

DPDK

Team Emertxe

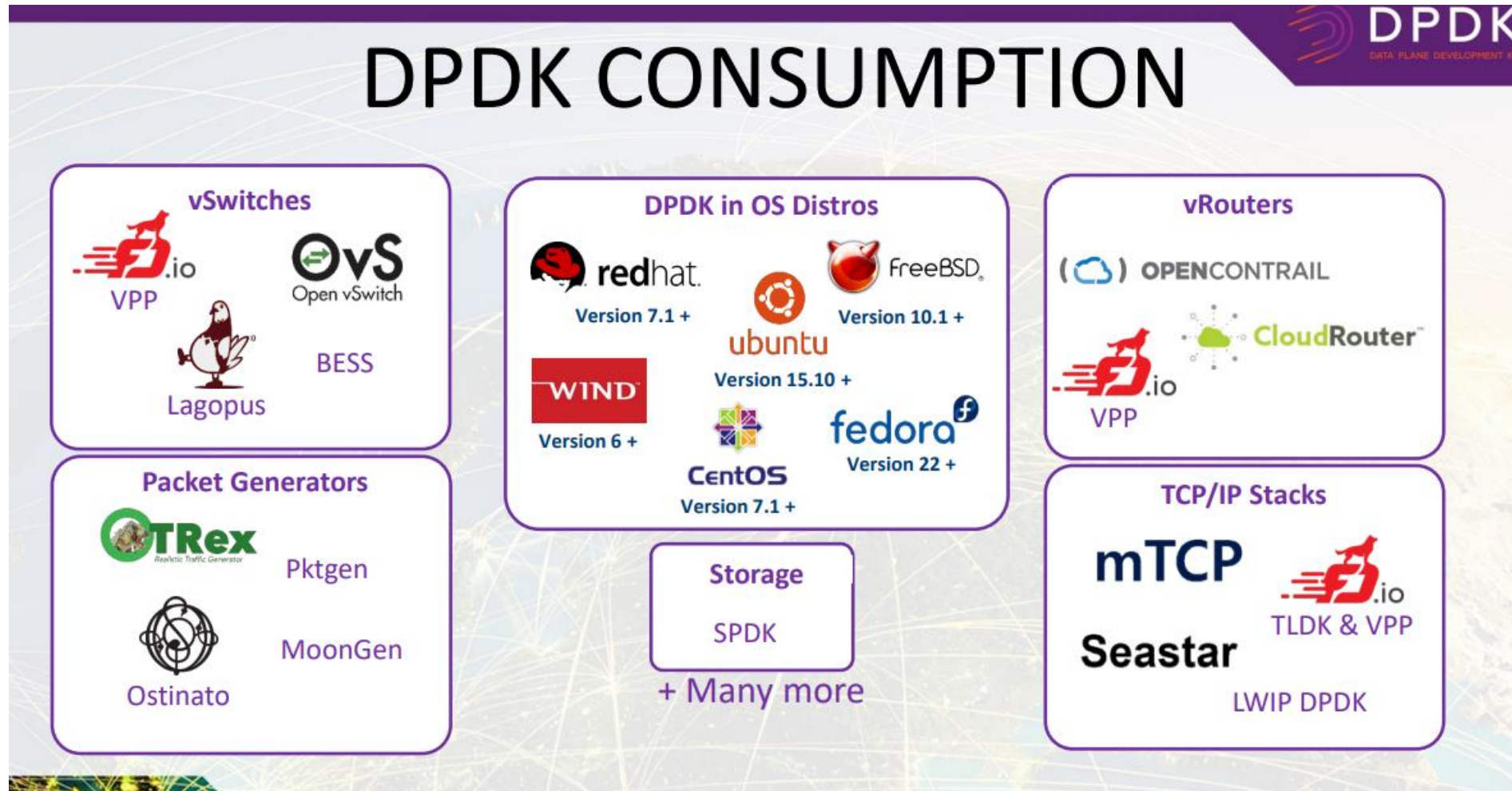


Topics



- DPDK Overview
- Why DPDK
- How to build
- DPDK code walkthrough

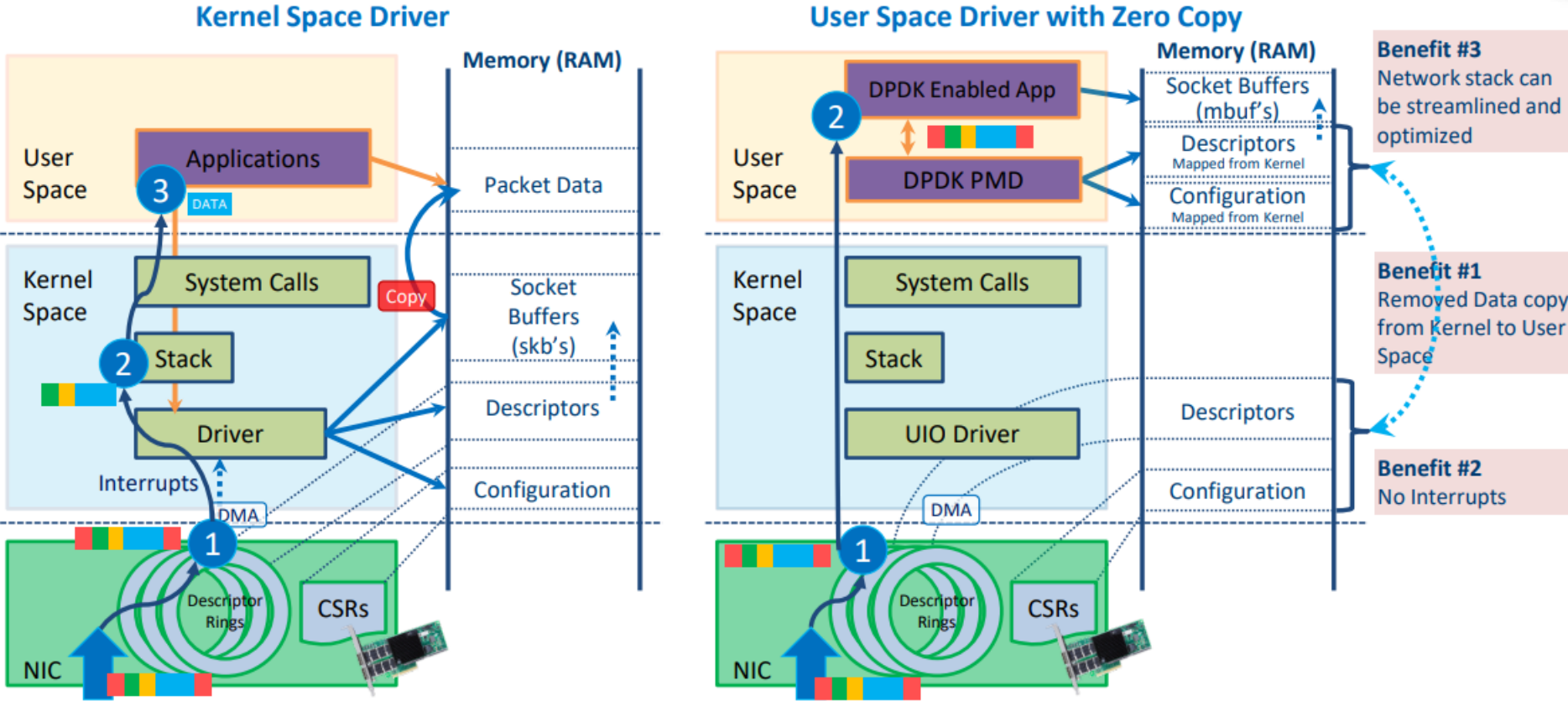
DPDK



DPDK



Packet Processing Kernel vs. User Space

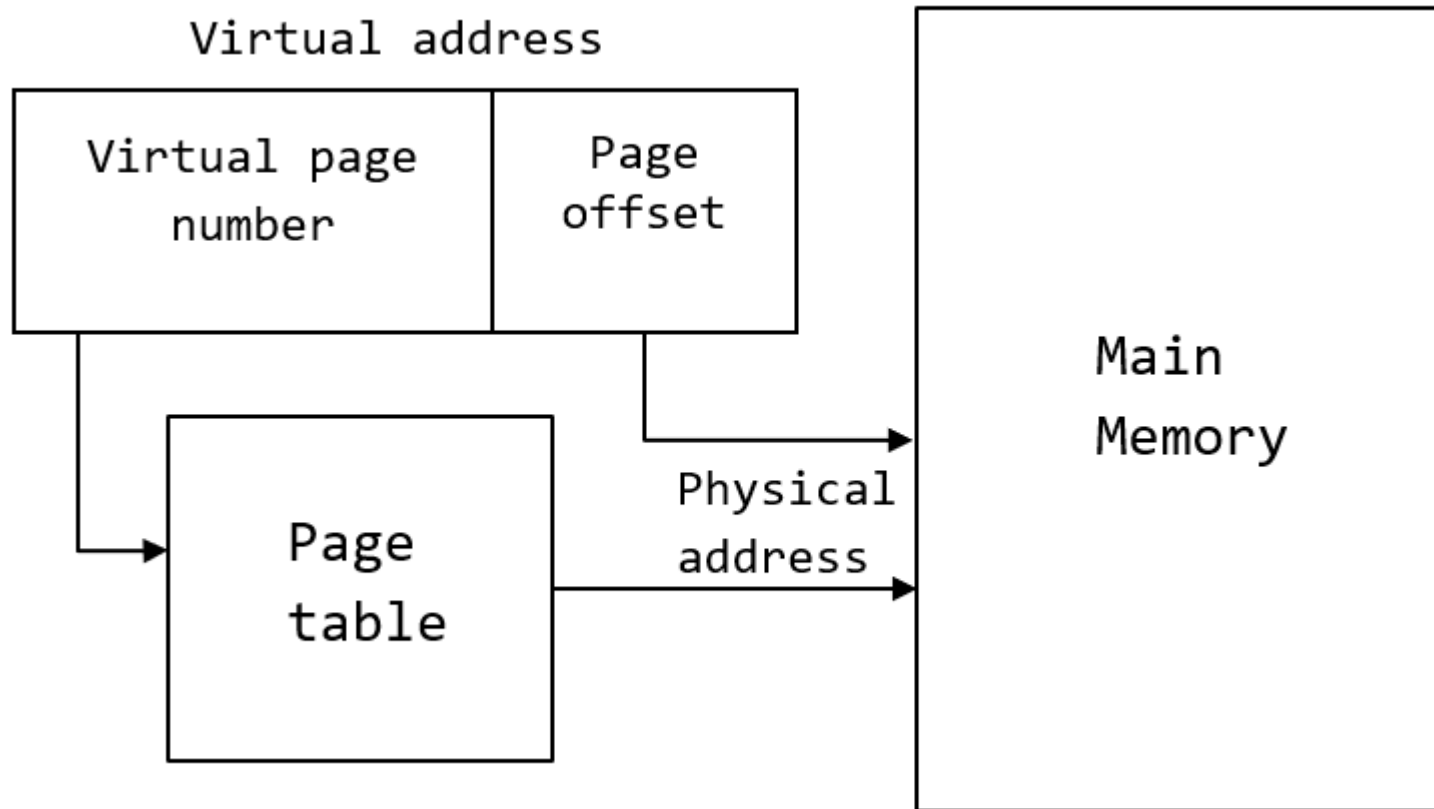


DPDK

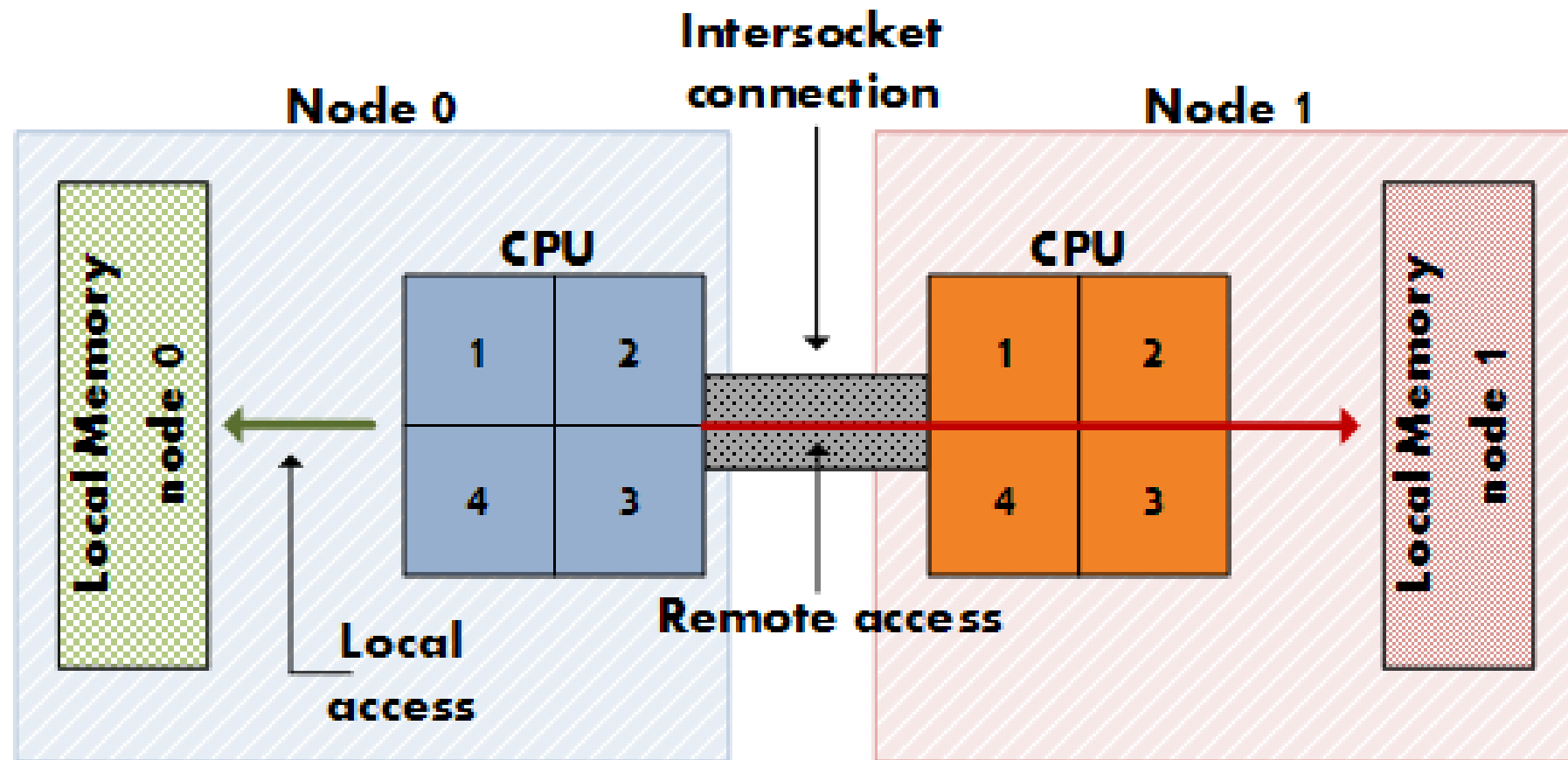


Why we need huge pages support

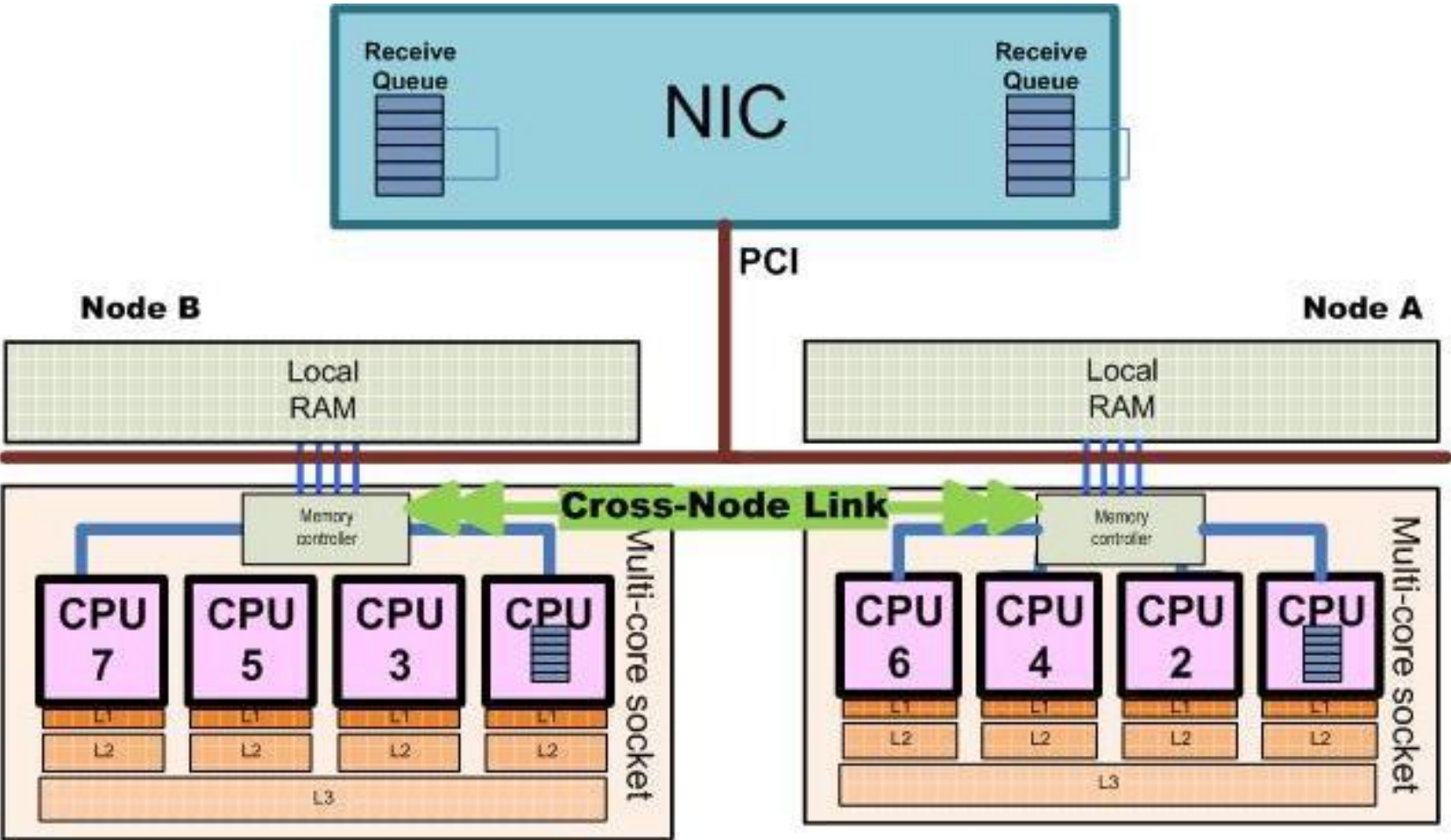
```
$sudo echo 64 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
```



DPDK



DPDK



DPDK

EAL

9.1.1. Lcore-related options

- `-c <core mask>`

Set the hexadecimal bitmask of the cores to run on.

- `-l <core list>`

List of cores to run on

The argument format is `<c1>[-c2][,c3[-c4],...]` where `c1`,
and 4095

9.1.4. Memory-related options

- `-n <number of channels>`

Set the number of memory channels to use.



DPDK

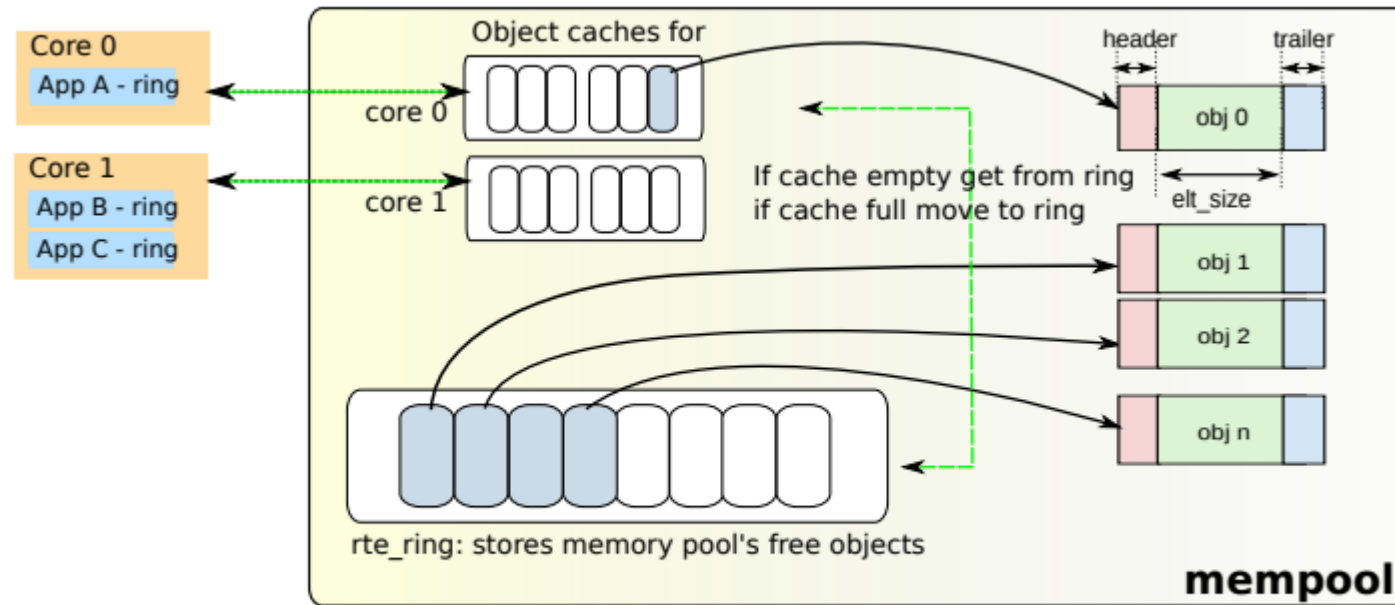
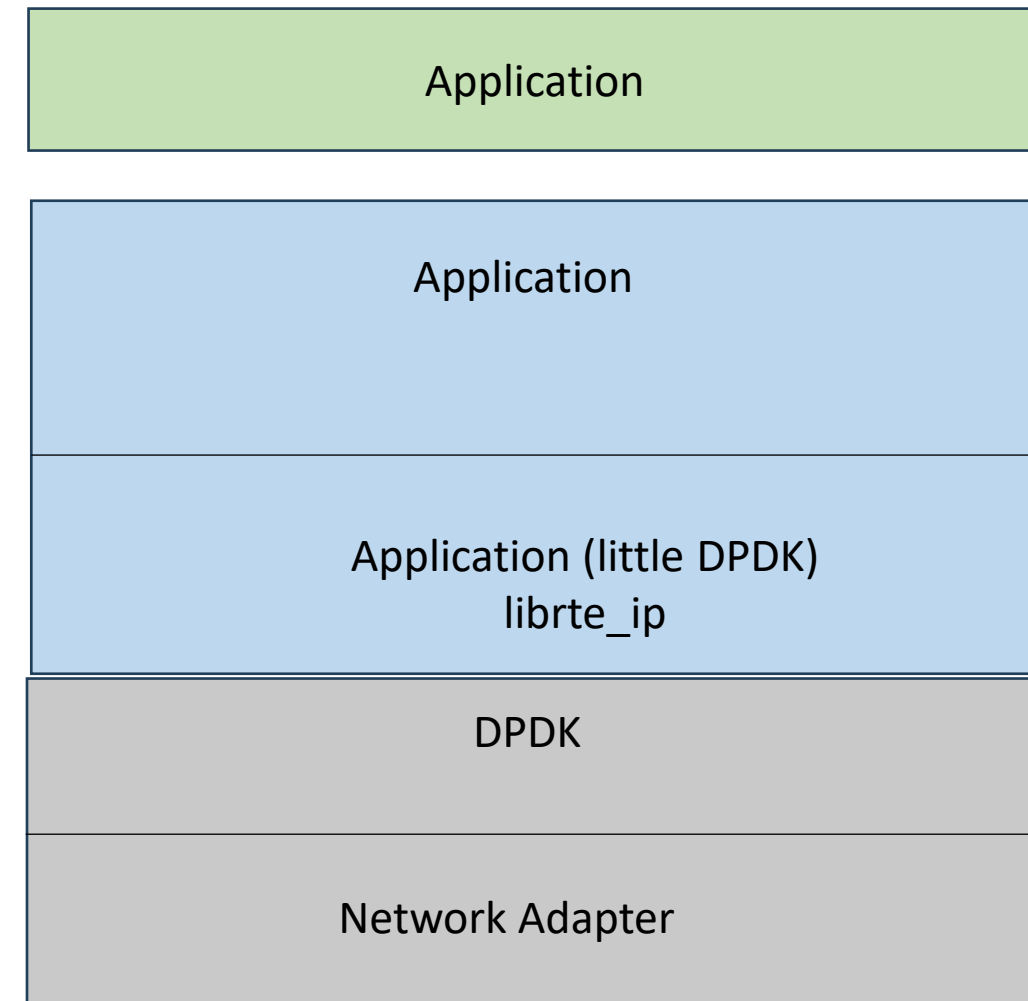
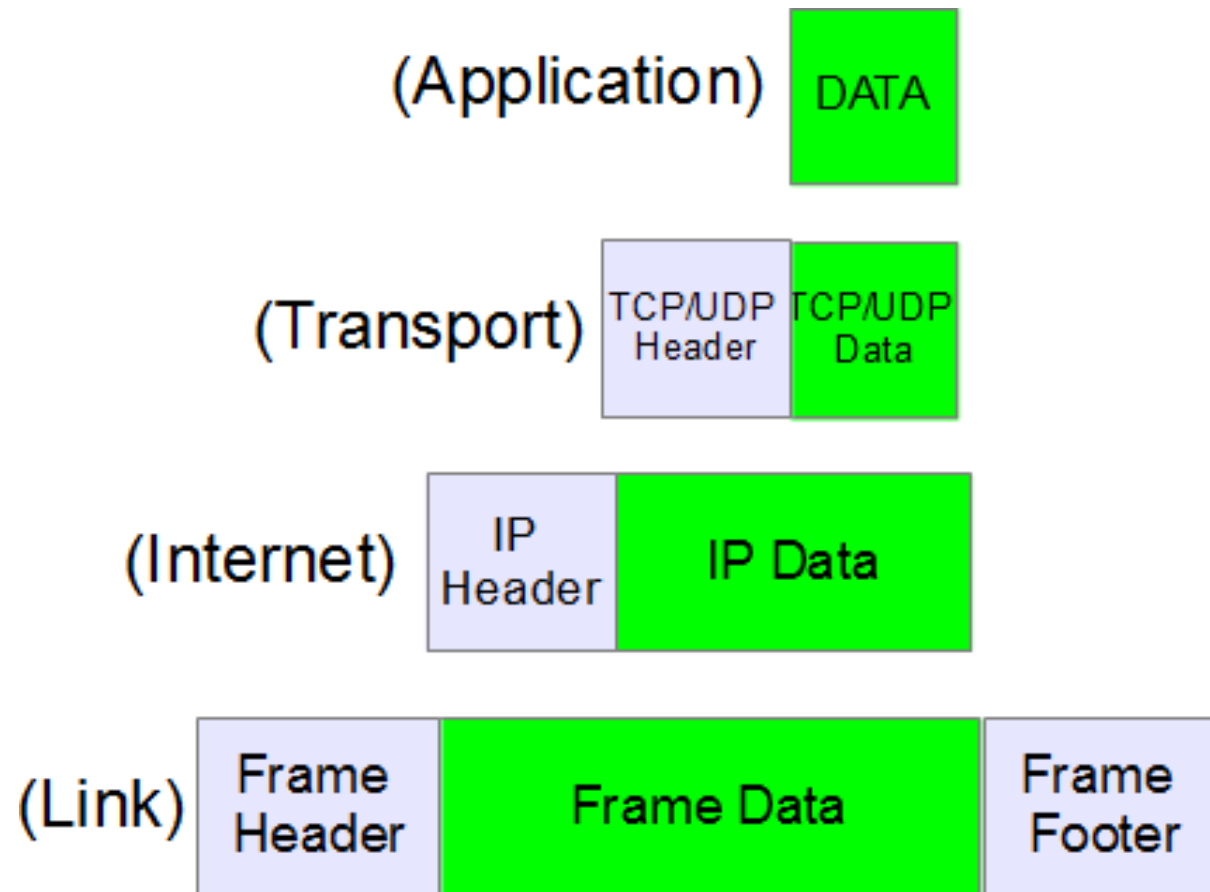


Figure 6.3: A mempool in Memory with its Associated Ring

DPDK



DPDK



NIC Driver: vmxnet3

- `void rte_pci_register(struct rte_pci_driver *driver)`
Populate “struct rte_pci_driver”

After registration, if user uses the device VMware:-

- `eth_vmxnet3_pci_probe()`
call wrapper function `rte_eth_dev_pci_generic_probe()`
Internally it will call `eth_vmxnet3_dev_init()`
`vmxnet3_dev_start()`
- `vmxnet3_dev_start()`
`vmxnet3_configure_msix()`
`vmxnet3_setup_driver_shared()`
`vmxnet3_dev_rxtx_init()`

DPDK

NIC Driver: vmxnet3

```
eth_dev->rx_pkt_burst =  
&vmxnet3_recv_pkts; ← To receive  
the packet
```

```
eth_dev->tx_pkt_burst =  
&vmxnet3_xmit_pkts; ← To send the  
packet
```

```
eth_dev->tx_pkt_prepare =  
vmxnet3_prep_pkts; ← Prepare the  
packet
```

DPDK



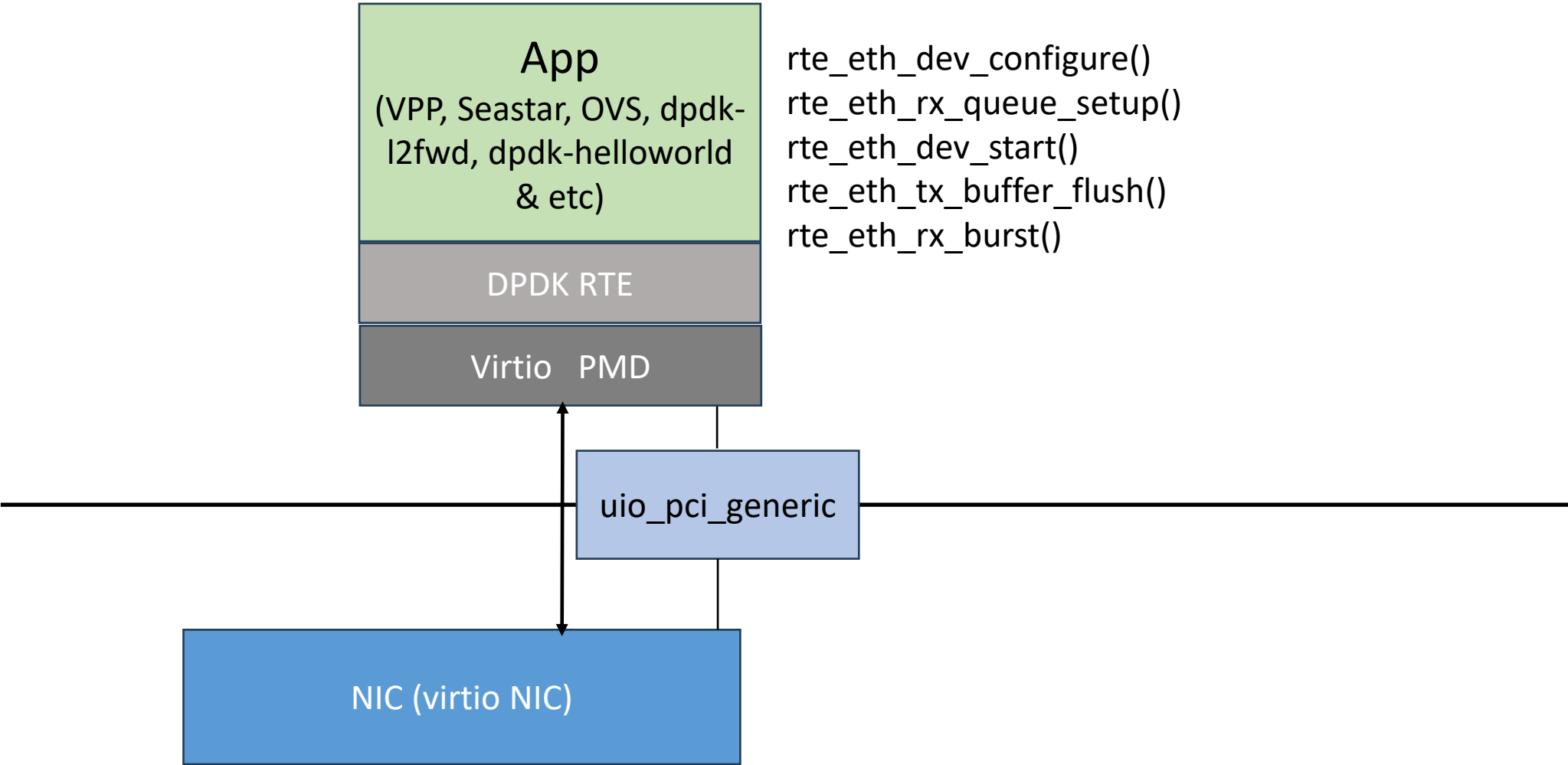
❖ Initializing the Environment Abstraction Layer (EAL):

- This should be the first API to be called. It initializes the EAL layer & makes way for the application to use the DPDK framework.

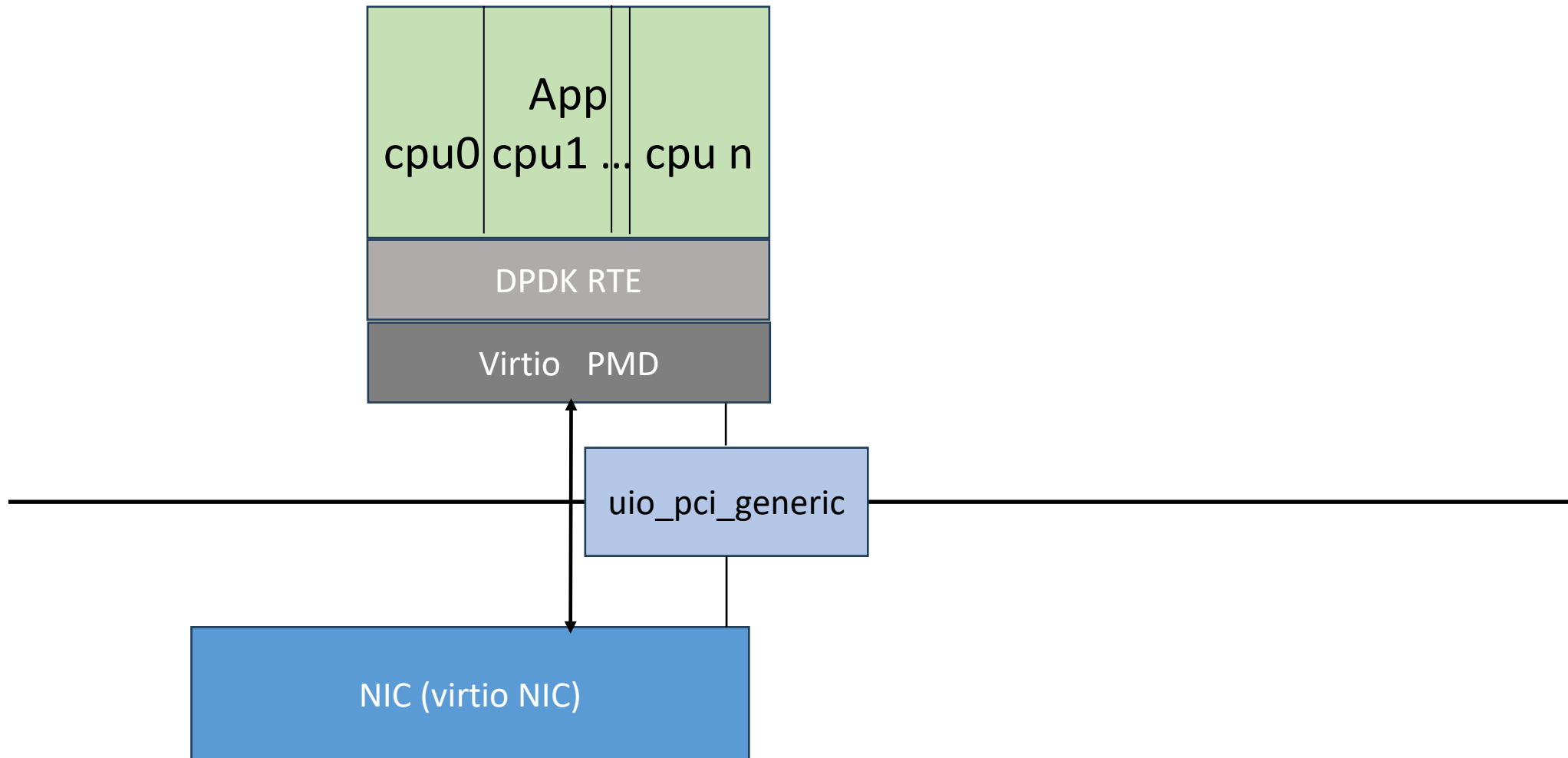
```
ret = rte_eal_init(argc, argv);
```

- *argc*: No. of command line arguments (both EAL & application specific parameters)
- *argv*: Array storing the command line arguments
- *ret*: On success, *ret* stores the no. of parsed arguments, which is equal to the no. of EAL parameters passed. The application can now use *argc* & *argv* to parse application specific parameters like any other normal C/C++ program using *int main(int argc, char *argv[])*.

DPDK



DPDK



DPDK & VPP

