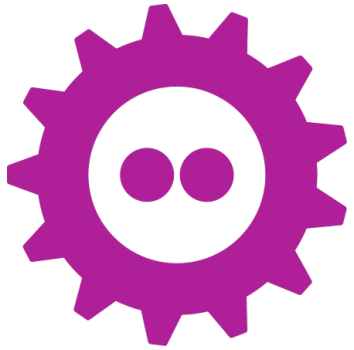


Introduction to dpdk-burst-replay tool

By Jonathan Ribas

FRAUDBUSTER

02/02/2019
Brussels

TABLE OF CONTENT

Tool objectives

01

First, tool identity document

Linux command line tool

Wrote in C, compiling on x86_64/arm64 archs.

Use DPDK stack

To achieve maximum replay performances.

Prerequisite packages

dpdk-dev and libnuma-dev.

Open source

Available at **git.dpdk.org** (mirrored on github) under BSD-3-clause license.

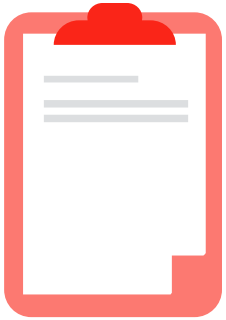
Early stage

Newborn tool, please be kind :)

Big community

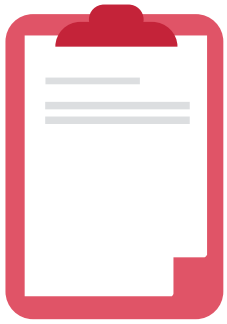
Of a single developer.. me.

Tool objectives



Fast

I want to use maximum hardware capabilities of network interface cards and PCI bus; without being struggled by HDD I/O and TCP/IP Linux stack (main tcpreplay bottlenecks).



Simple to use

Only 2 needed command line arguments are needed, the file and the port(s):

```
$> dpdk-replay foo.pcap 04:00.0
```

No need to understand or to configure manually DPDK.

Scriptable

As scripts are always useful and can facilitate continuous integration.

What can it do?

02

What can it do?

Load a pcap file

Standard input format for all common dumping/analyzing/replaying tools like tcpdump, tcpreplay, wireshark etc...

Put all loaded packets in cache(s)

To avoid reading the pcap file while its been replays.

Burst packets simultaneously on multiple ports

ATM, only NICs on the same NUMA can be addressed.

Run the same pcap several times in a row

It can be useful to extend a stress test without having to provide a bigger pcap file.

Output statistics

At the end, some stats are displayed for each ports (time spent, pkts-per-sec, total bitrate etc...)

Abstract DPDK stack

All EAL/mempool/ports initializations are automatically handled.

How to use it

03

How to use it

Be sure to have enough hugepages available

Add to your grub linux cmdline: **default_hugepagesz=1G hugepagesz=1G hugepages=4**

to create 4 hugepages of 1GB each.

Cf: *https://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt*

Bind wanted NIC ports to igb_uio

Here, just use the DPDK's bind script:

```
#> dpdk-devbind igb_uio 04:00.0 04:00.1 05:00.0 05:00.1
```

Then, just call dpdk-burst-replay:

```
#> dpdk-burst-replay --nbruns 10000 foo.pcap 04:00.0,04:00.1,05:00.0,05:00.1
```

It will burst 10000 times in a row the foo.pcap file in the four selected NIC ports.

Internal design

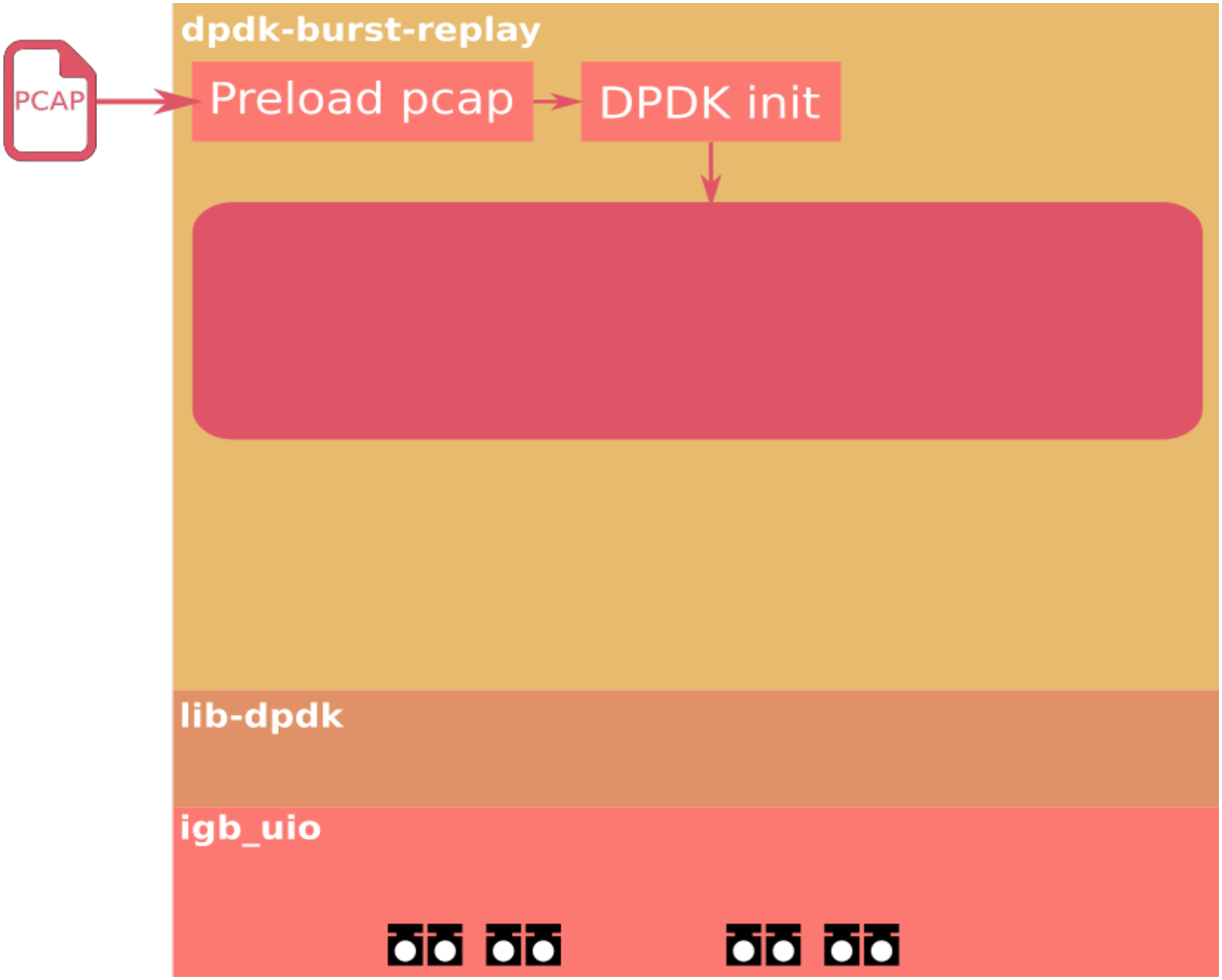
04

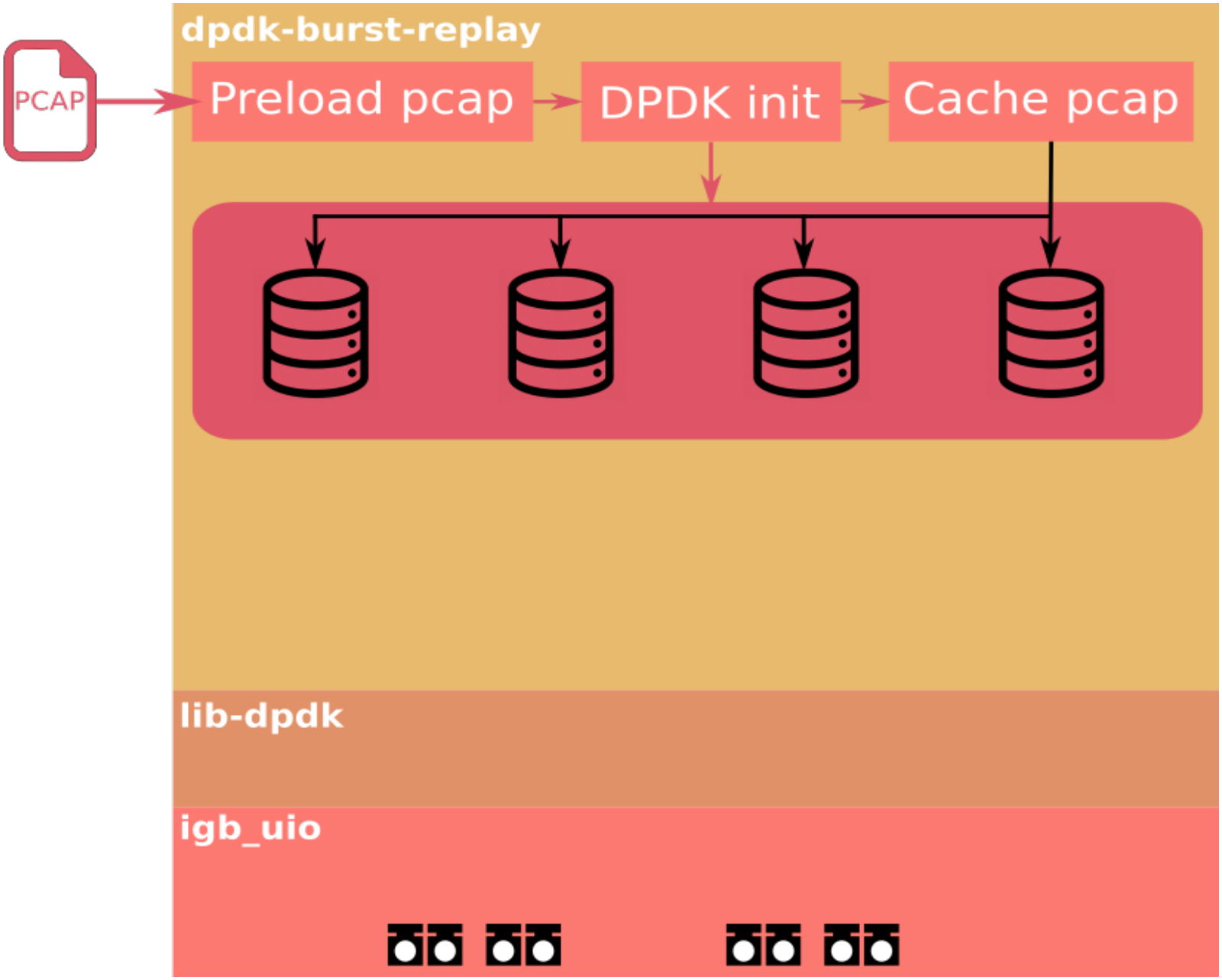
dpdk-burst-replay

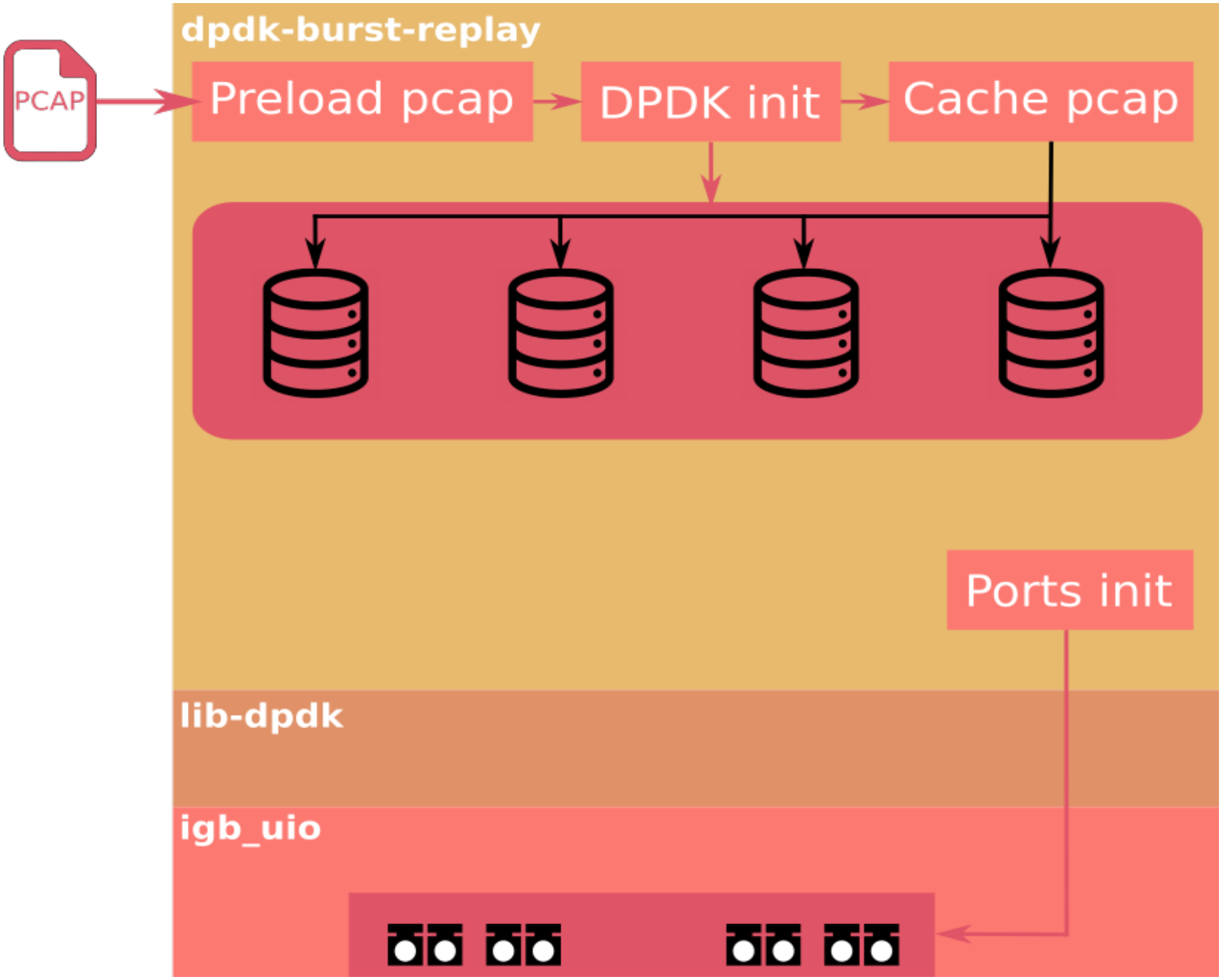
lib-dpdk

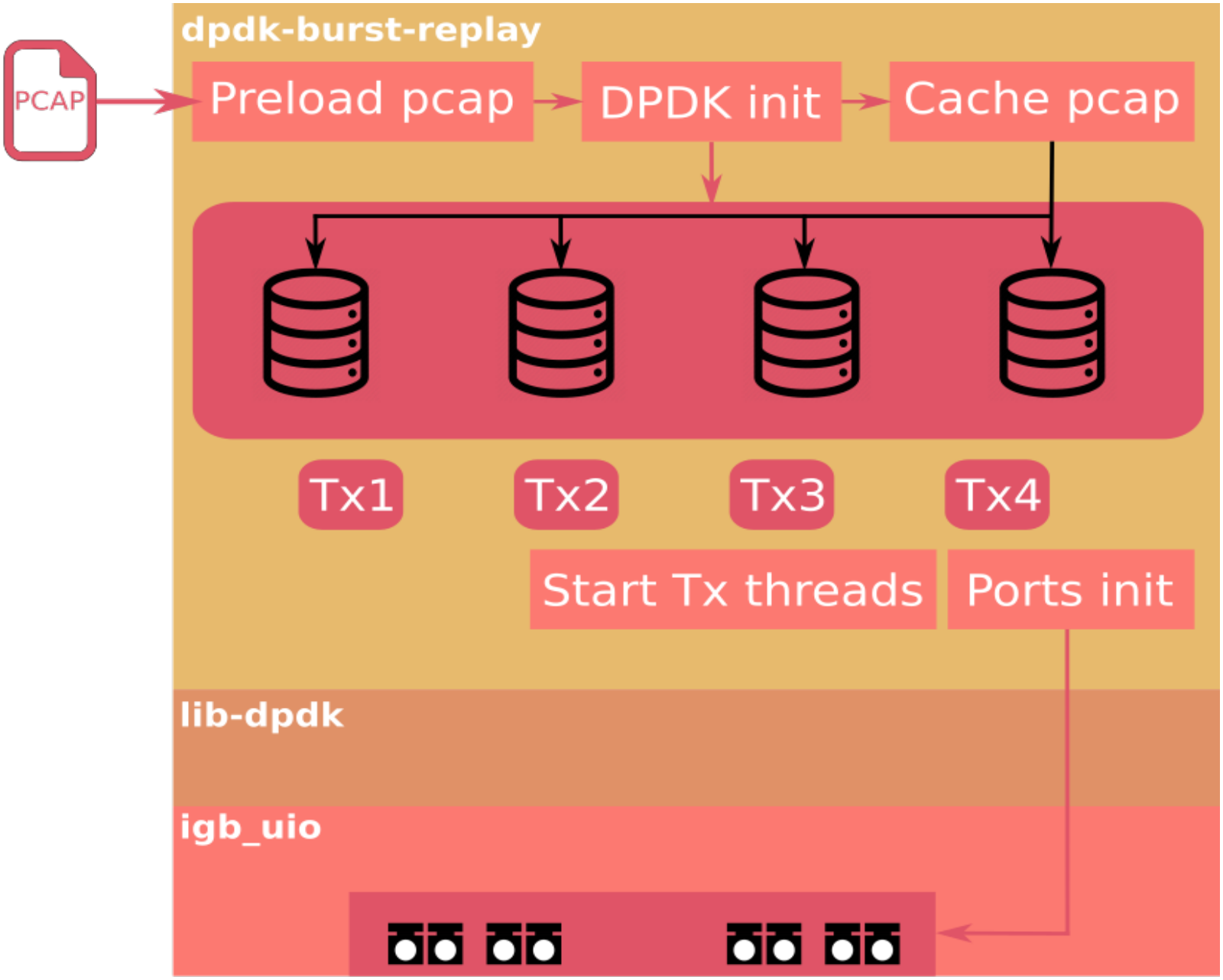
igb_uio

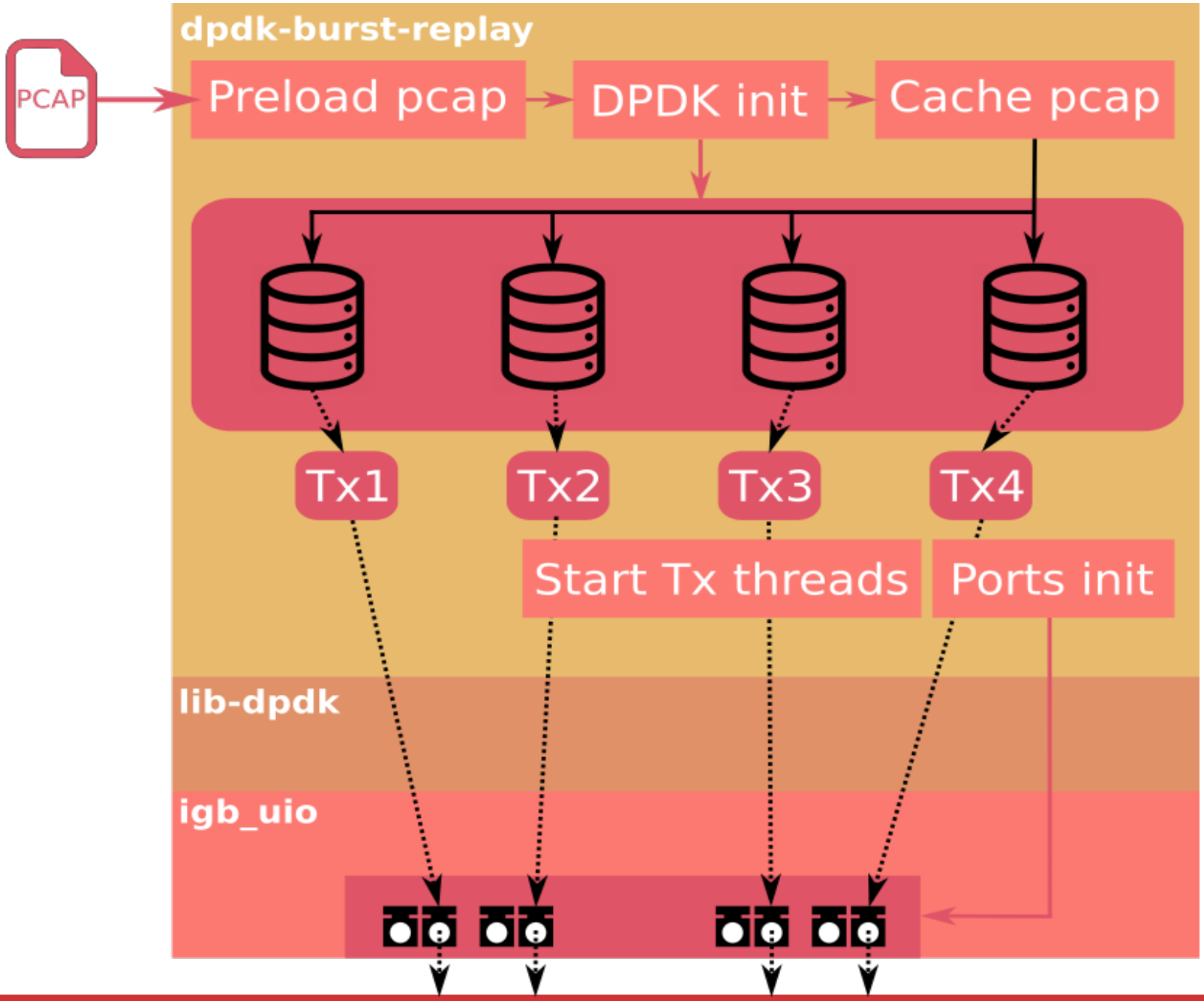

PCAP
dpdk-burst-replay
Preload pcap
lib-dpdk
igb_uio

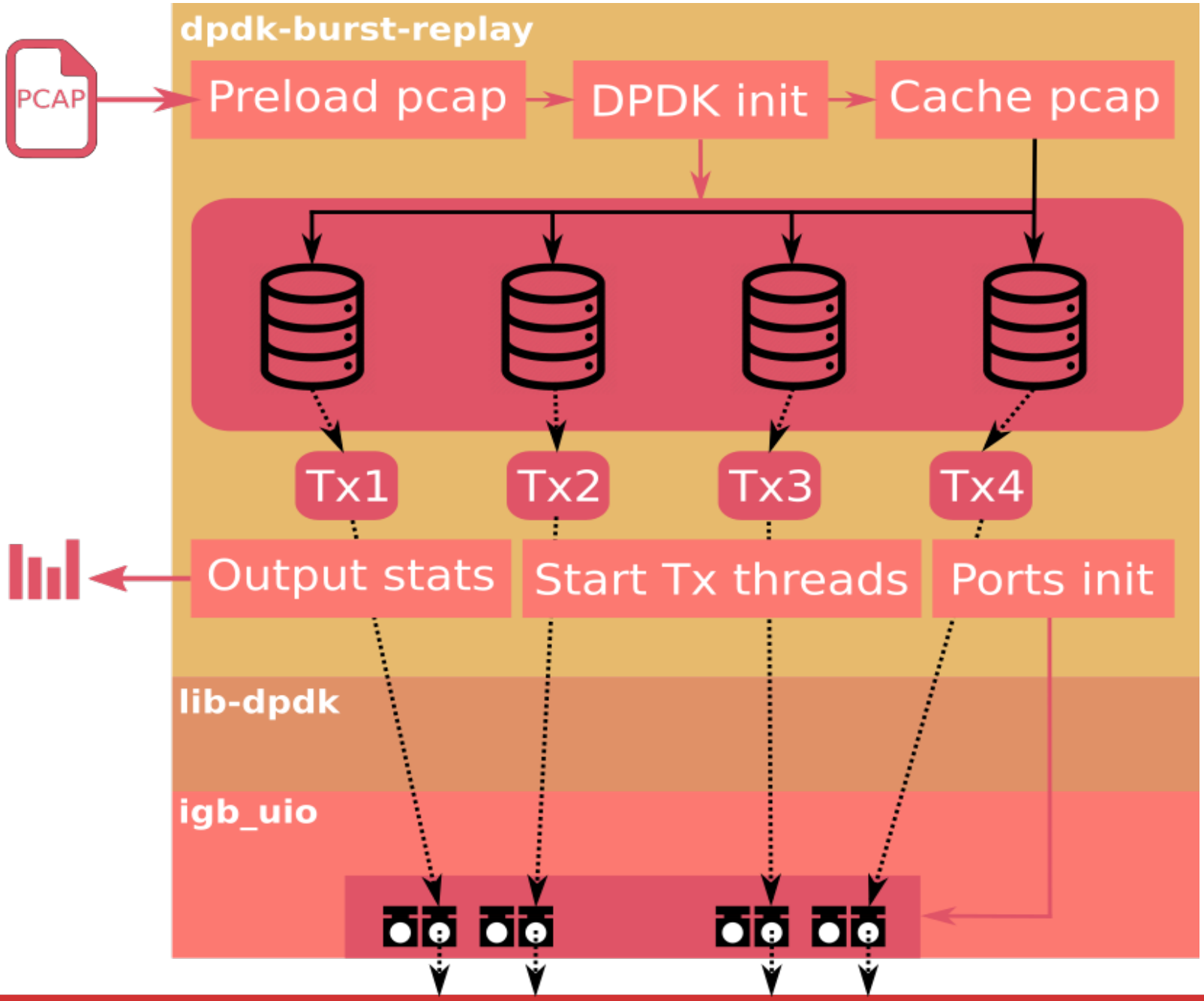

PCAP
dpdk-burst-replay
Preload pcap
DPDK init
lib-dpdk
igb_uio


PCAP
dpdk-burst-replay
Preload pcap
DPDK init
Cache pcap
lib-dpdk
igb_uio


PCAP
dpdk-burst-replay
Preload pcap
DPDK init
Cache pcap
Ports init
lib-dpdk
igb_uio


PCAP
dpdk-burst-replay
Preload pcap
DPDK init
Cache pcap
Tx1
Tx2
Tx3
Tx4
Start Tx threads
Ports init
lib-dpdk
igb_uio


PCAP
dpdk-burst-replay
Preload pcap
DPDK init
Cache pcap
Tx1
Tx2
Tx3
Tx4
Start Tx threads
Ports init
lib-dpdk
igb_uio


dpdk-burst-replay
PCAP
Preload pcap
DPDK init
Cache pcap
Tx1
Tx2
Tx3
Tx4
Output stats
Start Tx threads
Ports init
lib-dpdk
igb_uio

Terminal output


```
jribas@tchoio1:~/dpdk-burst-replay$ sudo src/dpdk-replay --numacore 1 --nbruns 10000 foo.pcap
82:00.0,82:00.1,84:00.0,84:00.1
preloading foo.pcap file (of size: 731207 bytes)
file read at 100.00%
read 4168 pkts (for a total of 731207 bytes). max paket length = 1301 bytes.
-> Needed MBUF size: 1430
-> Needed number of MBUFS: 32767
-> Needed Memory = 44.686 Mo
-> Needed Hugepages of 1 Go = 1
-> Needed cpus: 4
-> Create mempool of 32767 mbufs of 1430 octs.
-> Will cache 4168 pkts on 4 caches.
file read at 100.00%
-> NIC port 0 ready.
-> NIC port 1 ready.
-> NIC port 2 ready.
-> NIC port 3 ready.
Threads are ready to be launched, please press ENTER to start sending packets.

RESULTS :
[thread 00]: 8.962647 Gbit/s, 6856990.480036 pps on 6.078468 sec (0 pkts dropped)
[thread 01]: 8.962707 Gbit/s, 6857036.184448 pps on 6.078428 sec (0 pkts dropped)
[thread 02]: 8.962748 Gbit/s, 6857067.384277 pps on 6.078400 sec (0 pkts dropped)
[thread 03]: 8.962588 Gbit/s, 6856945.506091 pps on 6.078508 sec (0 pkts dropped)
-----
TOTAL        : 35.851 Gbit/s. 27428039.555 pps.
Total dropped: 0/166720000 packets (0.000000%)
jribas@tchoio1:~/dpdk-burst-replay$
```

Terminal output


```
jribas@tchoio1:~/dpdk-burst-replay$ sudo src/dpdk-replay --numacore 1 --nbruns 10000 foo.pcap
82:00.0,82:00.1,84:00.0,84:00.1
preloading foo.pcap file (of size: 731207 bytes)
file read at 100.00%
read 4168 pkts (for a total of 731207 bytes). max paket length = 1301 bytes.
-> Needed MBUF size: 1430
-> Needed number of MBUFS: 32767
-> Needed Memory = 44.686 Mo
-> Needed Hugepages of 1 Go = 1
-> Needed cpus: 4
-> Create mempool of 32767 mbufs of 1430 octs.
-> Will cache 4168 pkts on 4 caches.
file read at 100.00%
-> NIC port 0 ready.
-> NIC port 1 ready.
-> NIC port 2 ready.
-> NIC port 3 ready.
Threads are ready to be launched, please press ENTER to start sending packets.

RESULTS :
[thread 00]: 8.962647 Gbit/s, 6856990.480036 pps on 6.078468 sec (0 pkts dropped)
[thread 01]: 8.962707 Gbit/s, 6857036.184448 pps on 6.078428 sec (0 pkts dropped)
[thread 02]: 8.962748 Gbit/s, 6857067.384277 pps on 6.078400 sec (0 pkts dropped)
[thread 03]: 8.962588 Gbit/s, 6856945.506091 pps on 6.078508 sec (0 pkts dropped)
-----
TOTAL        : 35.851 Gbit/s. 27428039.555 pps.
Total dropped: 0/166720000 packets (0.000000%)
jribas@tchoio1:~/dpdk-burst-replay$
```

Future evolutions

05

Some ideas of future evolutions

- **Add an option to configure maximum bitrate.**
- **Add an option to send the pkts respecting the timings**
- **Be able to select multiple pcap files at once.**
- **Optimization of memory usage.**
- **Add a Python module to facilitate scripting.**
- **Add a configuration file to tune internal parameters.**
- **Waiting for YOUR needs/ideas/feedback.**

THANK YOU

QUESTIONS ?

http://git.dpdk.org/apps/dpdk-burst-replay

FRAUDBUSTER