DPDK
DATA PLANE DEVELOPMENT KIT

DPDK Summit

SAMPLE VNF in OPNFV

KANNAN BABU RAMIA, INTEL
ANAND B JYOTI, INTEL

LEGAL DISCLAIMER

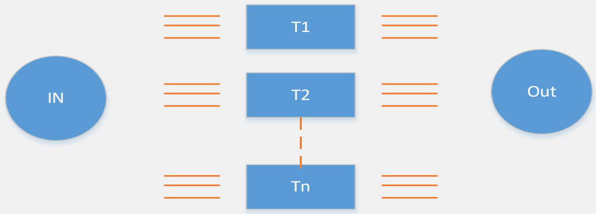
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, 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: http://www.intel.com/design/literature.htm
- Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others.
-
- 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
- Mileage may vary Disclaimer: Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **www.intel.com/benchmarks** Test and System Configurations: Estimates are based on internal Intel analysis using at least Data Plane Development Kit IpSec sample application on Intel(R) Xeon(R) CPU E5-2658 v4@ 2.30GHz with atleast using Intel(R) Communications Chipset(s) 8955 with Intel(R) QuickAssist Technology.

Agenda

- Packet Processing Concepts – Brief intro
- Best Known Methods for writing performance tuned application over dpdk
- Sample vnf in OPNFV
- Example code snippets
- Open Questions

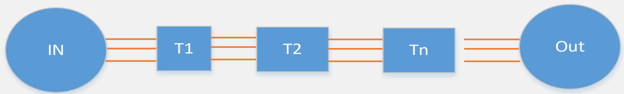
Packet Processing Concepts

Pool of Threads (Run To Completion)



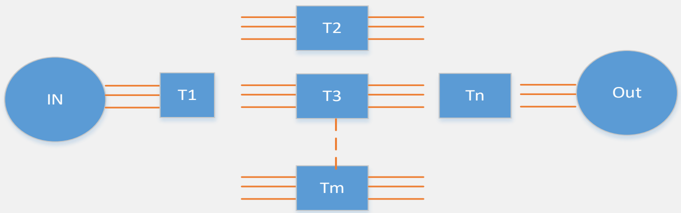
{T1...Tn} = F = { f1, f2, f3, f4...fn} AND (Ca * n) >= C_F

Context Pipeline



T1 + T2 +T3...+Tn = F1+F2+F3...+Fn & F={f1,f2,f3..fn} AND Ca >= C_{F1} or C_{F2}..C_{Fn}

Hybrid Pipeline



T1+T2...Tm+Tn=F1+F2+F3 & F2={f1,f2...f3} AND Ca >= C_{F1} or C_{F2}..C_{Fn}

Pros	Cons
Perfect for scaling	Need an efficient load balancer for distribution. Synchronization overheads (locks-reorder or flow affinity/atomicity)
Tolerate changes in the features and variants in packet processing	Statefull/Asynchronous processing stage might create imbalance in distribution
Easy in portability across platforms	Must have HW packet acceleration

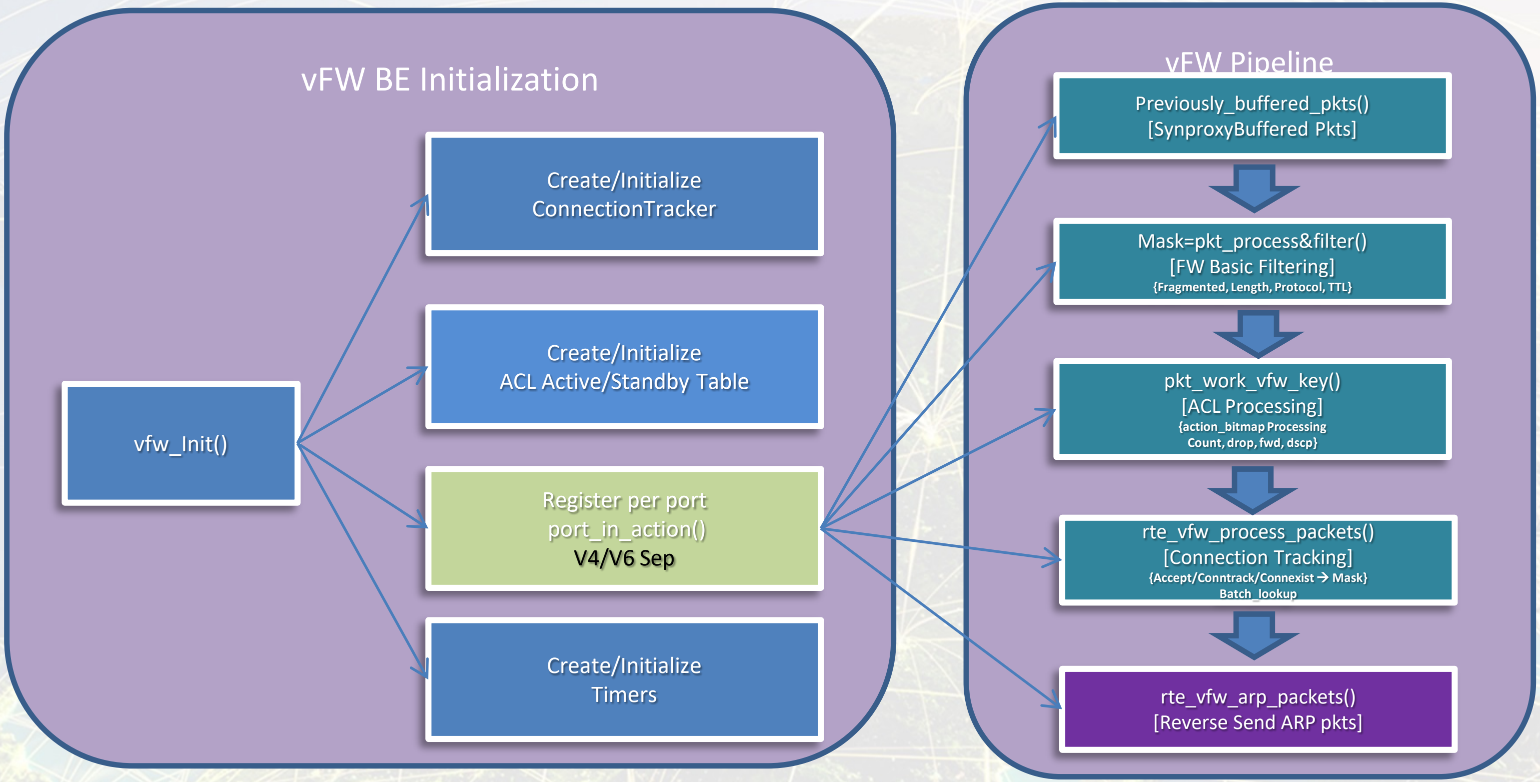
Pros	Cons
Perfect for handling statefull/asynchronous processing	Cant tolerate changes in features. Requires replanning of function partitioning
No dependency on HW packet acceleration	Performance limited by per core processing capacity
Suitable for high performance cores	Not easily portable across platforms

Pros	Cons
Mix of RTC and Context Pipeline	Relies on high performance core

BKMs for packet processing

1. Avoiding serialization in the packet-processing pipeline, including serializing events such as locks, special instructions such as CLFLUSH, and large critical sections
2. Accessing data from the cache where possible by making use of prefetch instructions and observing best practices in design of the software pipeline
3. Designing data structures to be cache-aligned and avoiding occurrences of data being spread across two cache lines. Avoid partial writes and contention between write/read operations.
4. Maintaining affinity between software threads and hardware threads. Isolating software threads from one another with regards to scheduling relative to hardware threads.
5. Breaking down data-plane functionality so that it can be implemented with a combination of RTC (Run to Completion) and pipeline methods
6. Use of pre-tuned open source optimized software components like DPDK libraries
7. Software pipelining, the concept is achieved by processing burst/bunch of packets and constructing multiple stages to hide any latencies experienced by the processing stages.
8. Also minimize the DTLB and ITLB misses and cache Ping-Pong effects.

vFW Processing Flow Diagram



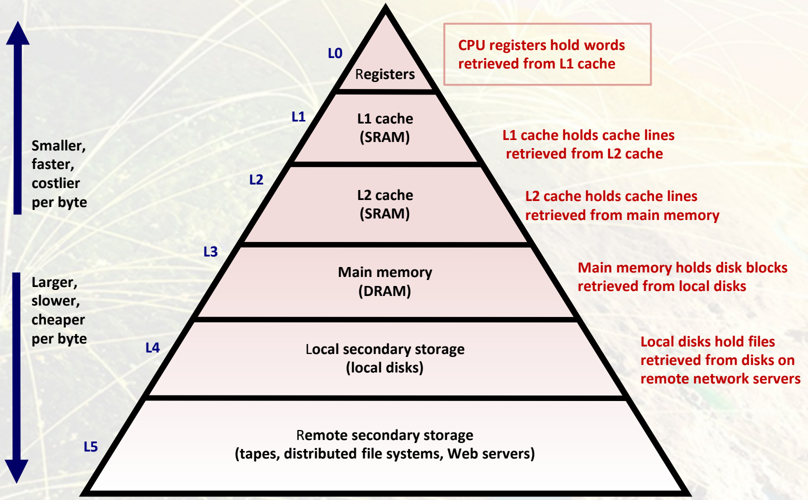
BKM#1

Avoiding serialization in the packet-processing pipeline
including serializing events such as locks, special instructions such as CLFLUSH, and large critical sections

- Multiple pipelines can run on separate cores
  - Packets being load shared across multiple pipeline for better latency and throughput
- ACL active/standby tables updated by CLI and single thread
  - Updates standby table and switches – Avoids locks
- Connection Tracker status structure
  - CT created per pipeline accessed by only one process(WT) – No Locks
  - Both ingress/egress traffic is handled by same thread. Ensures CT is accessed by single process.
- SWLB (SWLB tuple based load distribution)- TxRx Pipeline used (HW independent)
  - NIC → RXQ →SWLB → SWQs →VNF WT →SWQs →TXQ →NIC
  - Pros: Independent of NIC HW capability.
  - Cons: Load balancing to be done by a LB - More compute power
- HWLB (Set the filters in offload features of NIC)
  - NIC → RXQ →VNF WT → TXQ →NIC
  - Pros: Reduces the SWLB, Low latency and reduces a processing load due to SWLB
  - Cons: Only supported HW like Fortville NIC can be used

Example Cache Size and Latencies

- Intel i7-4770 (Haswell), 3.4 GHz (Turbo Boost off). DRAM 32 GB (PC3-12800 cl11 cr2).
- Cache Memory sizes
  - L1 Data cache = 32 KB, 64 B/line, 8-WAY. → per core → 512 cache lines
  - L1 Instruction cache = 32 KB, 64 B/line, 8-WAY. → per core
  - L2 cache = 256 KB, 64 B/line, 8-WAY → Unified Instruction/data per core
  - L3 cache = 8 MB, 64 B/line → Unified Instruction/Data per CPU
- Cache latencies
  - L1 Data Cache Latency = 4 cycles for simple access via pointer
  - L2 Cache Latency = 12 cycles
  - L3 Cache Latency = 36 cycles (3.4 GHz i7-4770)
  - L3 Cache Latency = 43 cycles (1.6 GHz E5-2603 v3)
  - L3 Cache Latency = 58 cycles (core9) - 66 cycles (core5) (3.6 GHz E5-2699 v3 - 18 cores)
- RAM Access Latencies (LLC miss latency)
  - RAM Latency = 36 cycles + 57 ns (3.4 GHz i7-4770) → 36+194 = 230 Cycles = 67.64ns
  - RAM Latency = 62 cycles + 100 ns (3.6 GHz E5-2699 dual) → 62+300 = 362 cycles = 100.5ns



http//www.7-cpu.com/cpu/Haswell.html

BKM#2 Accessing data from the cache

[make use of pre-fetch instructions and observing best practices in design of the software pipeline]

- Pipelining and pre-fetch in vFW
  - Pre-fetch the packets and associated data for processing to avoid cache miss latency
  - Burst packet handling is supported in all functions
    - May not be able to accommodate 32 packets(mbuf/ip/tcp) headers and associated data in the L1(32KB/core)/L2(256kB/core)/L3(56MB) cache leads to LLC cache miss
    - Batch process (4 packets at a time), while prefetching the packets and associated data for next batch
  - ARP processing – Needed batch processing and pre-fetching

- BPF, ACL and ConnTrack – Just prefetching helped to improve the performance
  - BPF operates on header information without any database
  - ACL – Uses DPDK optimized library with burst process handling
  - ConnTrack → Accesses only TCP/UDP headers → IPv4/IPv6 separation was not needed

- DPDK optimized functionalities
  - Cukkoo hash with bulk lookup is used from DPDK optimized libraries for CT
  - Timers are used from DPDK

Code snippets walk through

Basic Packet Filtering, ACL, CT

```
/* BPF & Counters*/
  rte_prefetch0(& vfw_pipe->counters);

   /* Pre-fetch all rte_mbuf header */
  for(j = 0; j < n_pkts; j++)
          rte_prefetch0(pkts[j]);

  memset(&ct_helper, 0, sizeof(struct rte_CT_helper));
  rte_prefetch0(& vfw_pipe->counters->pkts_drop_ttl);
  rte_prefetch0(& vfw_pipe->counters->entry_timestamp);

vfw_handle_buffered_packets()
rte_vfw_ipv4_packet_filter_and_process()

/*  ACL and CT*/
rte_prefetch0((void*)vfw_pipe->plib_acl);
rte_prefetch0((void*)vfw_rule_table_ipv4_active);
lib_acl_ipv4_pkt_work_key()
rte_ct_cnxn_tracker_batch_lookup_type()
```

ARP post processing

```
/* ARP processing */
 start_tsc_measure(vfw_pipe);
for(j = 0; j < (n_pkts & 0x3LLU); j++) {
          rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j], META_DATA_OFFSET));
          rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j], ETHERNET_START));
   }
   rte_prefetch0((void*)in_port_dir_a);
   rte_prefetch0((void*)prv_to_pub_map);
   uint8_t i;
   for (i = 0; i < (n_pkts & (~0x3LLU)); i += 4) {
          for (j = i+4; ((j < n_pkts) && (j < i+8)); j++) {
                 rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j],
META_DATA_OFFSET));
                    rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j],
ETHERNET_START));
          }
          pkt4_work_vfw_arp_ipv4_packets(&pkts[i], i, &keep_mask, synproxy_reply_mask,
vfw_pipe);
   }

   for (j = i; j < n_pkts; j++) {
          rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j], META_DATA_OFFSET));
          rte_prefetch0(RTE_MBUF_METADATA_UINT32_PTR(pkts[j], ETHERNET_START));
   }
   for (; i < n_pkts; i++) {
          pkt_work_vfw_arp_ipv4_packets(pkts[i], i, &keep_mask, synproxy_reply_mask,
vfw_pipe);
   }
 end_tsc_measure(vfw_pipe, n_pkts);
```

BKM#3 Designing data structures to be cache-aligned

(Avoid occurrences of data being spread across two cache lines, partial writes and contention between write and read operations.)

- __rte_cache_aligned compiler prefix is used for cache alignment for all required structures
- structure members 64byte aligned to avoid partial writes/contentions between Rd/Wr
- Missing alignments and re-arrange the members → Avoid cache ping/pong

Original		Optimized
struct rte_VFW_counter_block {		struct rte_VFW_counter_block {
32	char name[PIPELINE_NAME_SIZE];	/* in_port_action */
8	uint64_t pkts_received;	char name[PIPELINE_NAME_SIZE];
8	uint64_t bytes_processed	uint64_t pkts_received;
8	uint64_t internal_time_sum;	uint64_t bytes_processed;
8	uint64_t external_time_sum;	uint64_t num_batch_pkts_sum;
		uint32_t num_pkts_measurements;
8	uint64_t num_batch_pkts_sum;	**uint32_t unused_counter;**
4	uint32_t time_measurements;	/* Profiling */
4	uint32_t num_pkts_measurements;	uint64_t internal_time_sum;
4	uint32_t unused_counter;	uint64_t external_time_sum;
4 byte → HOLE		uint64_t entry_timestamp;
8	**uint64_t pkts_drop_without_rule;**	uint64_t exit_timestamp;
		uint32_t time_measurements;
	uint64_t pkts_drop_ttl;	/* ACL */
	uint64_t pkts_drop_bad_size;	uint32_t count_latencies;
	uint64_t pkts_drop_fragmented;	uint64_t sum_latencies;
	uint64_t pkts_drop_without_arp_entry;	uint64_t pkts_drop_without_rule;
	uint64_t pkts_drop_unsupported_type;	uint64_t pkts_acl_forwarded;
	struct rte_CT_counter_block *ct_counters;	/* BPF & ARP */
	uint64_t sum_latencies;	uint64_t pkts_drop_ttl;
	uint32_t count_latencies;	uint64_t pkts_drop_bad_size;
		uint64_t pkts_drop_fragmented;
	uint64_t pkts_fw_forwarded;	uint64_t pkts_drop_without_arp_entry;
	uint64_t pkts_acl_forwarded;	uint64_t pkts_drop_unsupported_type;
} __rte_cache_aligned;		uint64_t pkts_fw_forwarded;
		struct rte_CT_counter_block *ct_counters;
		} __rte_cache_aligned;

OPEN QUESTIONS?

- Exchange views
- Pickup any features in the sample vnf for development
- Attend tomorrows hand-on session

THANK YOU