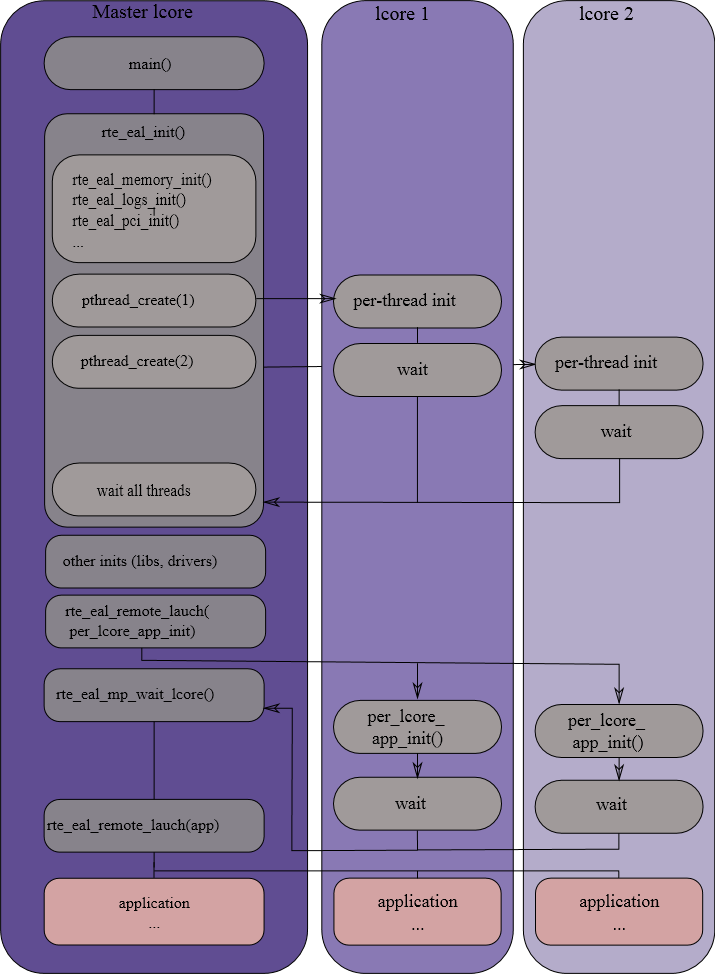
EAL分配内存是通过mmap()函数从hugetlbfs中获得，这部分内存的对dpdk服务层（如Mempool Library）可见。

由此可见，dpdk服务层在初始化完成后，通过pthread的亲和性调用，为每一个执行生成单独的用户态线程并绑定到指定的逻辑核上运行。

定时器的实现，则是通过mmap()，调用TSC(CPU Time-Stamp Counter)和HPET API来实现。

#### 3.1.1 初始化和核启动（core launching）

Dpdk最先初始化调用glibc的start function，然后检查编译参数中的架构配置参数是否支持当前运行的cpu。之后，main()函数开始执行，cpu核的初始化和启动由rte\_eal\_init()调用pthread库（如pthread\_self()、pthread\_create()、pthread\_setaffinity\_np()等）。



Note:初始化对象的时候，如memory zones, rings, memory pools, lpm tables 和hash tables，必须是在master core上完成，因为这些对象初始化函数并不是线程安全的，当然，一旦初始化完成，这些对象本身的使用，都是线程安全的。

### 3.1.2 多进程支持

Linuxapp eal同时支持多线程和多进程部署。

#### 3.1.3内存映射和内存保留

大片连续物理内存的分配使用hugetlbfs内核文件系统来完成，EAL提供了API来从这些连续内存中保留特定内存区域并给这片内存区域命名，被保留内存的物理地址由API 返回给用户。

Note：使用rte\_malloc库中的内存保留的API，也可以从hugetlbfs内核文件系统中保留内存，但是API返回的地址是逻辑地址而不是物理地址。

#### 3.1.4 dpdk支持xen dom0

Xen dom0不支持hugepages，dpdk通过rte\_dom0\_mm模块来支持xen dom0上的内存分配。

EAL通过ioctl的调用，通知linux内核模块rte\_dom0\_mm分配指定大小的内存，并获取所有内存段的信息；之后，eal使用mmap接口来映射已分配的内存；For each memory segment, the physical addresses are contiguous within it but actual hardware addresses are contiguous within 2MB？（什么意思）

#### 3.1.5 PCI访问

EAL使用内核提供的/sys/bus/pci公共接口来探测PCI总线的设备。

内核模块igb\_uio提供了/dev/uiox设备文件，该设备文件能够被映射到程序的pci地址空间并能够被访问。这利用了用户态驱动的特性。

3.1.6 per-lcore变量和共享变量

共享变量在多核中的行为与单核无异（共享变量所有线程访问的是同一份）。Per-lcore变量使用TLS(thread local storage)技术实现线程局部存储（这是一个水深处，汗）。

3.1.7 日志

EAL提供了日志API，在默认情况下，日志被发送syslog或者到控制台终端，同时，日志函数可以由用户重写成自己的日志机制。

3.1.7.1 跟踪和调试函数

有一些debug函数可以dump调用栈，rte\_panic()函数能够主动发送SIG\_ABRT信号，这个信号能够触发core file的生成（前提是开启ulimit）。

3.1.8 cpu特征识别

EAL能够在运行时探测cpu的特征，并检测哪些特征可用。

3.1.9 用户空间的中断和定时器处理

EAL创建一个host线程去轮询UIO 设备的文件描述符以探测中断。一个响应特定中断的回调函数能够在host线程中注册/注销，并被异步地调用。EAL能够通过定时回调的方式来响应NIC的中断。

Note：DPDK轮询模式中唯一支持的中断是检测网卡状态变化的中断。

3.1.10 黑名单

EAL PCI 设备黑名单的功能能够配置dpdk忽略哪些总线地址的NIC网口。

3.1.11 通用函数

支持各个体系架构上（i686和x86\_64）锁操作和原子操作

3.2 内存段（memory segments）和内存区域(memzone)

EAL将物理内存映射进来，由于物理内存有间隙，因此，映射进来的内存由一张表来描述，每个表项（rte\_memseg）代表了一段连续的内存。

Memzone 分配器的作用是保留（分配）连续的物理内存，分配出来的zones由特定的名字描述。

Rte\_memzone的描述项存放在configuration structure中，该结构体可以使用rte\_eal\_get\_configuration()函数访问到，还可以通过lookup方式查找到给定名称的rte\_memzone，并返回该memzone的描述项（描述项中包含该内存区域的物理地址）。

Memory zones can be reserved with specific start address alignment by supplying the align parameter (by default, they are aligned to cache line size). The alignment value should be a power of two and not less than the cache line size (64 bytes). Memory zones can also be reserved from either 2 MB or 1 GB hugepages, provided that both are available on the system.