DPDK

optimal performance everywhere

Rallying with a Formula 1

Thomas Monjalon – 6WIND

DPDK Summit Userspace – Dublin – 2016

Summary

- DPDK is about
  performance with various architectures/devices/environments
  thanks to optimizations/offloads and simplicity

- call to participation to fill the gaps

- more details?

Lightning Speed

1/ high throughput

- main priority

2/ low latency?

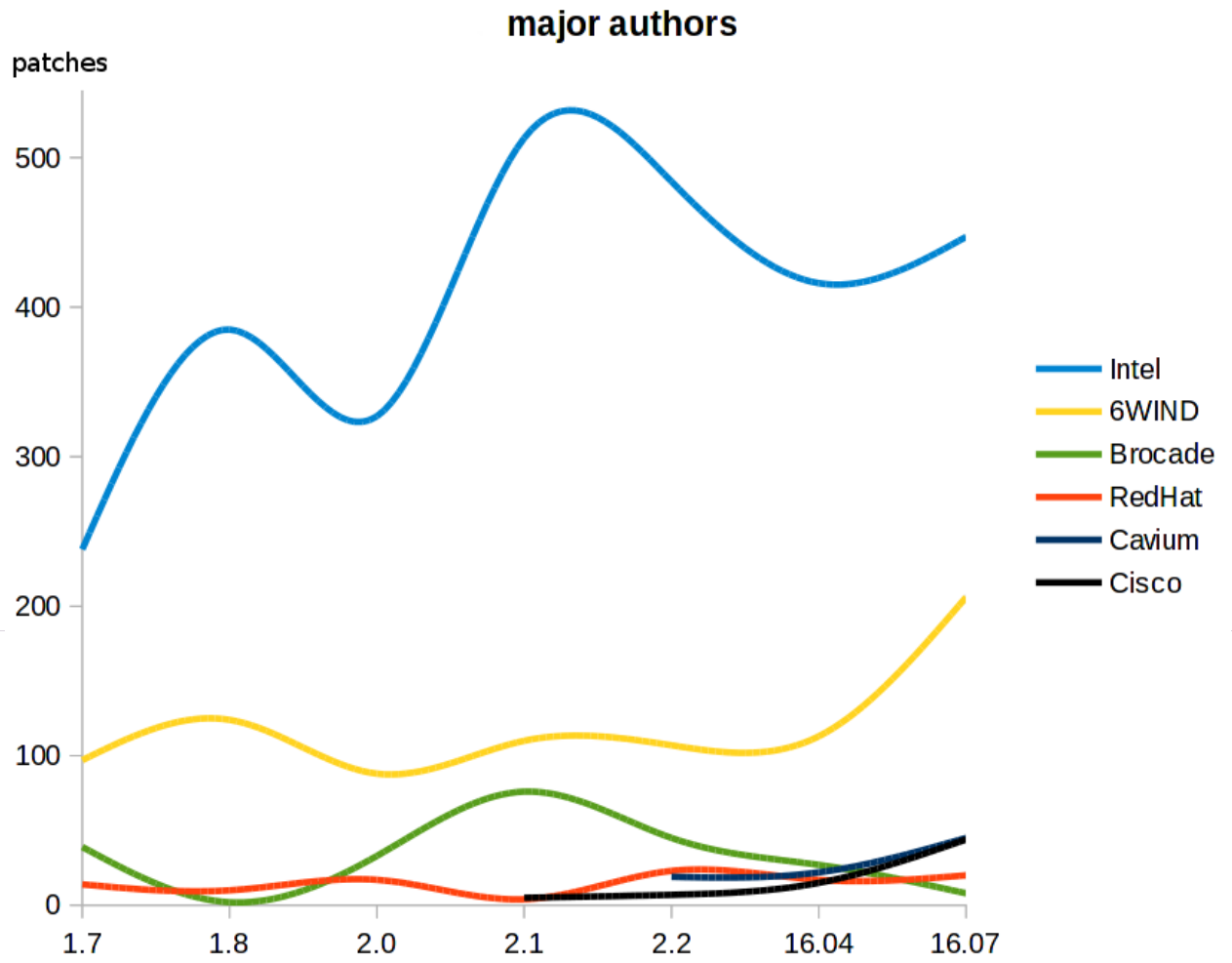
- may be studied

3/ no guarantee on low jitter

- real-time system?

Run on several Architectures

- DPDK is not Intel®
  - (not anymore ~~Intel® DPDK~~)
  - however the largest contributor
- Regression must be checked on every supported machines
  - best effort from contributors



	ARMv7	ARMv8	Power8	x86-32	x86-64
lib/acl	✓	✓	✓	✓	✓
lib/distributor	✓	✓	✓	✓	✓
lib/hash	✓	✓	✓	✓	✓
lib/ip_frag	✓	✓	✓	✓	✓
lib/lpm	✓	✓	✓	✓	✓
lib/reorder	✓	✓	✓	✓	✓
lib/sched	✓	✓	✓	✓	✓
net/bnx2x		-		-	✓
net/bnxt		-		-	✓
net/cxgbe		-		✓	✓
net/e1000		-		✓	✓
net/ena		-		✓	✓
net/enic		-		✓	✓
net/fm10k				✓	✓
net/i40e		✓		✓	✓
net/ixgbe		✓		✓	✓
net/mlx4			✓	✓	✓
net/mlx5			✓	✓	✓
net/nfp		-		-	✓
net/qede		-		-	✓
net/szedata2					✓
net/thunderx		✓			
net/vhost	✓	✓	✓	✓	✓
net/virtio	✓	✓	✓	✓	✓

Architecture-specific Implementation

- reusable and generic code in EAL
  - `librte_eal/common/include/generic/`
  - `librte_eal/common/include/arch/`
  - `librte_eal/common/arch/`
- library or driver specific code in separate files
  - `librte_acl/acl_run_altivec.c`
  - `librte_acl/acl_run_avx2.c`
  - `librte_acl/acl_run_neon.c`
  - `librte_acl/acl_run_scalar.c`
  - `librte_acl/acl_run_sse.c`
- build-time CPU features supported by the compiler
  - `#ifdef RTE_MACHINE_CPUFLAG_*`
- run-time CPU detection
  - `rte_cpu_get_flag_enabled(RTE_CPUFLAG_*)`
  - best available optimization in only one build/package (e.g. SSE3/SSE4/AVX2/AVX512)

Function Multi-Versioning

- manual/legacy method (currently used in DPDK)
  - specific compilation of whole file
  - function pointer defined at run-time
- flatten function attribute
  - inline calls in the function
  - allow more code to be optimized by compiler
- target function attribute
  - build function with specific flags
- target_clones function attribute
  - build function clones with specific flags
  - select the best one at run-time through ifunc resolver
  - no manual tuning

Vector (SIMD) Optimizations

- ISA-specific intrinsic functions
- generic GCC vector type
  - __attribute__ ((vector_size (n)))
  - limited to simple operations

- Maintenance of vectorized code
  - Who is responsible and/or expert? lib maintainer? arch maintainer?
  - How to coordinate a change affecting several drivers on several architectures?
  - Risk of deviating features/behaviour in driver paths

CPU/cache Optimizations

- Many techniques
  - hot/cold attributes
  - inlining
  - cache alignment
  - bulk
  - prefetch
  - ...

- How generic is the performance gain (or loss)?

Long list of supported Device

- multi-bus
  - PCI
  - SoC
  - virtual
- generic interfaces
  - net (ethdev)
  - crypto (cryptodev)

Feature	af_packet	bnx2x	bnx2xvf	bnxt	bonding	cxgbe	e1000	ena	enic	fm10k	fm10k_vec	fm10kvf	fm10kvfvec	i40e	i40e_vec	i40evf	i40evf_vec	igb	igbvf	ixgbe	ixgbe_vec	ixgbevf	ixgbevfvec	mlx4	mlx5	mpipe	nfp	null	pcap	qede	qedevf	ring	szedata2	thunderx	vhost	virtio	virtio_vec	vmxnet3	xenvirt
Speed capabilities																																						
Link status	Y	Y	Y		Y	Y		Y						Y	Y			Y		Y	Y	Y	Y	Y	Y				Y	Y		Y	Y	Y	Y	Y	Y	
Link status event	Y	Y				Y		Y						Y	Y			Y		Y	Y			Y	Y				Y	Y				Y	Y		Y	
Queue status event																																		Y					
Rx interrupt						Y			Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y																
Queue start/stop				Y		Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y			Y	Y								Y	Y		Y	Y	Y	
MTU update						Y	Y	Y	P									Y		Y	Y	Y	Y	Y	Y				Y	Y				Y				Y	
Jumbo frame						Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y				Y	Y	Y			Y				Y	
Scattered Rx						Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y								Y	Y		Y				
LRO																				Y	Y	Y	Y															Y	
TSO						Y		Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y															Y	
Promiscuous mode	Y	Y	Y			Y	Y		Y	Y	Y			Y	Y	Y	Y	Y	Y	Y	Y			Y	Y				Y	Y		Y	Y		Y	Y	Y	
Allmulticast mode						Y	Y		Y	Y				Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y		Y	Y		Y	Y	Y	
Unicast MAC filter	Y	Y	Y			Y		Y	Y	Y				Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y					Y	Y	Y	
Multicast MAC filter	Y	Y	Y					Y	Y	Y				Y	Y	Y	Y							Y	Y				Y	Y					Y	Y		
RSS hash					Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y				Y	Y			Y					Y	
RSS key update						Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y		Y				Y	Y			Y						
RSS reta update			Y			Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y		Y				Y	Y			Y						
VMDq								Y	Y					Y	Y			Y		Y	Y																		
SR-IOV		Y				Y								Y	Y			Y		Y	Y			Y	Y					Y			Y						
DCB														Y	Y			Y		Y	Y																		
VLAN filter						Y		Y	Y	Y				Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y					Y	Y	Y	
Ethertype filter														Y	Y			Y		Y	Y																		
N-tuple filter																		Y		Y	Y																		
SYN filter																		Y		Y	Y																		
Tunnel filter														Y	Y					Y	Y																		
Flexible filter																		Y																					
Hash filter														Y	Y	Y	Y																						
Flow director									Y					Y	Y					Y	Y			Y															
Flow control					Y	Y								Y	Y			Y		Y	Y								Y	Y									
Rate limitation																				Y	Y																		
Traffic mirroring														Y	Y					Y	Y																		
CRC offload					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y		Y		Y					Y	Y			Y						
VLAN offload					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y		Y		Y					Y	Y			P					Y	
QinQ offload						Y								Y		Y		Y	Y	Y		Y																	
L3 checksum offload					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y		Y		Y	Y								Y						
L4 checksum offload					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y		Y		Y	Y								Y					Y	
Inner L3 checksum							Y							Y		Y				Y		Y		Y															
Inner L4 checksum							Y							Y		Y				Y		Y		Y															
Packet type parsing					Y			Y	Y	Y	Y	Y	Y	Y		Y		Y	Y	Y		Y		Y	Y					Y	Y		Y				Y	
Timesync														Y	Y			Y		Y	Y																		
Basic stats	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y	Y		Y	Y	Y	Y	Y	Y	
Extended stats	Y	Y	Y				Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y							Y	Y		Y		Y				
Stats per queue					Y				Y	Y	Y	Y								Y	Y			Y	Y					Y	Y		Y	Y		Y	Y	Y	

Unlock the full power of the Devices

- offloads in NIC
  - load balancing (flow steering)
    - new Rx filtering API
  - segmentation offloads (LRO, TSO)
    - new software implementation for virtio
  - checksum offloads
    - new flags
- common support
  - software emulation to fill the gaps
- early access to hardware features
  - unstable API
  - specific features in common interface or picked in drivers?

Unlock the full power of the Machine

- custom mempool handlers
  - not used yet?
- event driven model
  - NPU
  - other usages in software design?

Flexible Packaging

- split in multiple libraries
  - or combined in one linker script
- static .a
  - more efficient
- dynamic .so
  - distributions choice
- drivers as plugins
- standard make install (since v2.2)

- integrated in some distributions

Choices of Linux kernel Bypass

- userspace-friendly kernel modules
  - vmxnet3-usermap
  - mlx/verbs thanks to RDMA
    - no root access required
    - less code in PMD
      - i40e: 34.8 kSLOC
      - mlx4: 4.2 kSLOC
- UIO kernel modules
  - igb_uio (out-of-tree)
  - uio_pci_generic (no MSI/MSI-X, i.e. no VF device)
- VFIO kernel module
  - vfio-pci (IOMMU, performance loss?)
  - vfio-pci noiommu mode (since v4.5)

Multiple Environments

- not only Linux
- FreeBSD
- not only kernel bypass in a full blown OS?
  - OsV unikernel?
- no hugepage
  - works with virtual devices
  - requires more work for DMA

Usability

- more/better default values
  - command line (-m, -n, -c, etc)
  - thresholds
- avoid build-time configuration
- run-time configuration
  - by application
    - argc/argv must be replaced by a simpler API
  - by user
    - command line
    - file
    - specific to the application

From bare-metal Framework to Library

- a long road
- ease compilation in existing projects
  - must generate a pkg-config file
- pluggable logs
  - should be fixed now
- avoid forcing application design
  - thread management?
- no exit()
  - kill rte_panic() in libraries

Developer Tools

- More debug tools
  - pdump
  - valgrind
- Language Bindings
  - C native
  - C++ supported as best effort
  - Other generic languages? Go? Rust?
  - Specific languages? P4? eBPF/XDP?

Who is driving this super car?

- Vendors
  - show capabilities of their devices and processors
- R&D labs
  - userspace development accelerate time to market
- Manufacturers
  - highest performance

Where is it Used?

- Equipments
  - Telecom, High End Switches, Large Volume Servers, Security
- Technology
  - Legacy, SDN, NFV
- OS
  - RHEL, Fedora, Ubuntu, Clear Linux, Mirantis OpenStack
- Stacks
  - 6WIND, OVS, BESS, VPP, ODP, OpenFastPath, Seastar, ANS, mTCP, Butterfly, Packet-journey
- Traffic Generators
  - pktgen-dpdk, Moongen, TRex, WARP17

Gathering Contributors

- started for x86 with Intel drivers only
- 2012: DPDK users working without cooperation
  - private DPDK forks
- 2013: 6WIND launched dpdk.org initiative
- other similar projects were started
  - Cisco VPP (closed source before this year)
  - Italian projects from Pisa University (Netmap, PF_RING, PFQ)
  - 2016: Linux kernel start building XDP solution
- 2016: major hardware vendors involved in DPDK
  - IBM Power and ARM architectures
  - drivers for almost all fast NICs
- 2017: network processors (NPU)?
- New Governance?

2016 - Welcome ARM!

Questions?

Thomas Monjalon

thomas.monjalon@6wind.com

tmonjalo / freenode #DPDK