DPDK

DPDK's Best Kept Secret
Micro-benchmarks

M Jay

Muthurajan.Jayakumar@intel.com

DPDK Summit - San Jose 2017

Legal Information

DPDK

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User an
Reference Guides for more information regarding the specific instruction sets covered by this notice. Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does
not guarantee any costs or cost reduction. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or servic
activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or
learn more at https://networkbuilders.intel.com/network-technologies/intelselectfasttrackkit.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact
your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

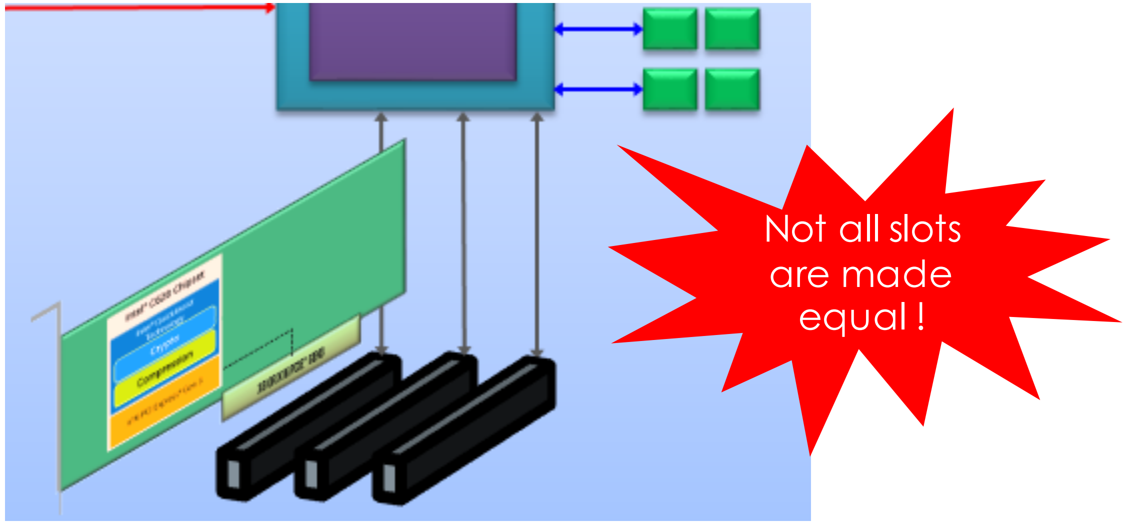
The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Agenda

- Why should I care about DPDK Micro-benchmarks?
- What do they benchmark?
- How do I run them?

Not all slots are made equal

Ensure that you have plugged in your NIC card in most optimal slot



Not all slots are made equal !

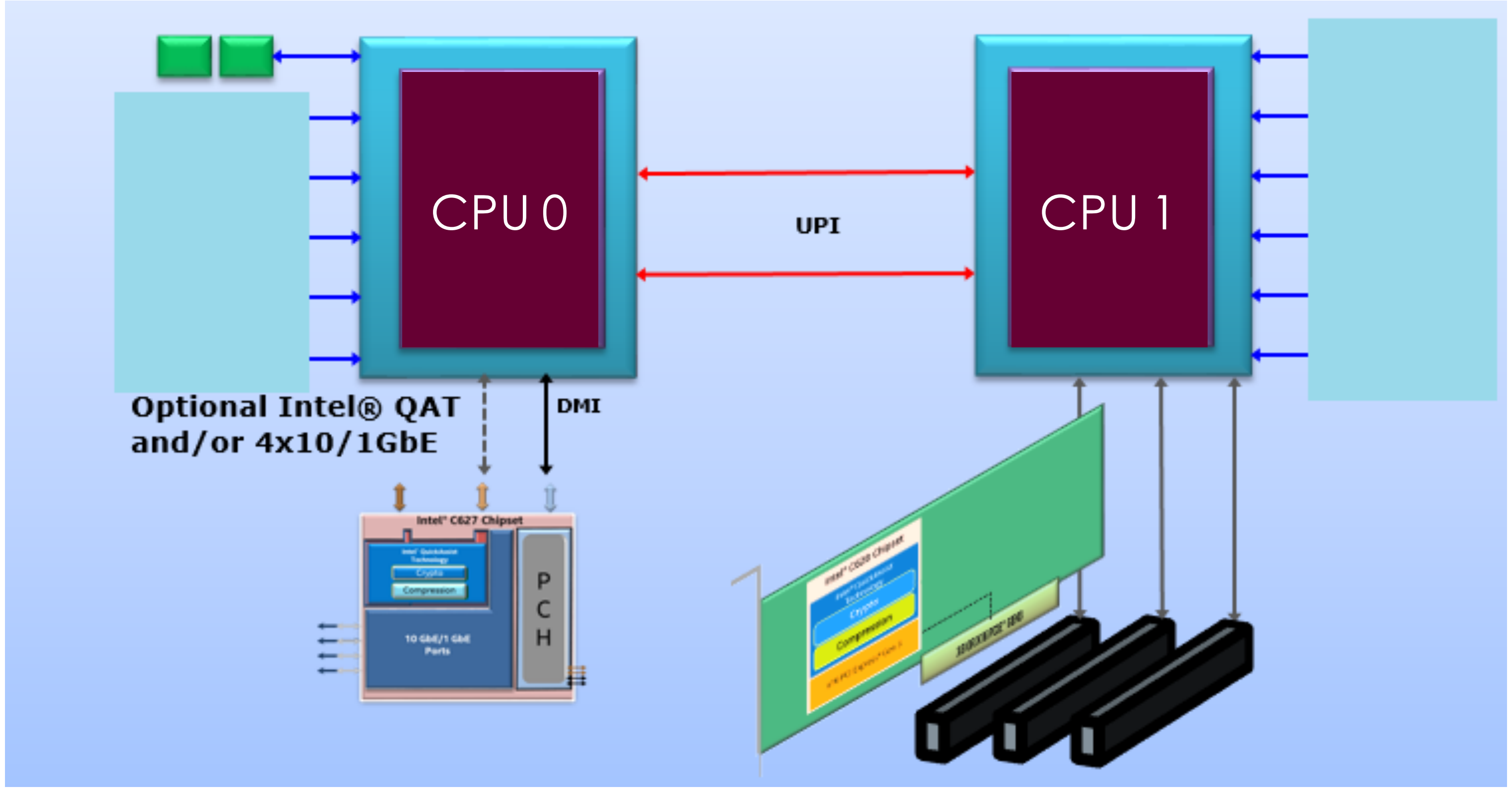
How many lcores, you think, are there in this 2 socket server?

64 lcores?

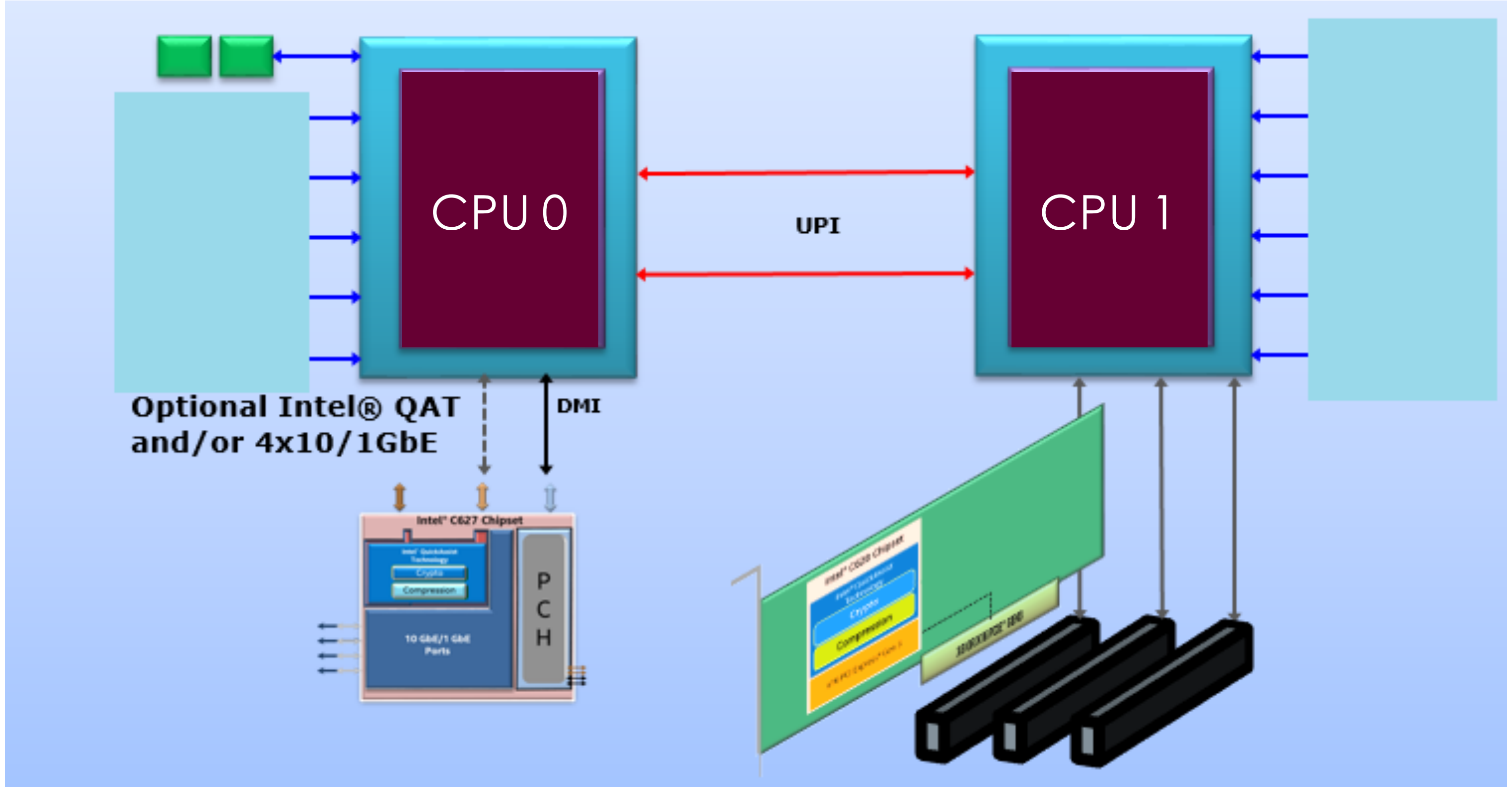
96 lcores?

More than 100 lcores?

Question: What can be Improved here?



Improvements -n 4



I/O Plugged in CPU1's Slot
How much memory do you see in CPU1 node?
ZERO !

CPU 0 has only One Channel memory populated.

In Which Socket lcore # 50 resides? Socket 0 or Socket 1?

More than 100 lcores

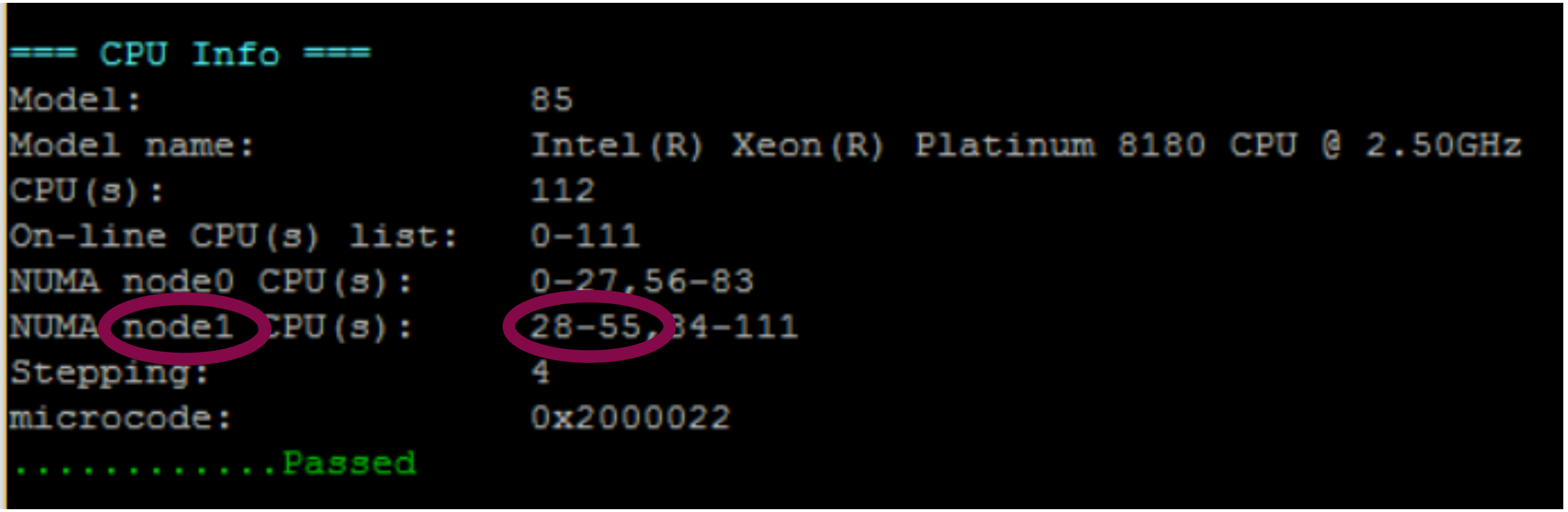
- **Question:**
- In which socket you think lcore# 50 resides? – socket 0? Or socket 1?

Socket 0?

Socket 1?

- Assume NIC is Plugged in socket 0
- Will the performance be best or sub-optimal?

Why Should I Care About DPDK Micro-benchmarks?

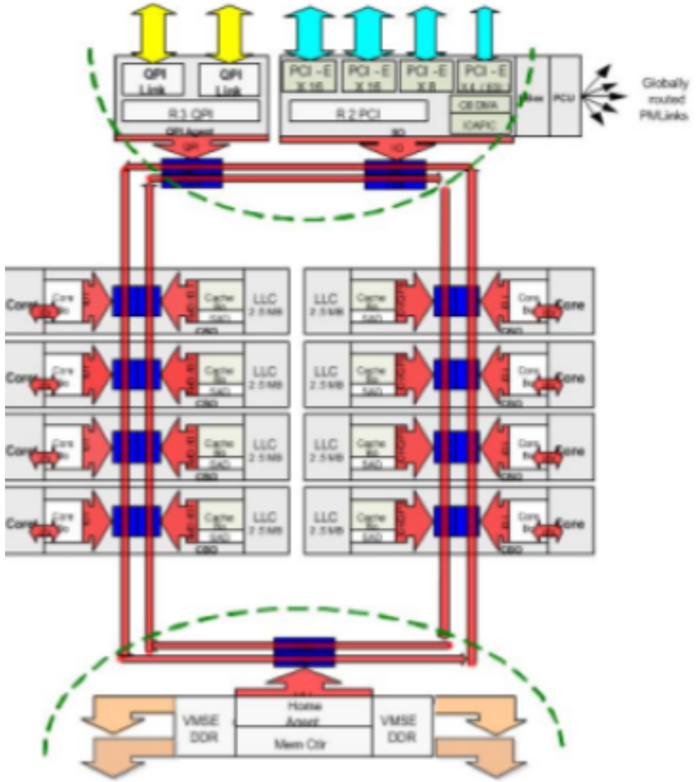


```
=== CPU Info ===
Model:                  85
Model name:             Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz
CPU(s):                 112
On-line CPU(s) list:    0-111
NUMA node0 CPU(s):      0-27,56-83
NUMA node1 CPU(s):      28-55,84-111
Stepping:               4
microcode:              0x2000022
............Passed
```

- We thought lcore # 50 resides in socket 0.
- But actually, you can see it is in socket 1.
- So, NIC in socket 0 is actually sub-optimal.
- How to quantitatively ensure that system is set for optimal performance?

QUIZ: Cores Within A Socket – All In Same Loop?

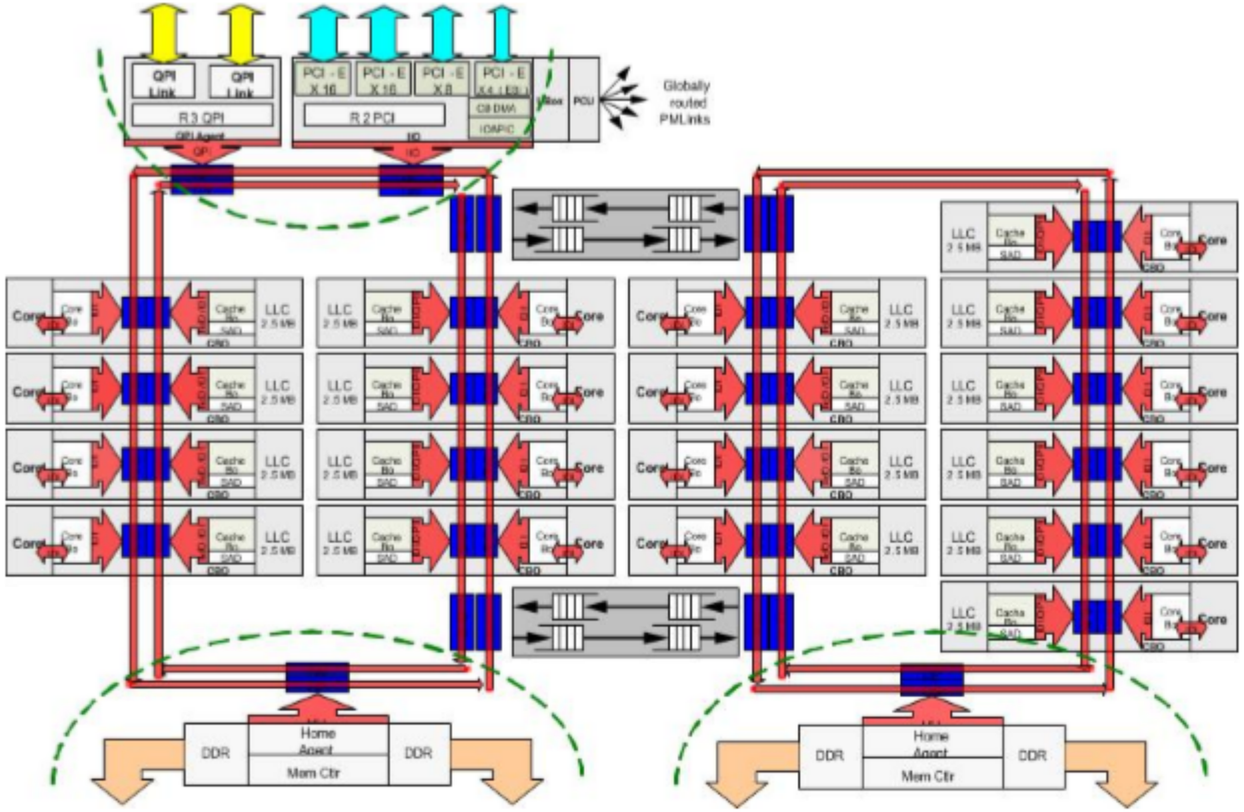
4-8 Core (LCC)



Demo

Cores Within A Socket – Not equal proximity

14-18 Core (HCC)

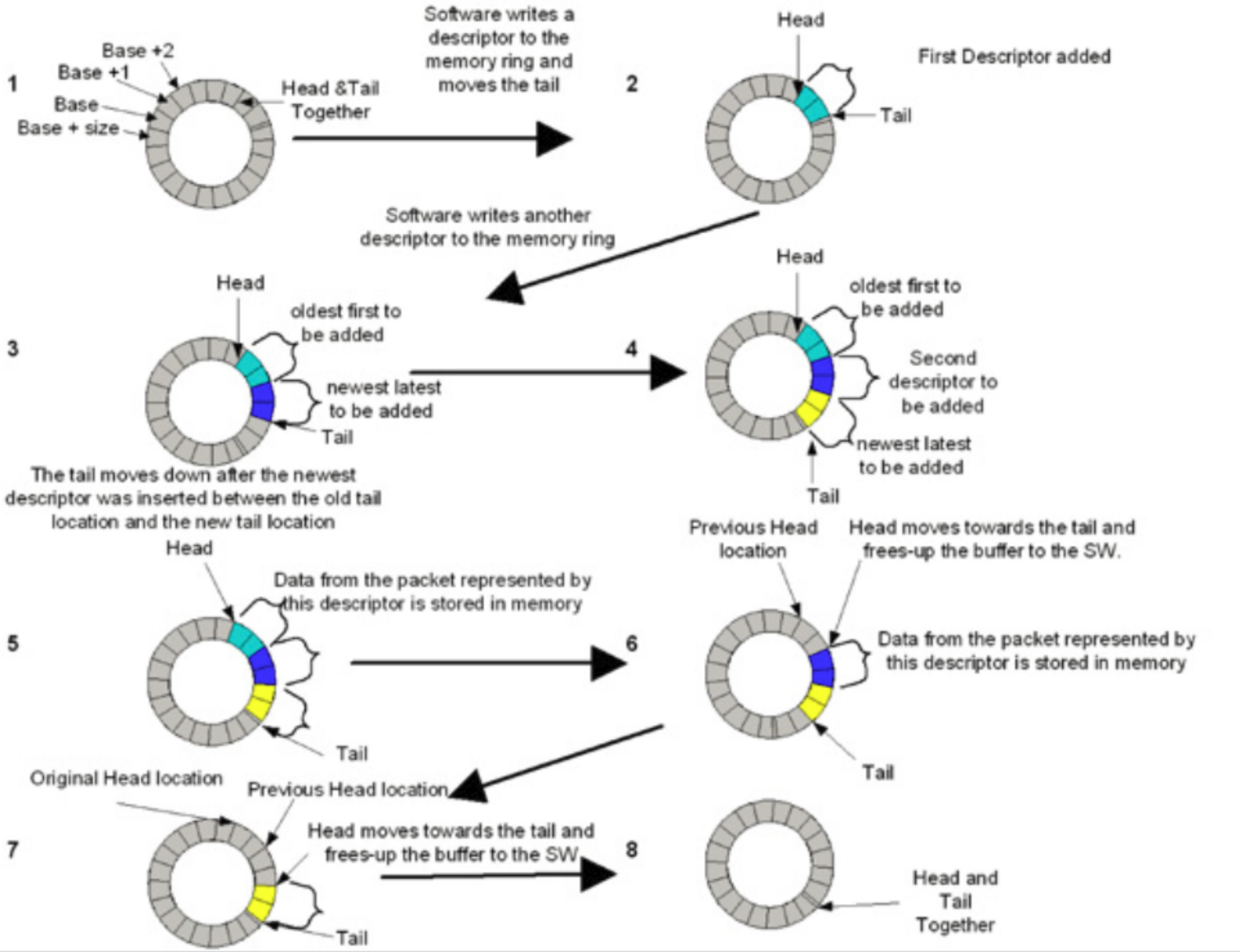


Prior to application level benchmarking..

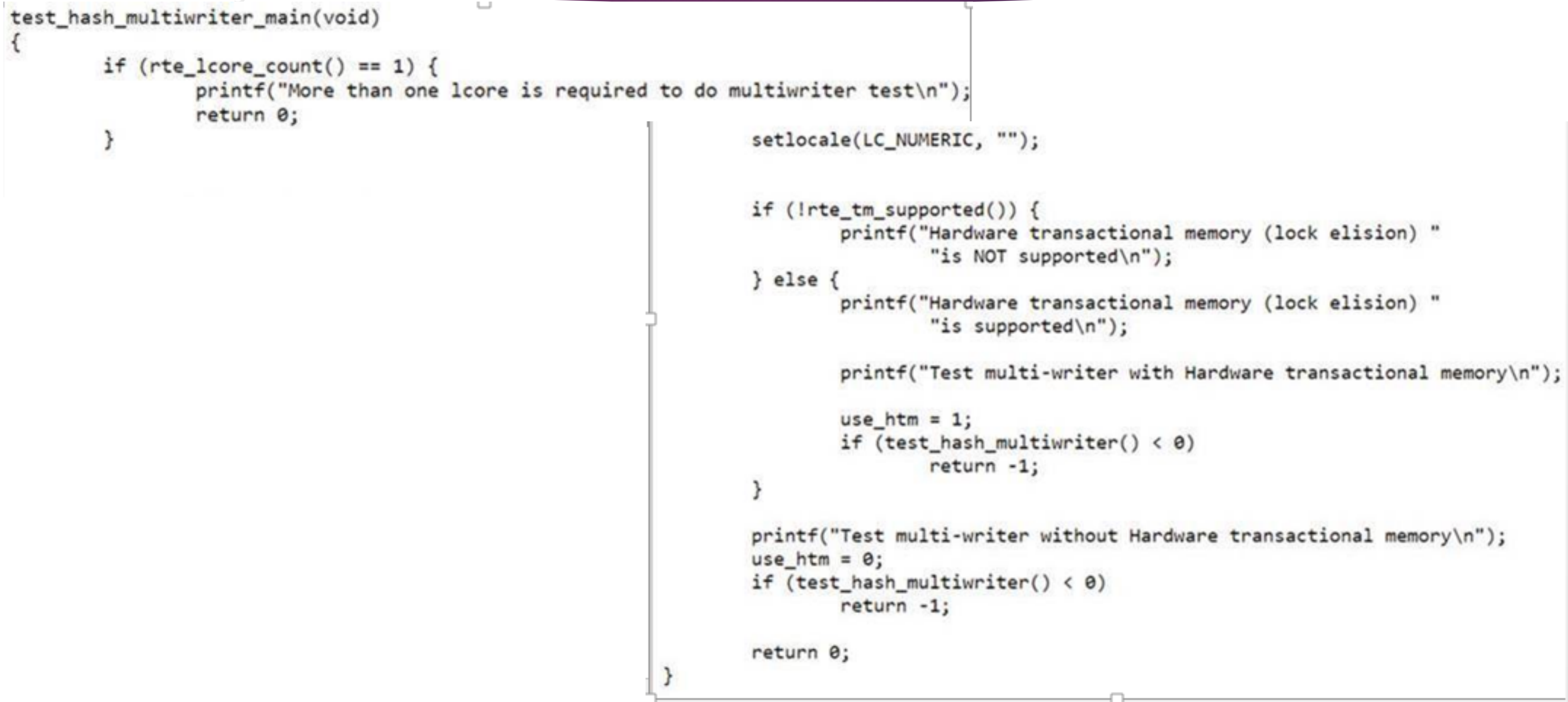
- Without tightening these, if you start developing your application…
- And on top of that, if you start measuring application level performance
- Root cause analysis is made unnecessarily complex
-
- Instead… what if ..
- What if you can do basic benchmarking of key performant elements / ops
- You will build strong foundation first
- Will help you develop Applications confidently towards overall higher performance

What Objects, What Operations to benchmark?

- In other words, what are the key high performant **objects** and **operations**?
- **Objects:**
  - Ring
  - Mem pool
  - Mbuf
- **Operations:**
  - Mem copy
  - Hash Operations
  - Flow Classification


1
Base +2
Base +1
Base
Base + size
Head &Tail Together
Software writes a descriptor to the memory ring and moves the tail
2
Head
First Descriptor added
Tail
Software writes another descriptor to the memory ring
3
Head
oldest first to be added
newest latest to be added
Tail
The tail moves down after the newest descriptor was inserted between the old tail location and the new tail location
4
Head
oldest first to be added
Second descriptor to be added
newest latest to be added
Tail
5
Head
Data from the packet represented by this descriptor is stored in memory
Tail
6
Previous Head location
Head moves towards the tail and frees-up the buffer to the SW.
Data from the packet represented by this descriptor is stored in memory
Tail
7
Original Head location
Previous Head location
Head moves towards the tail and frees-up the buffer to the SW
Tail
8
Head and Tail Together

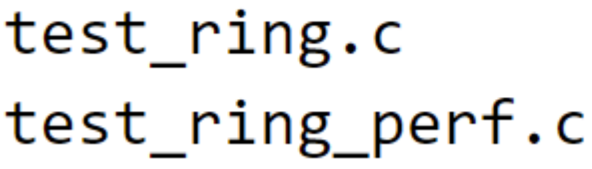
Test_hash_multiwriter_main( )
Hash – Multi-writer – Transactional Memory



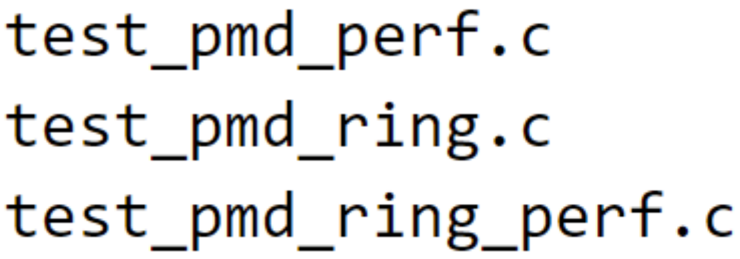
```
test_hash_multiwriter_main(void)
{
        if (rte_lcore_count() == 1) {
                printf("More than one lcore is required to do multiwriter test\n");
                return 0;
        }
```

```
        setlocale(LC_NUMERIC, "");


        if (!rte_tm_supported()) {
                printf("Hardware transactional memory (lock elision) "
                        "is NOT supported\n");
        } else {
                printf("Hardware transactional memory (lock elision) "
                        "is supported\n");

                printf("Test multi-writer with Hardware transactional memory\n");

                use_htm = 1;
                if (test_hash_multiwriter() < 0)
                        return -1;
        }

        printf("Test multi-writer without Hardware transactional memory\n");
        use_htm = 0;
        if (test_hash_multiwriter() < 0)
                return -1;

        return 0;
}
```

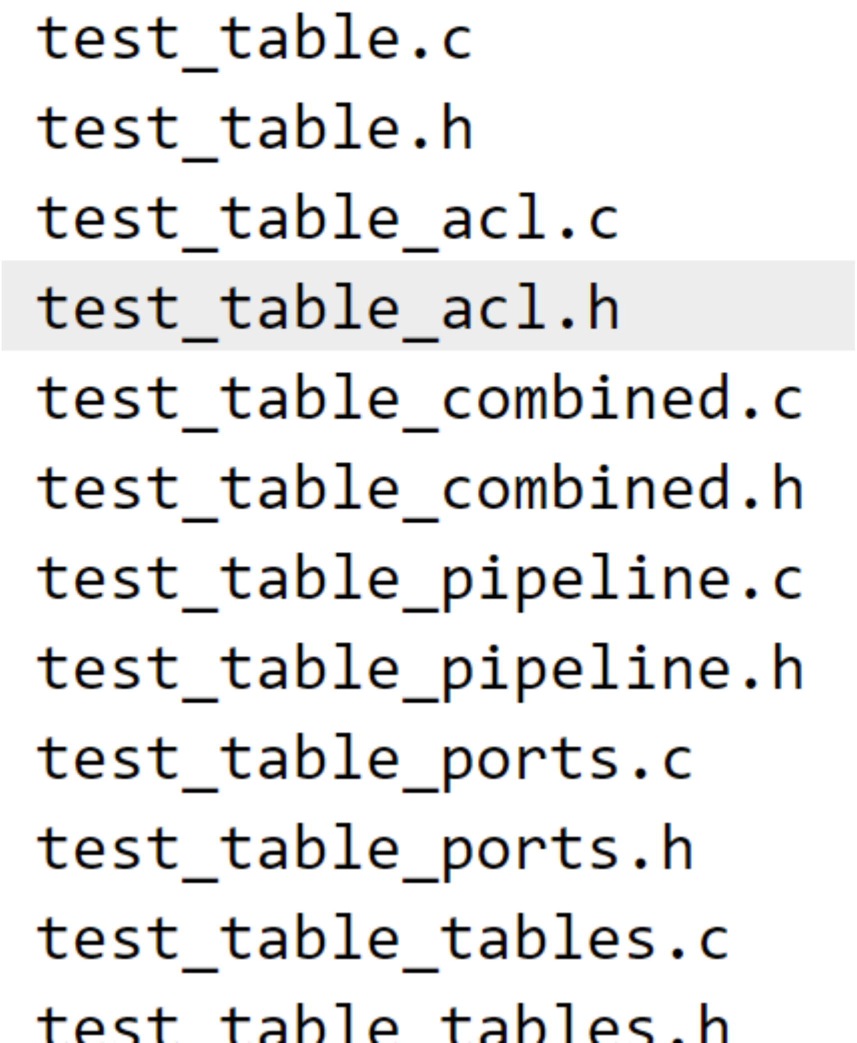
Tests: Ring, PMD, Table



```
test_ring.c
test_ring_perf.c
```



```
test_pmd_perf.c
test_pmd_ring.c
test_pmd_ring_perf.c
```



```
test_table.c
test_table.h
test_table_acl.c
test_table_acl.h
test_table_combined.c
test_table_combined.h
test_table_pipeline.c
test_table_pipeline.h
test_table_ports.c
test_table_ports.h
test_table_tables.c
test_table_tables.h
```

Router, Memcpy, Hash

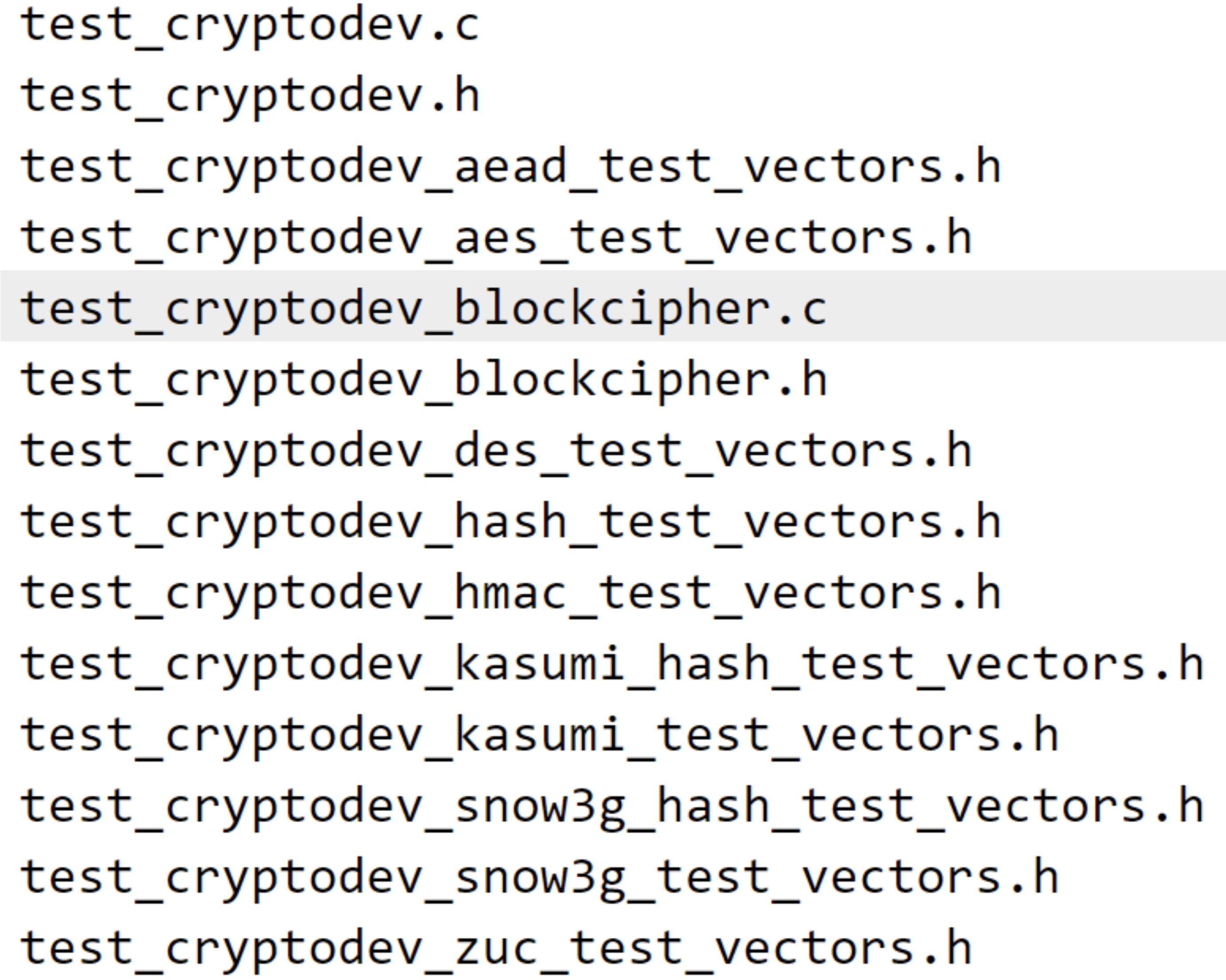
DPDK

```
test_lpm.c
test_lpm6.c
test_lpm6_data.h
test_lpm6_perf.c
test_lpm_perf.c
```

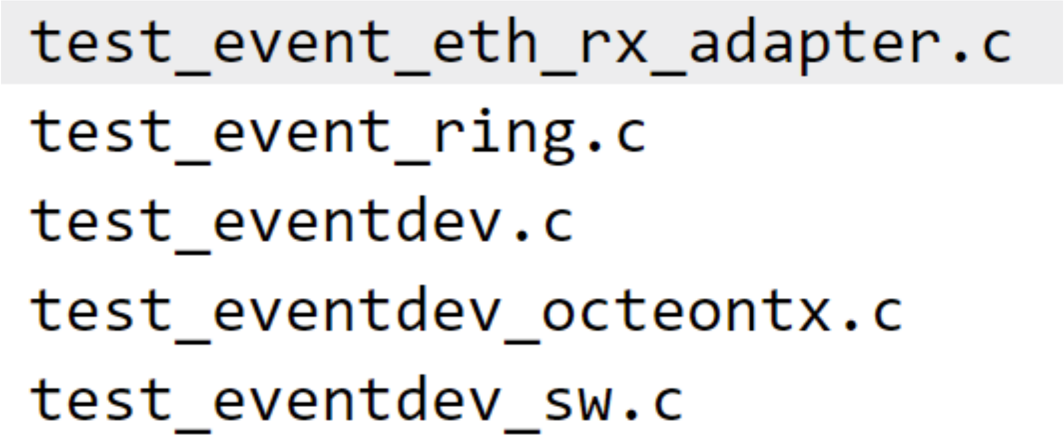
```
test_malloc.c
test_mbuf.c
test_member.c
test_member_perf.c
test_memcpy.c
test_memcpy_perf.c
test_memory.c
test_mempool.c
test_mempool_perf.c
test_memzone.c
```

```
test_hash.c
test_hash_functions.c
test_hash_multiwriter.c
test_hash_perf.c
test_hash_scaling.c
```

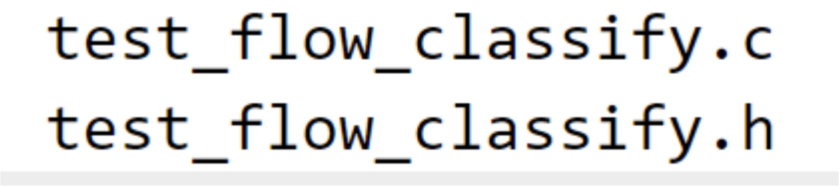
Tests: Crypto, Event, Flow Classify



test_cryptodev.c
test_cryptodev.h
test_cryptodev_aead_test_vectors.h
test_cryptodev_aes_test_vectors.h
test_cryptodev_blockcipher.c
test_cryptodev_blockcipher.h
test_cryptodev_des_test_vectors.h
test_cryptodev_hash_test_vectors.h
test_cryptodev_hmac_test_vectors.h
test_cryptodev_kasumi_hash_test_vectors.h
test_cryptodev_kasumi_test_vectors.h
test_cryptodev_snow3g_hash_test_vectors.h
test_cryptodev_snow3g_test_vectors.h
test_cryptodev_zuc_test_vectors.h

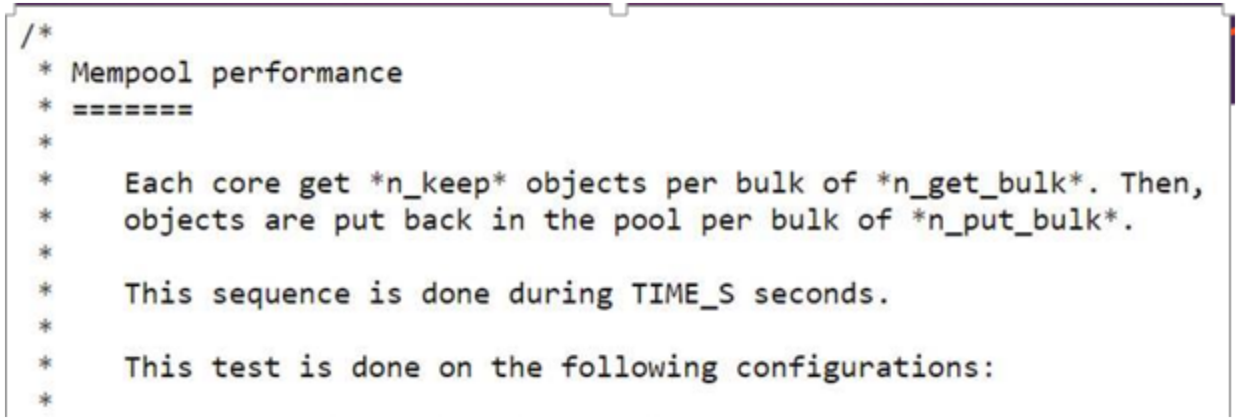


test_event_eth_rx_adapter.c
test_event_ring.c
test_eventdev.c
test_eventdev_octeontx.c
test_eventdev_sw.c

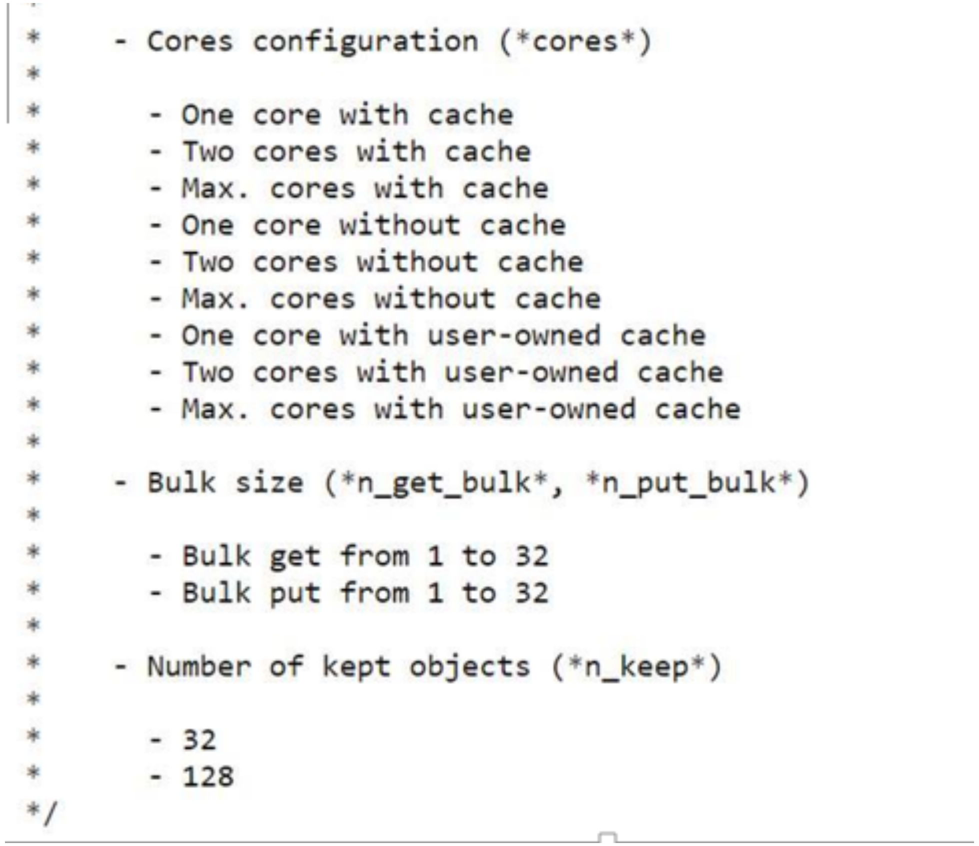


test_flow_classify.c
test_flow_classify.h

Mempool

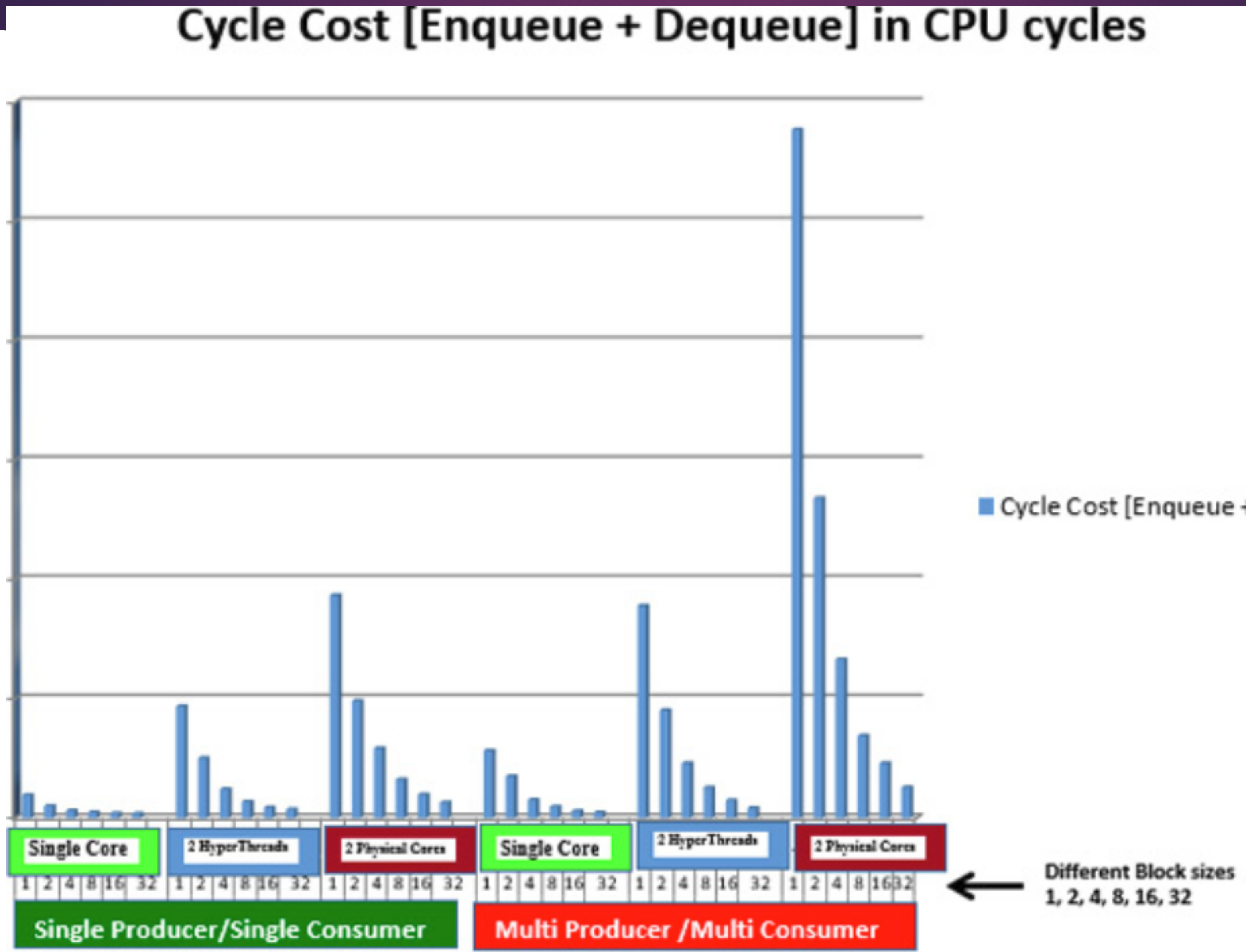


```
/*
 * Mempool performance
 * =======
 *
 *    Each core get *n_keep* objects per bulk of *n_get_bulk*. Then,
 *    objects are put back in the pool per bulk of *n_put_bulk*.
 *
 *    This sequence is done during TIME_S seconds.
 *
 *    This test is done on the following configurations:
 *
```



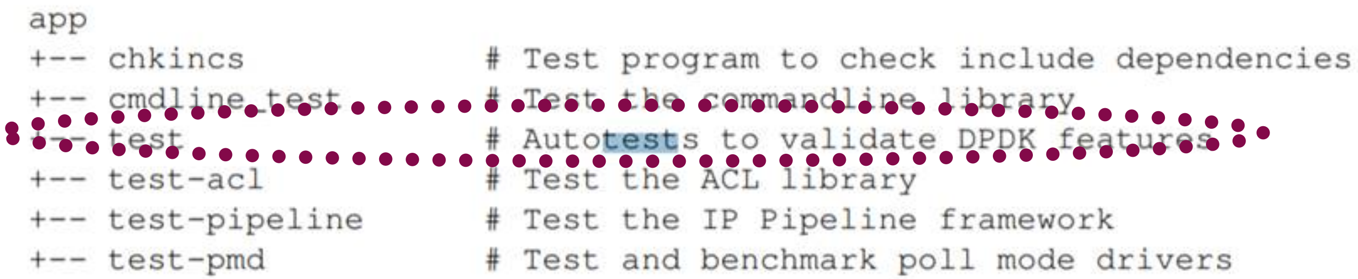
```
 *    - Cores configuration (*cores*)
 *
 *      - One core with cache
 *      - Two cores with cache
 *      - Max. cores with cache
 *      - One core without cache
 *      - Two cores without cache
 *      - Max. cores without cache
 *      - One core with user-owned cache
 *      - Two cores with user-owned cache
 *      - Max. cores with user-owned cache
 *
 *    - Bulk size (*n_get_bulk*, *n_put_bulk*)
 *
 *      - Bulk get from 1 to 32
 *      - Bulk put from 1 to 32
 *
 *    - Number of kept objects (*n_keep*)
 *
 *      - 32
 *      - 128
 */
```

SPSC MPMC – Time Taken

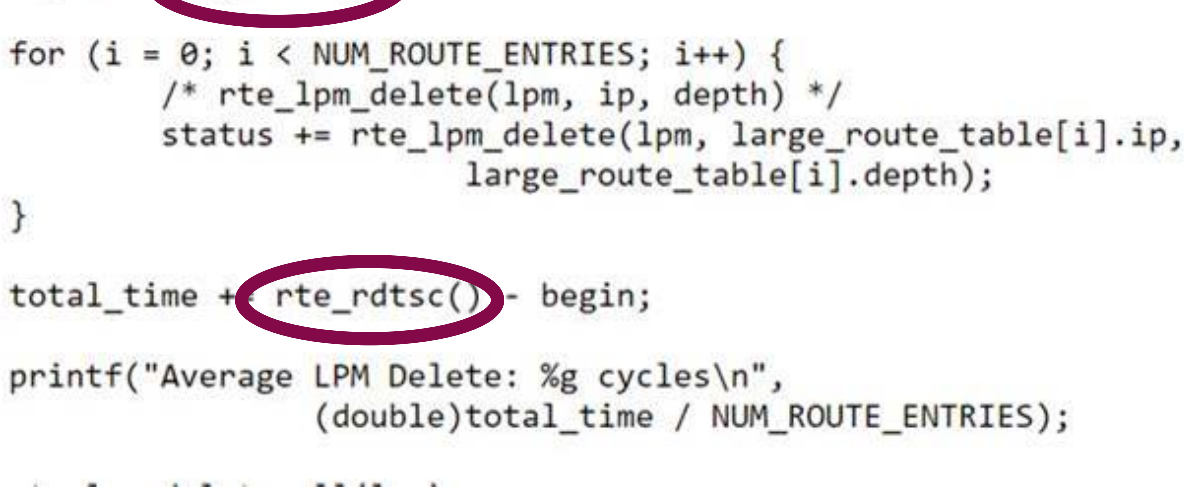


Call To Action: Where To Find Them & How It Measures?

The app directory contains sample applications that are used to test DPDK (such as autotests) or the Poll Mode Drivers (test-pmd):



```
app
+-- chkincs             # Test program to check include dependencies
+-- cmdline_test        # Test the commandline library
+-- test                # Autotests to validate DPDK features
+-- test-acl            # Test the ACL library
+-- test-pipeline       # Test the IP Pipeline framework
+-- test-pmd            # Test and benchmark poll mode drivers
```



```
/* Delete */
status = 0;
begin = rte_rdtsc();

for (i = 0; i < NUM_ROUTE_ENTRIES; i++) {
        /* rte_lpm_delete(lpm, ip, depth) */
        status += rte_lpm_delete(lpm, large_route_table[i].ip,
                        large_route_table[i].depth);
}

total_time += rte_rdtsc() - begin;

printf("Average LPM Delete: %g cycles\n",
                (double)total_time / NUM_ROUTE_ENTRIES);
```

Optimization Notice

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2®, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Questions?

M Jay

Muthurajan.Jayakumar@intel.com