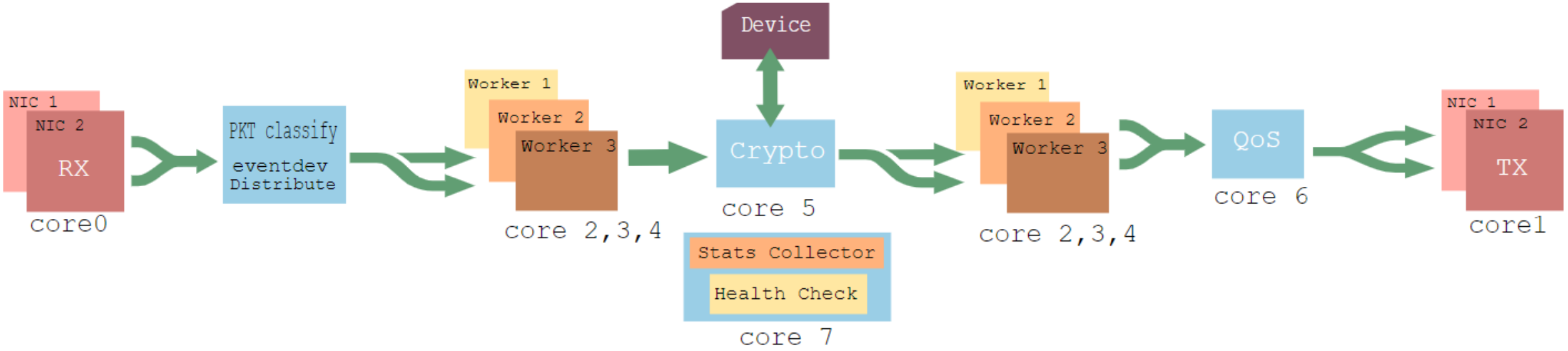

# 11 Minutes to Debug, Troubleshoot and Analyze

PRESENTED BY
VIPIN VARGHESE

# Packet Life - Overview



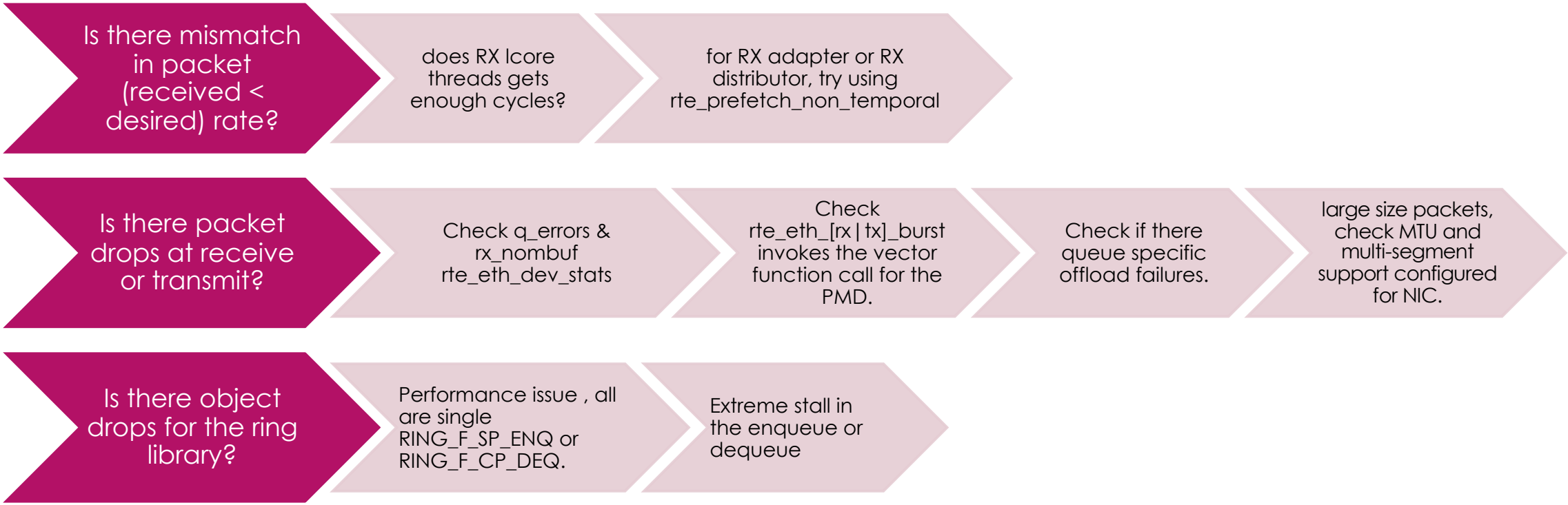

- Applications also can be modeled as
  - single or multiple primary processes.
  - single primary and single secondary.
  - single primary and multiple secondaries.


# AGENDA

"It is tedious to isolate, debug, and understand various behaviors which occur randomly or periodically. The goal of the guide is to consolidate a few commonly seen issues for reference. Then, isolate to identify the root cause through step by step debug at various stages."

1. Explore the cause of drops.
2. Debug and troubleshoot guide.
3. Changes in pro-info.
4. Malloc Scanner.
5. Future work.

# Bottleneck Analysis



# Bottleneck Analysis - Contd

DPDK
DATA PLANE DEVELOPMENT KIT

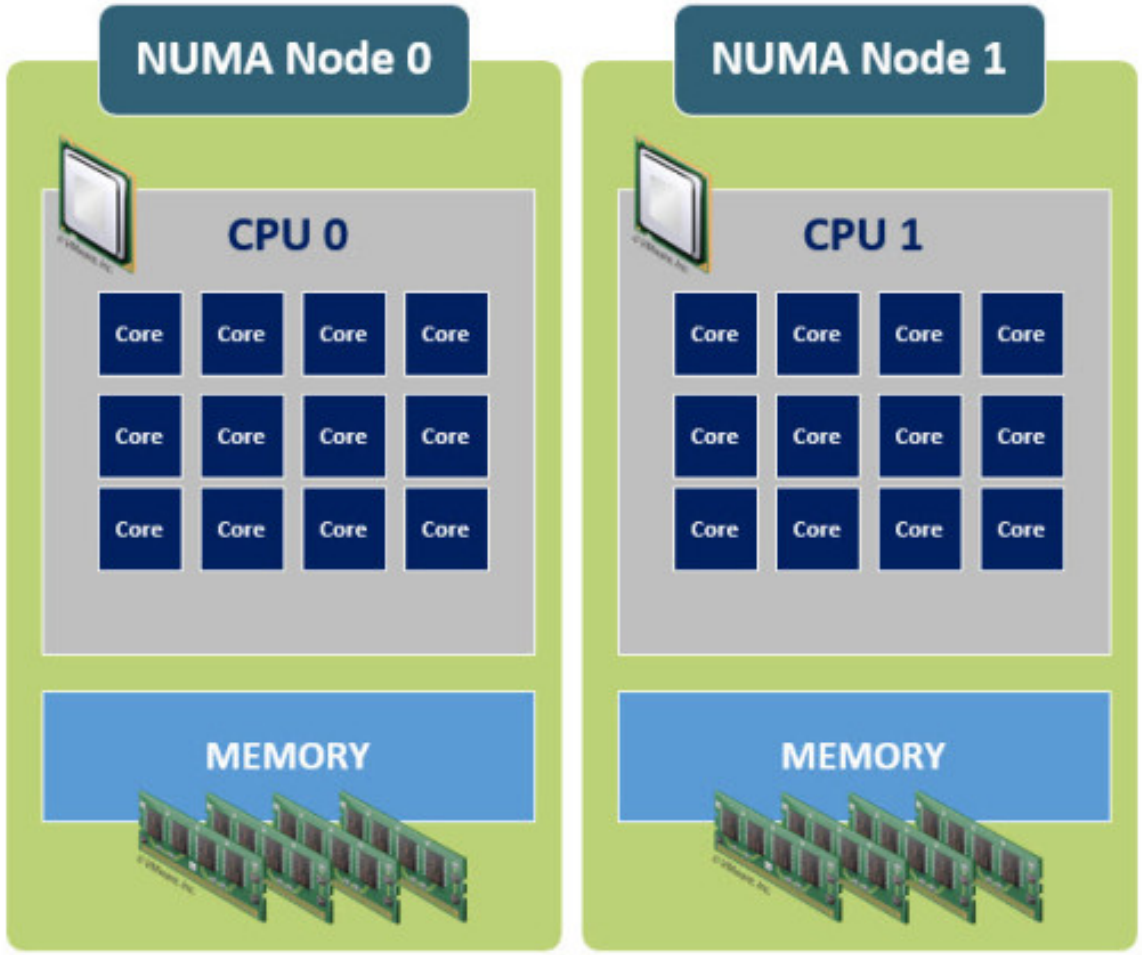
| | | | | |
|---|---|---|---|---|
| Is there a variance in packet or object processing rate in the pipeline? | Are we linearize multi-segment; software offload like checksum, TSO, and VLAN strip. | Packet or object buffer copy if not done selectively. | Application logic sets rte_pktmbuf_refcnt_ set to higher and holds longer. | If absent for larger huge pages, try rte_mem_lock_page. |
| Is there a variance in cryptodev performance? | Parallelize enqueue \| dequeue thread for varied multiple queue pair. | software-assist is in use, ensure the library is built with right (SIMD) flags. | hardware-assist is in use, ensure both firmware and drivers are up to date. | |
| Is user functions performance is not as expected? | functions performance depends upon context switches (use lcore threads). | rte_thread_get_affi nity to isolate functions running on the same CPU core. | Use of 'objdump' to ensure opcode is matching to the desired uops. | |
| Is the execution cycles for dynamic service functions are not frequent? | Run parallel on all cores with RTE_SERVICE_CAP_ MT_SAFE. | If services share the lcore, overall execution should fit budget. | | |

# Bottleneck Analysis - Contd

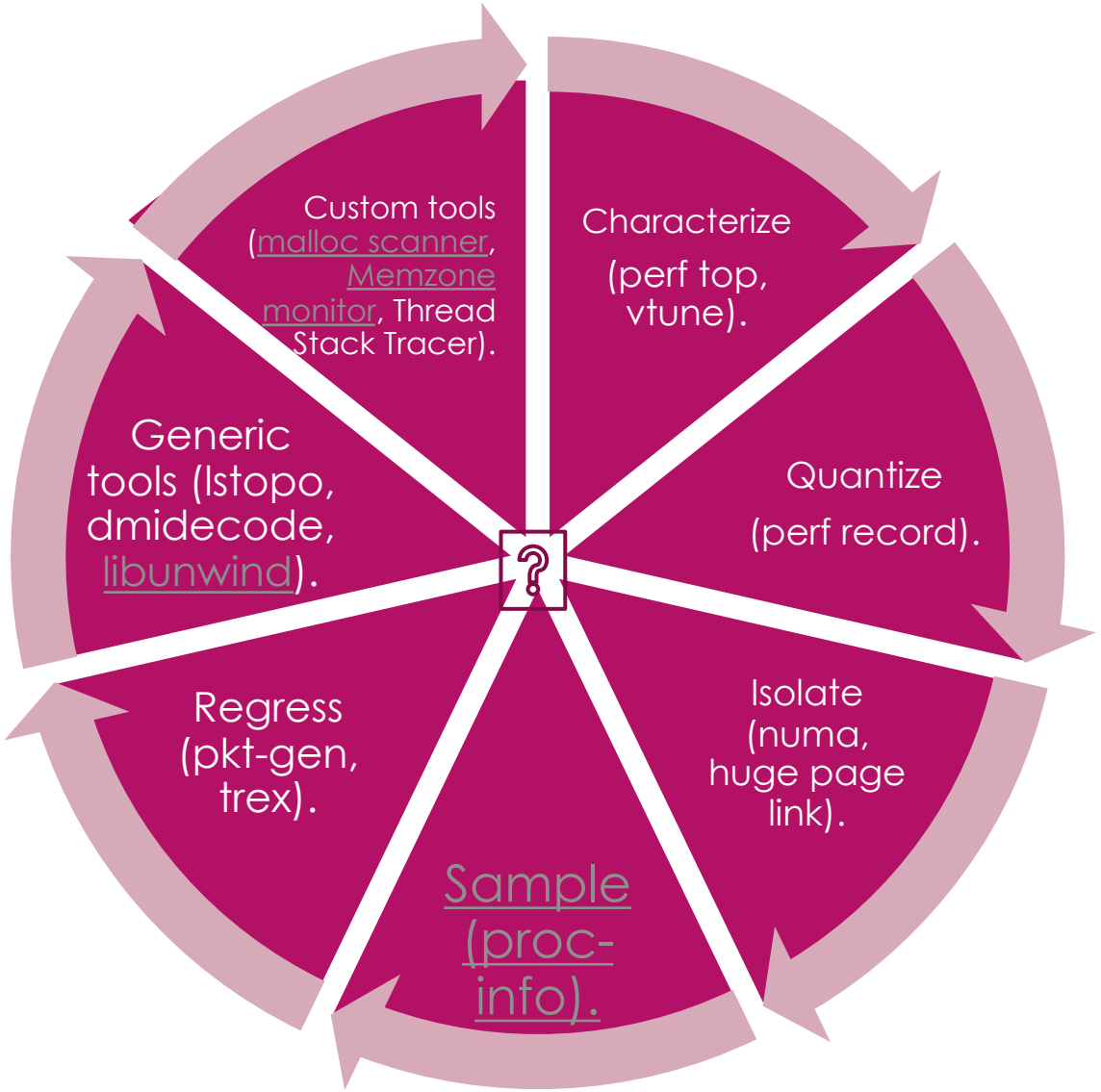
| | | | |
|---|---|---|---|
| Is there a bottleneck in the performance of eventdev? | Check for loops between event queue. | identify the starvation in queues. | Check for stalls using in-flight events. |
| Is there a variance in traffic manager? | If SW pipeline - insufficient CPU cycles. | flow drops can be narrowed down to WRED, priority, and rates limiters. | rte_tn_get_number_of_leaf_node and flow table to pin down the leaf. |
| Is the packet not in the unexpected format? | First, isolate at NIC entry and exit. | Second, isolate at pipeline entry and exit. | |
| Does the issue still persist? | Check for errors due to vendor \| app META data. | In multi-process check possible errors in configuration and data corruption in the data plane. | Try using the cache allocation technique to minimize the effect between applications. |

# What are the various drop?

- Memory misconfiguration.
- NUMA access.
- Miscalculation in number of objects in pool.
- Unused cached objects per lcore.
- Turbo boosting SISD instead SIMD.
- Allowing single or multiple physical cores run at full boost.
- Cache misaligned.
- Exception path overflowing with packet scenarios.



# How can the drops be identified?

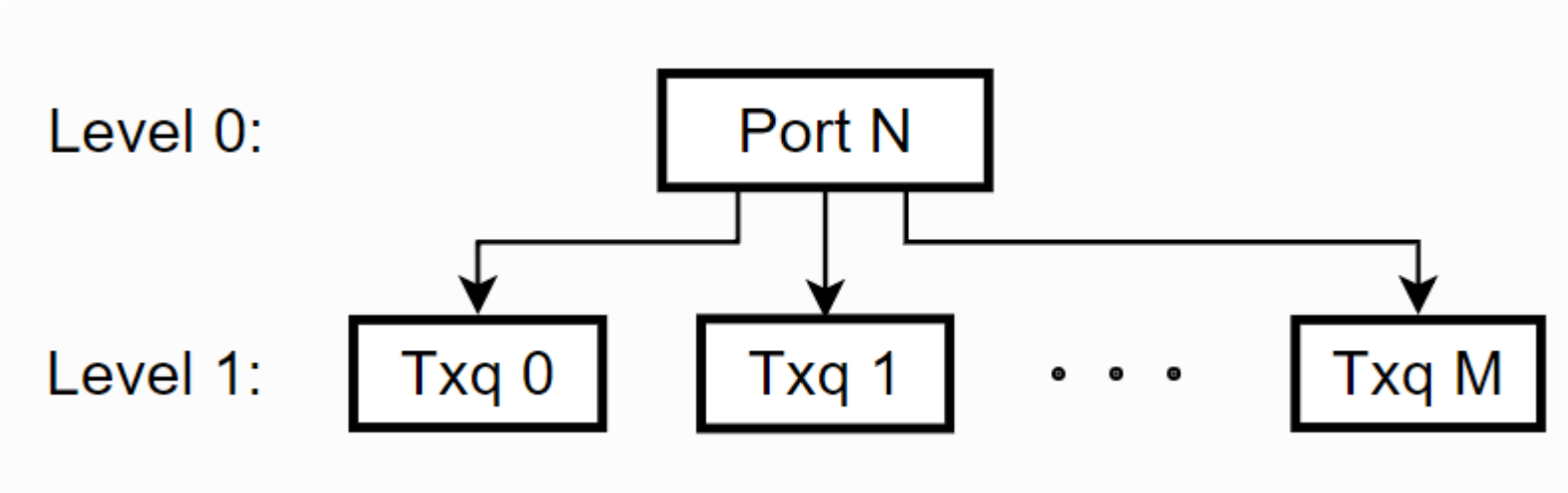


# Why are there drops at stages?

DPDK
DATA PLANE DEVELOPMENT KIT

- Memory access.
- Configuration.
- Insufficient objects in pool.
- Variance in Platform (BUS or CPU speed).
- Misconfiguration.
- Thermal-Power throttling.
- No or over optimizations (Backend queuing up).
- Intermediate or burst exceptions.

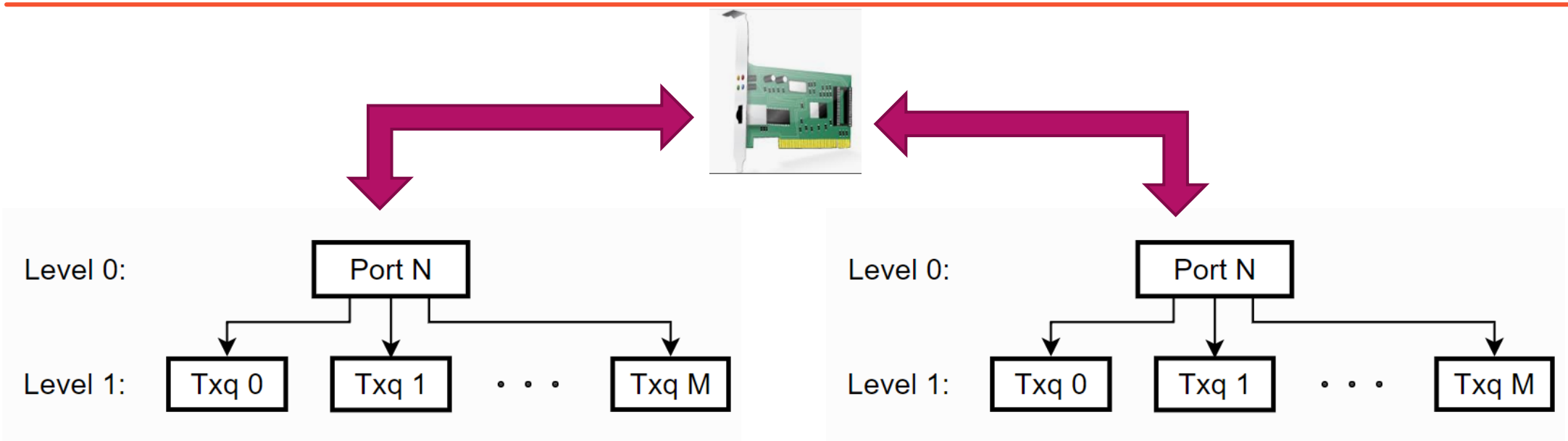
# Traffic Manager Issue (PF)?




If it is multi process with shared interface (primary secondary)

- Who owns TM (Primary or Secondary)?
- Who owns root node (Primary or Secondary)?
- How will validation for shapers takes places (Primary | Secondary)?
- Who can call TM API ((Primary and Secondary))?
- Primary configures and Secondary fetch stats (PMD implementation dependent)
- Primary and Secondary share multiple queues.

# Traffic Manager Issue (on PF-VF)?




If it is single or multi process with PF -VF

- Who owns PF is owned by Kernel (Traffic Class [tc])?
- Who owns PF is owned by DPDK, root node by PF or VF or both?
- Can root be shared for multiple VF with max capacity (who validates TM configuration)?

# Can I run in non root user?

- Create non root user with no sudo or root permission.
- Create workspace.
- Change ownership of huge page mount area for non root user.
- export XDG_RUNTIME_DIR=<user workspace>
- Bind interface with igb_uio | uio_pci_generic | vfio_genreric.
- Change ownership device to user.
- Change ownership of application to user.
- Run application with huge-dir user mount area.

# Future Work

## Proc-Info

- Eventdev
- Compressiondev
- ACL detailed dump (Key & Result)
- LPM result dump

## Debug & Troubleshoot guide

- More use cases
- Runs as non root
- Cache partitioning
- How to run application as non root user.

## Tools

- Share malloc-free scanner & Memzone monitor as RFC or github
- eBPF or BPF based debug on user space DPDK libraries and PMD.

DPDK
DATA PLANE DEVELOPMENT KIT

# Thanks All, Q & A?

SCREEN SHOTS AS BACKUP

| | |
|---|---|
| LICENSE | Create LICENSE |
| Makefile | Add files via upload |
| README.md | Update README.md |
| contextunwind.c | Add files via upload |
| contextunwind.h | Add files via upload |
| main.c | Add files via upload |
| main_org.c | Add files via upload |

README.md

## DPDK-THREADTRACE-WITHOUTGDB

dump all threads stack and register information. useful in target where no gdb or pdump is not available.

- [Depdency]

> libunwind, pthread_self (on each thread whose back trace and register extract is required.)

- [BUILD]

> CFLAGS += -L/usr/lib/x86_64-linux-gnu/ -lunwind , LDFLAGS += -L/usr/lib/x86_64-linux-gnu/ -lunwind

### Use Cases

- Binary are stripped.
- Binary and Application have no debug symbols.

| | | |
|---|---|---|
| LICENSE | Create LICENSE | 21 days ago |
| README.md | Update README.md | 21 days ago |
| main.c | Add files via upload | a month ago |

README.md

## DPDK-MALLOCFREE-SCAN

### Purpose

debug tool for accumlating information on `rte_calloc|zalloc|malloc` and rte_free in DPDK

### Motivation

DPDK allocas like rte_malloc, rte_calloc and rte_zalloc does not map alloc region name to address. The variables are unsed. This makes it difficult to track the uasage on dynamically allocates instance.

### Solution

- Create a container **to hold malloc|calloc|zalloc name, pointer and size**.
- on every successfull allocation fetch an element holder from fb_array and update the details.

| | | |
|---|---|---|
| LICENSE | LICENSE | 21 days ago |
| Makefile | Add files via upload | a year ago |
| README.md | Update README.md | a month ago |
| main.c | Add files via upload | a year ago |
| main.c_bkp | Add files via upload | a year ago |

README.md

## DPDK-MEMZONEMONITOR

### Motivation

DPDK applications can be modeled with single run to completion or multi pipe line stage completion models. Single or multiple process can itnerac twith packets and data alike. Hence corruption in any of the data, lookup tabels or counters in hard to isolate. This can occur when certan excpetion code process is taken hence diffcult to reproduce too.

### Solution

We can isolate above scenarios if low overhead process monitors changes occuring in a table or counters which is not expected to update. Using DPDK multi-process model we can get access to all huge page content from primary or othe rmulti process applciations

```
./$(RTE_TARGET)/app/dpdk-procinfo -- -m | [-p PORTMASK] [--stats | --xstats |
--stats-reset | --xstats-reset] [ --show-port | --show-tm | --show-crypto |
--show-ring[=name] | --show-mempool[=name] | --iter-mempool=name ]
```

# DPDK-MALLOCFREE-SCANNER

DPDK
DATA PLANE DEVELOPMENT KIT

## Primary

ADK_MALLOC_REGIONS 0x1000013f8 addr 0x100330f40hello from core 4

rte_malloc for 16 bytes!

name 0-0 0x100330ec0

index 0rte_malloc for 16 bytes!

name 0-1 0x100330e40

index 1rte_malloc for 16 bytes!

name 0-2 0x100330dc0

index 2rte_malloc for 16 bytes!

name 0-3 0x100330d40

index 3rte_malloc for 16 bytes!

name 0-4 0x100330cc0

index 4rte_malloc for 16 bytes!

name 0-5 0x100330c40

index 5rte_malloc for 16 bytes!

name 0-6 0x100330bc0

## Secondary

Process is RTE_PROC_SECONDARY!

ADK_MALLOC_REGIONS 0x1000013f8 addr 0x100330f40

element index 0, fetch 2, count 0

name 0-0 ptr 0x100330ec0

name 0-1 ptr 0x100330e40

element index 2, fetch 2, count 2

name 0-2 ptr 0x100330dc0

name 0-3 ptr 0x100330d40

element index 4, fetch 2, count 4

name 0-4 ptr 0x100330cc0

name 0-5 ptr 0x100330c40

element index 6, fetch 2, count 6

name 0-6 ptr 0x100330bc0

name 0-7 ptr 0x100330b40

## 11.2. Bottleneck Analysis

A couple of factors that lead the design decision could be the platform, scale factor, and target. This distinct preference leads to multiple combinations, that are built using PMD and libraries of DPDK. While the compiler, library mode, and optimization flags are the components are to be constant, that affects the application too.

### 11.2.1. Is there mismatch in packet (received < desired) rate?

RX Port and associated core Fig. 11.2.

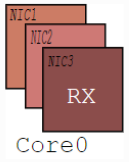



*Fig. 11.2 RX packet rate compared against received rate.*

1. Is the configuration for the RX setup correctly?
   - Identify if port Speed and Duplex is matching to desired values with rte_eth_link_get.
   - Check MTU value is set to the expected packet size to receive with rte_eth_get_mtu if there are large packet drops.
   - Check promiscuous mode if the drops do not occur for unique MAC address with rte_eth_promiscuous_get.
2. Is the drop isolated to certain NIC only?
   - Make use of rte_eth_dev_stats to identify the drops cause.
   - If there are mbuf drops, check nb_desc for RX descriptor as it might not be sufficient for the application.
   - If rte_eth_dev_stats shows drops are on specific RX queues, ensure RX lcore threads has enough cycles for rte_eth_rx_burst on the port queue pair.
   - If there are redirect to a specific port queue pair with, ensure RX lcore threads gets enough cycles.
   - Check the RSS configuration rte_eth_dev_rss_hash_conf_get if the spread is not even and causing drops.
   - If PMD stats are not updating, then there might be offload or configuration which is dropping the incoming traffic.

### 11.2.2. Is there packet drops at receive or transmit?

RX-TX port and associated cores Fig. 11.3.

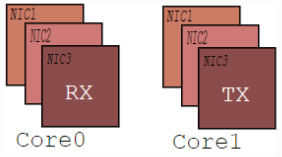



*Fig. 11.3 RX-TX drops*

1. At RX
   - Identify if there are multiple RX queue configured for port by nb_rx_queues using rte_eth_dev_info_get.
   - Using rte_eth_dev_stats fetch drops in q_errors, check if RX thread is configured to fetch packets from the port queue pair.
   - Using rte_eth_dev_stats shows drops in rx_nombuf, check if RX thread has enough cycles to consume the packets from the queue.
2. At TX
   - If the TX rate is falling behind the application fill rate, identify if there are enough descriptors with rte_eth_dev_info_get for TX.
   - Check the nb_pkt in rte_eth_tx_burst is done for multiple packets.
   - Check rte_eth_tx_burst invokes the vector function call for the PMD.
   - If oerrors are getting incremented, TX packet validations are failing. Check if there queue specific offload failures.
   - If the drops occur for large size packets, check MTU and multi-segment support configured for NIC.

### 11.2.3. Is there object drops in producer point for the ring library?

Producer point for ring Fig. 11.4.


*Fig. 11.4 Producer point for Rings*

1. Performance issue isolation at producer
   - Use rte_ring_dump to validate for all single producer flag is set to RING_F_SP_ENQ.
   - There should be sufficient rte_ring_free_count at any point in time.
   - Extreme stalls in dequeue stage of the pipeline will cause rte_ring_full to be true.

### 11.2.4. Is there object drops in consumer point for the ring library?

Consumer point for ring Fig. 11.5.


*Fig. 11.5 Consumer point for Rings*

1. Performance issue isolation at consumer
   - Use rte_ring_dump to validate for all single consumer flag is set to RING_F_SC_DEQ.
   - If the desired burst dequeue falls behind the actual dequeue, the enqueue stage is not filling up the ring as required.
   - Extreme stall in the enqueue will lead to rte_ring_empty to be true.

### 11.2.5. Is there a variance in packet or object processing rate in the pipeline?

Memory objects close to NUMA Fig. 11.6.

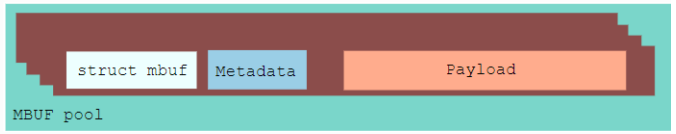



*Fig. 11.6 Memory objects have to be close to the device per NUMA.*

1. Stall in processing pipeline can be attributes of MBUF release delays. These can be narrowed down to
   - Heavy processing cycles at single or multiple processing stages.
   - Cache is spread due to the increased stages in the pipeline.
   - CPU thread responsible for TX is not able to keep up with the burst of traffic.
   - Extra cycles to linearize multi-segment buffer and software offload like checksum, TSO, and VLAN strip.
   - Packet buffer copy in fast path also results in stalls in MBUF release if not done selectively.
   - Application logic sets rte_pktmbuf_refcnt_set to higher than the desired value and frequently uses rte_pktmbuf_prefree_seg and does not release MBUF back to mempool.
2. Lower performance between the pipeline processing stages can be
   - The NUMA instance for packets or objects from NIC, mempool, and ring should be the same.
   - Drops on a specific socket are due to insufficient objects in the pool. Use rte_mempool_get_count or rte_mempool_avail_count to monitor when drops occurs.
   - Try prefetching the content in processing pipeline logic to minimize the stalls.
3. Performance issue can be due to special cases
   - Check if MBUF continuous with rte_pktmbuf_is_contiguous as certain offload requires the same.
   - Use rte_mempool_cache_create for user threads require access to mempool objects.
   - If the variance is absent for larger huge pages, then try rte_mem_lock_page on the objects, packets, lookup tables to isolate the issue.

### 11.2.6. Is there a variance in cryptodev performance?

Crypto device and PMD Fig. 11.7.

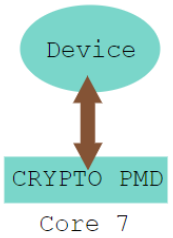



*Fig. 11.7 CRYPTO and interaction with PMD device.*

1. Performance issue isolation for enqueue
   - Ensure cryptodev, resources and enqueue is running on NUMA cores.
   - Isolate if the cause of errors for err_count using rte_cryptodev_stats.
   - Parallelize enqueue thread for varied multiple queue pair.
2. Performance issue isolation for dequeue
   - Ensure cryptodev, resources and dequeue are running on NUMA cores.
   - Isolate if the cause of errors for err_count using rte_cryptodev_stats.
   - Parallelize dequeue thread for varied multiple queue pair.
3. Performance issue isolation for crypto operation
   - If the cryptodev software-assist is in use, ensure the library is built with right (SIMD) flags or check if the queue pair using CPU ISA for feature_flags AVX|SSE|NEON using rte_cryptodev_info_get.
   - If the cryptodev hardware-assist is in use, ensure both firmware and drivers are up to date.
4. Configuration issue isolation
   - Identify cryptodev instances with rte_cryptodev_count and rte_cryptodev_info_get.

### 11.2.11. Is the packet not in the unexpected format?

Packet capture before and after processing Fig. 11.11.

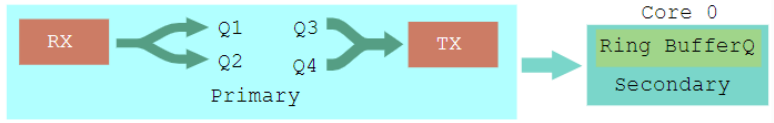



*Fig. 11.11 Capture points of Traffic at RX-TX.*

1. To isolate the possible packet corruption in the processing pipeline, carefully staged capture packets are to be implemented.
   - First, isolate at NIC entry and exit.

     Use pdump in primary to allow secondary to access port-queue pair. The packets get copied over in RX|TX callback by the secondary process using ring buffers.
   - Second, isolate at pipeline entry and exit.

     Using hooks or callbacks capture the packet middle of the pipeline stage to copy the packets, which can be shared to the secondary debug process via user-defined custom rings.

**Note**

Use similar analysis to objects and metadata corruption.

### 11.2.12. Does the issue still persist?

The issue can be further narrowed down to the following causes.

1. If there are vendor or application specific metadata, check for errors due to META data error flags. Dumping private meta-data in the objects can give insight into details for debugging.
2. If there are multi-process for either data or configuration, check for possible errors in the secondary process where the configuration fails and possible data corruption in the data plane.
3. Random drops in the RX or TX when opening other application is an indication of the effect of a noisy neighbor. Try using the cache allocation technique to minimize the effect between applications.

### 11.2.9. Is there a bottleneck in the performance of eventdev?

1. Check for generic configuration
   - Ensure the event devices created are right NUMA using rte_event_dev_count and rte_event_dev_socket_id.
   - Check for event stages if the events are looped back into the same queue.
   - If the failure is on the enqueue stage for events, check if queue depth with rte_event_dev_info_get.
2. If there are performance drops in the enqueue stage
   - Use rte_event_dev_dump to dump the eventdev information.
   - Periodically checks stats for queue and port to identify the starvation.
   - Check the in-flight events for the desired queue for enqueue and dequeue.

### 11.2.10. Is there a variance in traffic manager?

Traffic Manager on TX interface Fig. 11.10.

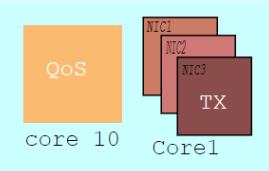



*Fig. 11.10 Traffic Manager just before TX.*

1. Identify the cause for a variance from expected behavior, is due to insufficient CPU cycles. Use rte_tm_capabilities_get to fetch features for hierarchies, WRED and priority schedulers to be offloaded hardware.
2. Undesired flow drops can be narrowed down to WRED, priority, and rates limiters.
3. Isolate the flow in which the undesired drops occur. Use rte_tn_get_number_of_leaf_node and flow table to ping down the leaf where drops occur.
4. Check the stats using rte_tm_stats_update and rte_tm_node_stats_read for drops for hierarchy, schedulers and WRED configurations.

### 11.2.8. Is the execution cycles for dynamic service functions are not frequent?

service functions on service cores Fig. 11.9.

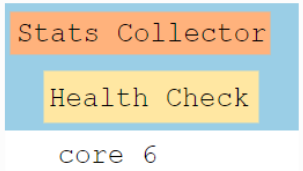



*Fig. 11.9 functions running on service cores*

1. Performance issue isolation
   - Services configured for parallel execution should have rte_service_lcore_count should be equal to rte_service_lcore_count_services.
   - A service to run parallel on all cores should return RTE_SERVICE_CAP_MT_SAFE for rte_service_probe_capability and rte_service_map_lcore_get returns unique lcore.
   - If service function execution cycles for dynamic service functions are not frequent?
   - If services share the lcore, overall execution should fit budget.
2. Configuration issue isolation
   - Check if service is running with rte_service_runstate_get.
   - Generic debug via rte_service_dump.

### 11.2.7. Is user functions performance is not as expected?

Custom worker function Fig. 11.8.

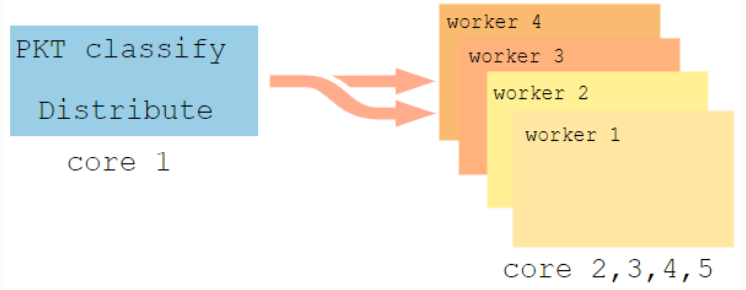



*Fig. 11.8 Custom worker function performance drops.*

1. Performance issue isolation
   - The functions running on CPU cores without context switches are the performing scenarios. Identify lcore with rte_lcore and lcore index mapping with CPU using rte_lcore_index.
   - The functions running on CPU cores without context switches are the performing scenarios. Identify lcore with rte_lcore and lcore index mapping with CPU using rte_lcore_index.
   - Use rte_thread_get_affinity to isolate functions running on the same CPU core.
2. Configuration issue isolation
   - Identify core role using rte_eal_lcore_role to identify RTE, OFF and SERVICE. Check performance functions are mapped to run on the cores.
   - For high-performance execution logic ensure running it on correct NUMA and non-master core.
   - Analyze run logic with rte_dump_stack, rte_dump_registers and rte_memdump for more insights.
   - Make use of 'objdump' to ensure opcode is matching to the desired state.