FTF FREESCALE TECHNOLOGY FORUM 2014

Data Path Acceleration Architecture (DPAA) Usage Scenarios

FTF-NET-F0147

Sam Siu

APR.2014

freescale™

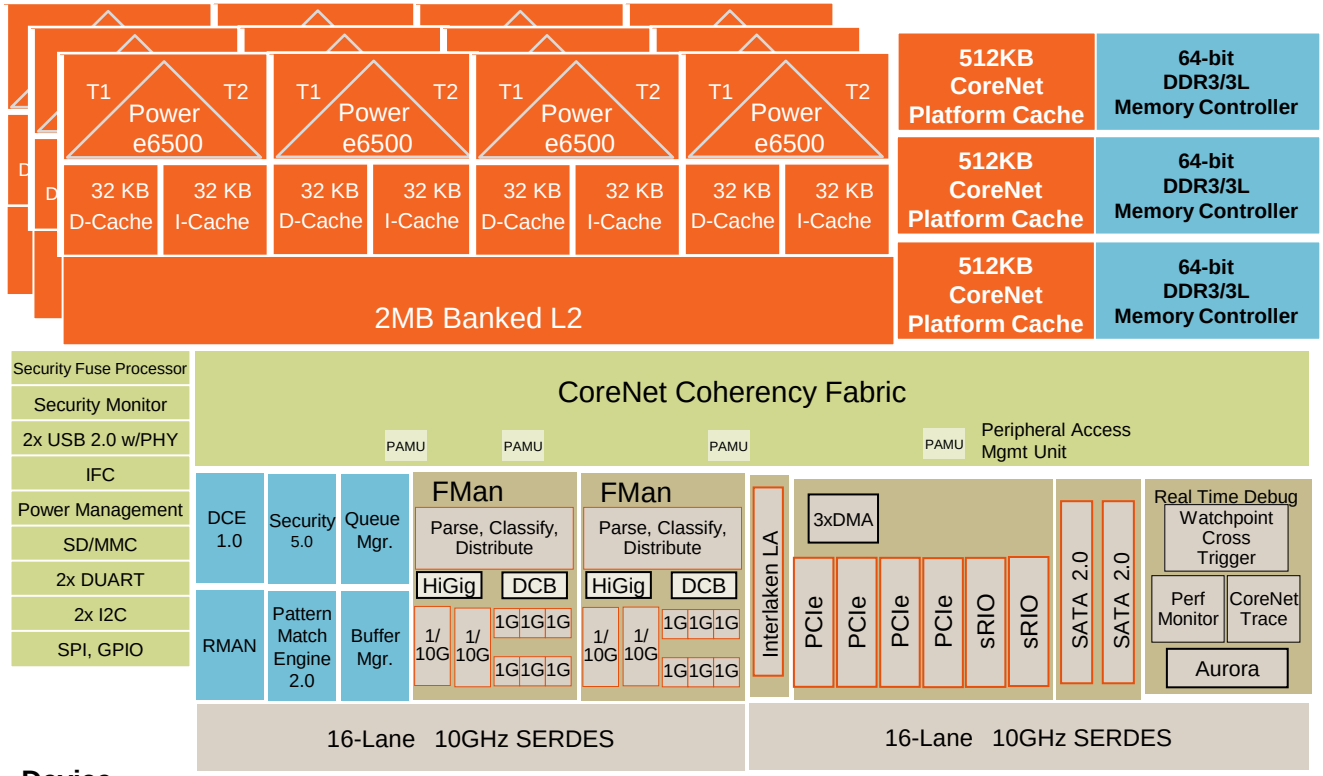
External Use

Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, Qorivva, SafeAssure, the SafeAssure logo, StarCore, Symphony and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SMARTMOS, Tower, TurboLink, UMEMS, Vybrid and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2014 Freescale Semiconductor, Inc.

Agenda

- QorIQ Data Path Acceleration Architecture (DPAA)
- QorIQ Use Case:
  - User Space Application Accessing DPAA (USDPAA)
  - Virtualization (KVM) and Software Define Network (SDN)
  - Intelligent Network Interface Card (iNIC)
  - Data Center Server with DCB
  - Smart Network Appliance: Data Replicator with DPAA Accelerator (FMAN/DCE/PME)
- Summary

QorIQ T4240



Processor

- 12x e6500, 64b, up to 1.8GHz
- Dual threaded, with128b AltiVec
- Arranged as 3 clusters of 4 CPUs, with 2MB L2 per cluster; 256KB per thread

Memory SubSystem

- 1.5MB CoreNet Platform Cache w/ECC
- 3x DDR3 Controllers up to 1.87GHz
- Each with up to 1TB addressability (40 bit physical addressing)

CoreNet Switch Fabric

High Speed Serial IO

- 4 PCIe Controllers, with Gen3
  - SR-IOV support
- 2 sRIO Controllers
  - Type 9 and 11 messaging
  - Interworking to DPAA via Rman
- 1 Interlaken Look-Aside at up to10GHz
- 2 SATA 2.0 3Gb/s
- 2 USB 2.0 with PHY

Network IO

- 2 Frame Managers, each with:
  - Up to 25Gbps parse/classify/distribute
  - 2x10GE, 6x1GE
  - HiGig, Data Center Bridging Support
  - SGMII, QSGMII, XAUI, XFI

Device

- TSMC 28HPM Process
- 1932-pin BGA package
- 42.5x42.5mm, 1.0mm pitch

Power targets

- ~54W thermal max at 1.8GHz
- ~42W thermal max at 1.5GHz

Datapath Acceleration

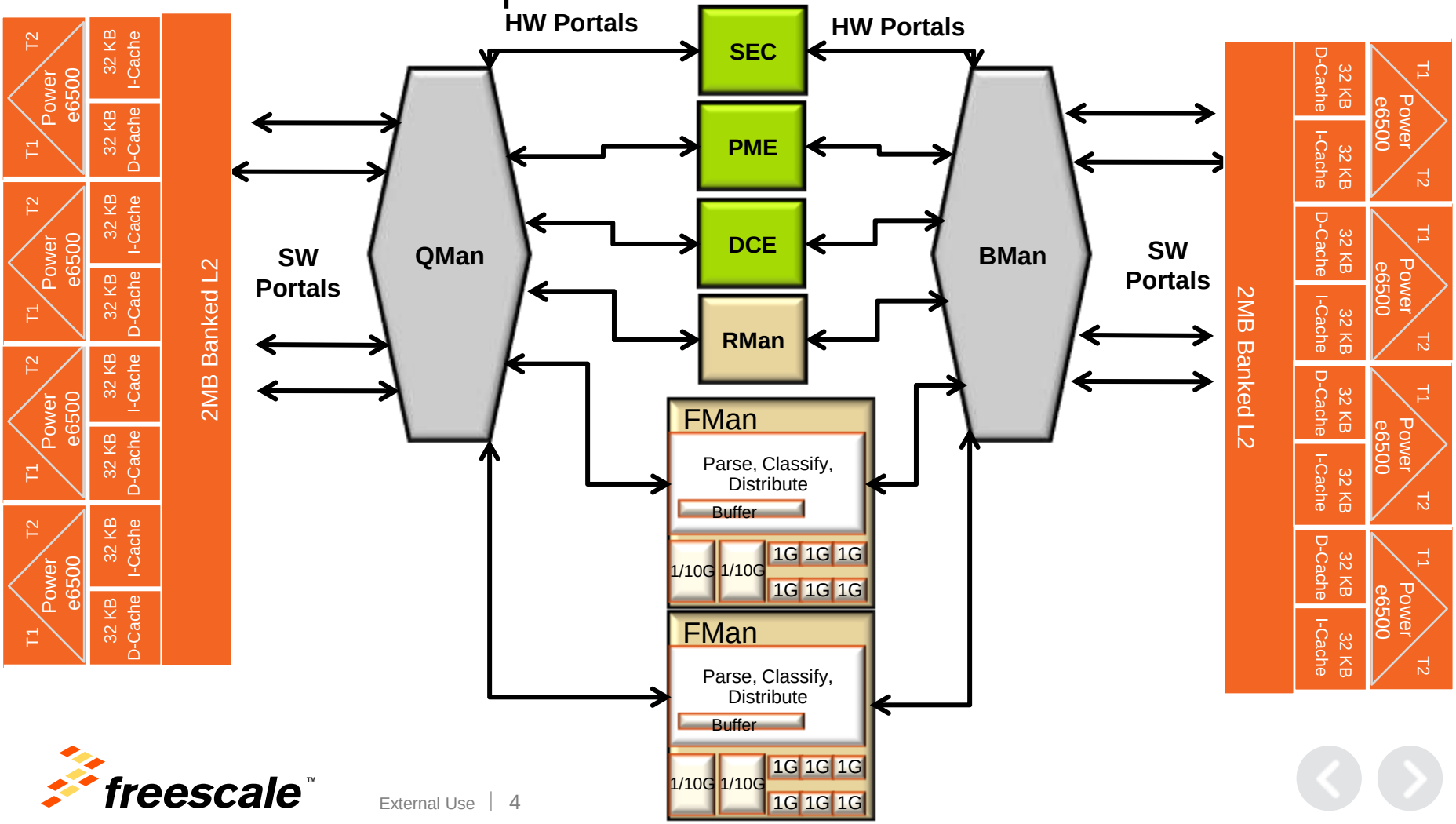
- **SEC**- crypto acceleration 40Gbps
- **PME-** Reg-ex Pattern Matcher 10Gbps
- **DCE-** Data Compression Engine 20Gbps

Industry's Most Scalable Processor Portfolio

	T2080	T4080	T4160	T4240
CPU (64b)	e6500			
# of CPU cores, threads	4, 8 threads		8, 16 threads	12, 24 threads
Max frequency	1.8GHz			
L2 Cache per core	512KB			
Platform Cache	512KB	1MB		1.5MB
DRAM Interface	1x DDR 64b 3/3L	2x DDR 64b 3/3L		3x DDR 64b 3/3L
IPFwding perf (small pkt)	24Gbps	24Gbps	36Gpbs	48Gbps
IPsec perf (large pkt)	14Gbps	32Gpbs		
Max # Ethernet	4x 1/10GbE + 4x 1GbE	2x 1/10 GbE + 14x 1GbE		4x 1/10GbE + 12x 1GbE
Other High Speed Serial	4x PCIe: Gen 2.0/3.0	4x PCIe: Gen 2.0/3.0		4x PCIe: Gen 2.0/3.0
Power (typ 65C)	11W-1.2GHz	19W-1.5GHz	25W-1.5GHz	30W-1.5GHz
Power (max 105C)	28W-1.8GHz	47W-1.8GHz	53W-1.8GHz	63W-1.8GHz
Pin Compatibility	25x25 mm 896p FCBGA	42.5x42.5mm 1932-pin FCBGA		

QorIQ Datapath Acceleration Architecture

- Any packet to any CPU to any accelerator or network interface without locks or semaphores



User Space Application Accessing DPAA (USDPAA)

USDPAA Software Overview

- Linux user space processes that contain at least one USDPAA thread, which can directly access DPAA hardware for maximal data plane performance
- Linked with a user space library providing a driver layer for the portals and an access and control API
- Rely on the Linux Userspace I/O (UIO) framework, for mappings and interrupt handling
  - See kernel.org/doc/htmldocs/uio-howto.html
- Run in the context of an SMP Linux instance on a CoreNet SoC (e.g. T4240)
- User space driver libraries, do not need system calls to do I/O
  - Need not switch into and out of the kernel's execution context
- User space applications can directly access data buffers
  - Guarantees zero copy I/O in all cases
- A BMan and a QMan software portal are allocated for the USDPAA application to allow direct access
  - No other thread or entity accesses these portals

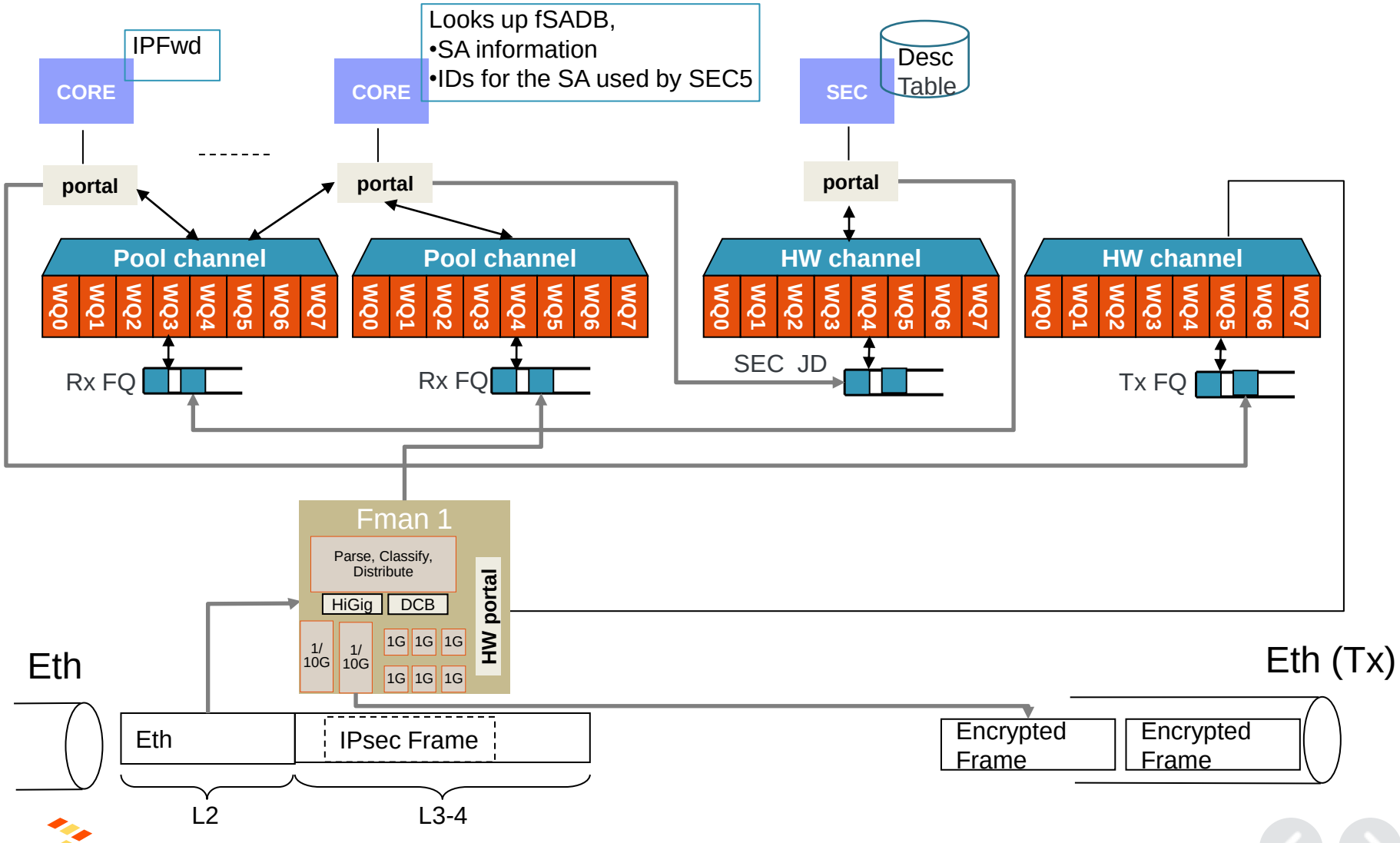
USDPAA Components

- Device-tree handling
  - Configuration and resource details are defined within the "device-tree" used to
    boot Linux
- QMan and BMan drivers and C API
  - The Queue Manager (QMan) and Buffer Manager (BMan) drivers are the heart
    of USDPAA
- DMA memory management
  - The Freescale DPAA hardware provides several peripherals such as FMan, SEC, and PME that read and write memory directly using DMA
- Network configuration
  - The USDPAA QMan and BMan drivers do not, in and of themselves, dictate which resources such as frame queues or buffer pools are used
- CPU isolation
- Packet Processing Application Core/Module (PPAC/PPAM)
- SEC Run Time Assembly (RTA), Descriptor construction library (DCL)

USDPAA Sample Applications

- USDPAA Application with Packet Processing Application Core (PPAC)
  - An IP forwarding performance demonstration, "ipfwd"
  - An IPFwd application based upon Longest Prefix Match methodology, "lpm_ipfwd"
  - An application to route IPv4 packets after performing encryption/decryption, "IPsecfwd"
  - A cryptographic accelerator example, "simple_crypto"
    - SEC Descriptor construction library (DCL)
    - Runtime Assembler Library (RTA)
  - A pattern-matching accelerator example, "pme_loopback_test"
  - Freescale USDPAA Freescale RMan Application (FRA)
  - Freescale USDPAA Serial RapidIO application (SRA)
  - USDPAA RapidIO Message Unit Application (RMU)
- A non-PPAC based stand-alone application
  - "hello_reflector"

Use Case: USDPAA IPsec Forwarding Application

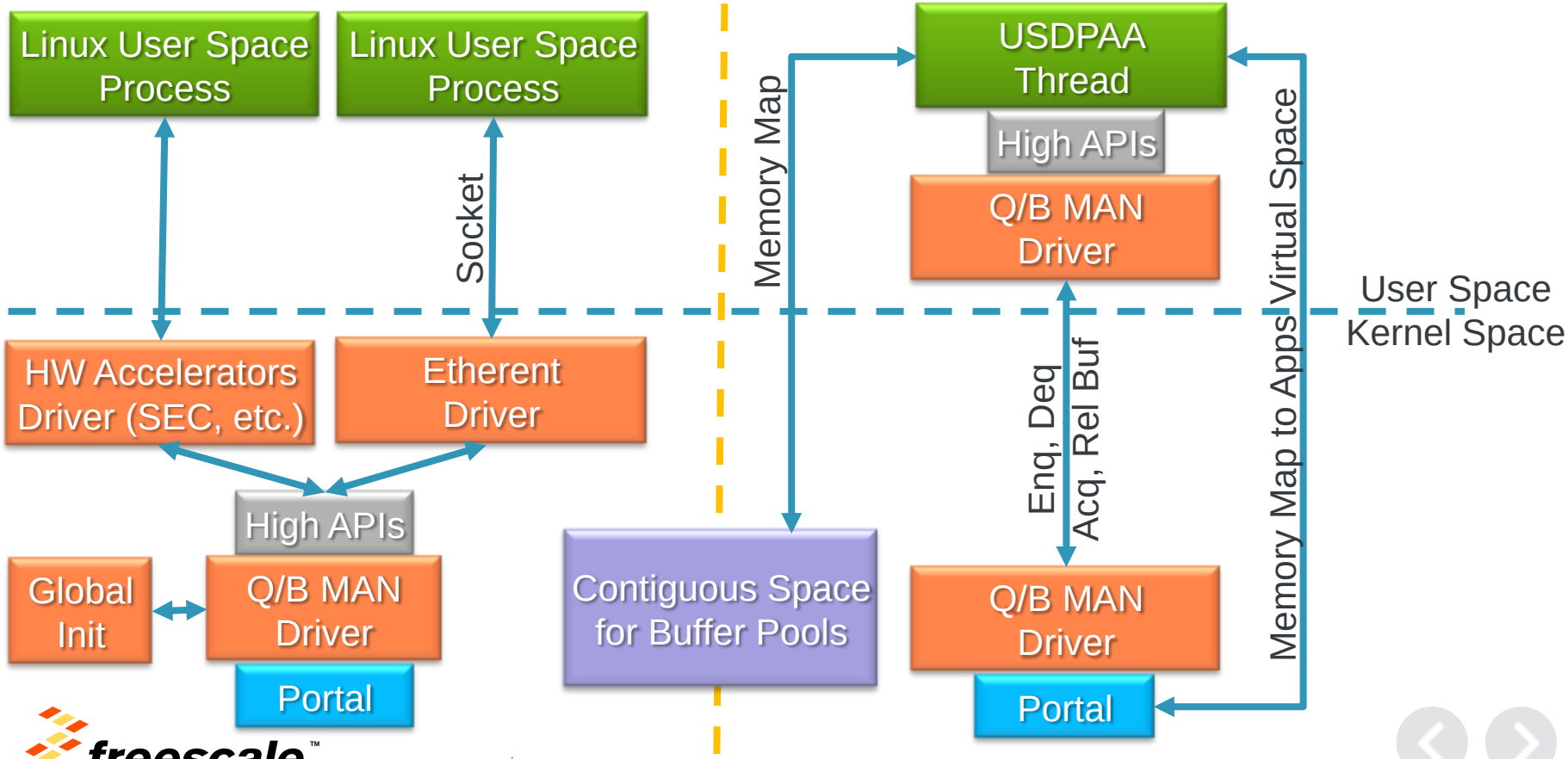


USDPAA Software Model

- Run-to-Completion Model
  - USDPAA threads continuously poll their portals for available work, e.g. servicing non-empty FQs:
    - Threads are always running or ready to run
    - The associated cores will appear 100% loaded
    - QMan's hardware based priority scheduler effectively distributes work to the cores
    - Affine to a core to allow for stashing to per-core caches
- Interrupt-Driven Model
  - The Linux UIO framework allows USDPAA threads to wait for interrupts from software portals by doing file operations like select(fd) on the user space device
  - USDPAA threads will dequeue and process frames from portals after a data available interrupt (enqueued FD's available)
    - Until dequeue processing is complete (no more enqueued FD's available)
    - The interrupt is then re-enabled
  - More operating system overhead than run-to-completion model
  - PPAC-based example applications implement a hybrid:
    - Interrupt-driven mode is used when the packet-processing has been idle for a short period of time
    - Switches back to run-to-completion once processing resumes

Kernel And User Space QMan/BMan Portal Drivers

- USDPAA provides Linux user space applications direct access to DPAA Queue Manager and Buffer Manager software portals. No system call or kernel context switch to access a portal
- The physical address ranges of software portals are mapped into the virtual address space of user space processes
- User space applications can use normal load and store instructions to perform operations on the portals



USDPAA: DMA Memory Management

- FMan, SEC, PME read / write memory directly using DMA
  - Buffers allocated from DMA Memory
- Freescale USDPAA shared memory driver:
  - Kernel reserves a contiguous region of memory of 64MB (default) very early in the kernel boot process for use as DMA memory
  - Memory size and alignment is hard-coded into the kernel via Kconfig option
    - CONFIG_FSL_USDPAA_SHMEM
    - Device Drivers → Misc devices → Freescale USDPAA shared memory driver
  - Reserved memory is exposed via device /dev/fsl_usdpaa_shmem
  - Hook is placed in memory-management code to "catch" page faults within this memory range and ensure that they are resolved by a single TLB1 mapping that spans the entire memory reservation
- User Space "dma_mem" Driver
  - ioctl()
    - Copy memory region physical start address and size to struct in user space
  - mmap()
    - Map physical memory region to a contiguous range in application's virtual address space
  - Compute difference between physical addr and virtual address
    - dma_mem_ptov(), dma_mem_vtop() APIs

USDPAA: Linux Kernel QMan/BMan Drivers

- Configuration Interface
  - CCSR register space and global/error interrupt source

```
179         bman-portals@ff4000000 {
180                 #address-cells = <0x1>;
181                 #size-cells = <0x1>;
182                 compatible = "simple-bus";
183                 ranges = <0x0 0xf 0xf4000000 0x200000>;
184                 bman-portal@0 {
185                         cell-index = <0x0>;
186                         compatible = "fsl,p4080-bman-portal", "fsl,bman-portal";
187                         reg = <0x0 0x4000 0x100000 0x1000>;
188                         cpu-handle = <&cpu0>;
189                         interrupts = <105 2 0 0>;
190                 };
191                 bman-portal@4000 {
192                         cell-index = <0x1>;
193                         compatible = "fsl,p4080-bman-portal", "fsl,bman-portal";
194                         fsl,usdpaa-portal;
195                         reg = <0x4000 0x4000 0x101000 0x1000>;
196                         cpu-handle = <&cpu1>;
197                         interrupts = <107 2 0 0>;
198                 };
```

- Presence of a portal node property "**fsl,usdpaa-portal**" indicates the portal is dedicated to a USDPAA thread
- Otherwise portal will be used only within Linux kernel

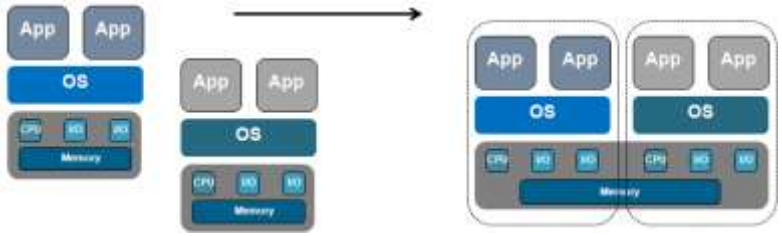
USDPAA: QMan/BMan UIO Portal Drivers Interface

- Standard Linux Userspace I/O (UIO) System
  - Each UIO device is accessed through a device file (/dev/uio0, /dev/uio1, ...) and sysfs attribute files
  - User Space Library layered on top of UIO infrastructure
    - BMan API examples
      - bman_new_pool(), bman_release(), bman_acquire()
    - QMan API examples
      - qman_create_fq(), qman_init_fq()
      - qman_poll_dqrr(), qman_enqueue()
    - USDPAA-specific APIs
      - qman_thread_init(), bman_thread_init()
- Linux kernel driver
  - struct dpa_uio_info
  - dpa_uio_open(), dpa_uio_release(), dpa_uio_mmap(), dpa_uio_irq_handler()
- User Space driver
  - open() device
  - mmap()
  - bman_create_affine_portal(), qman_create_affine_portal()
    - Portal initialization
    - Similar to Linux kernel driver for portals used by kernel

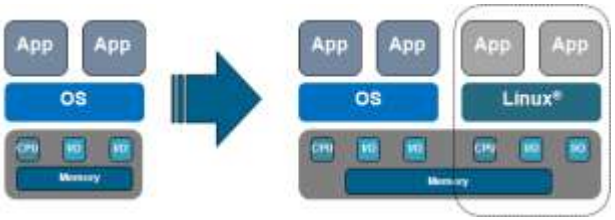
Virtualization Support

Virtualization Use Cases

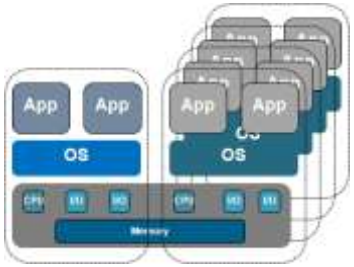
Cost Reduction/Consolidation



Security/Sandboxing

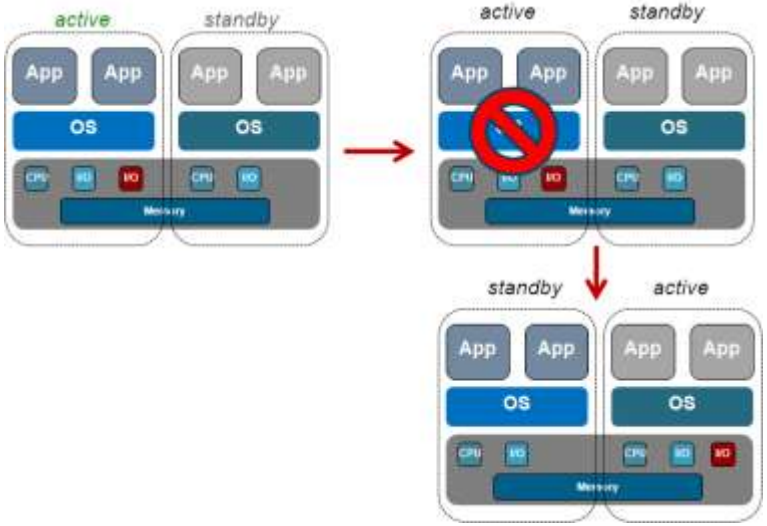


Utilization



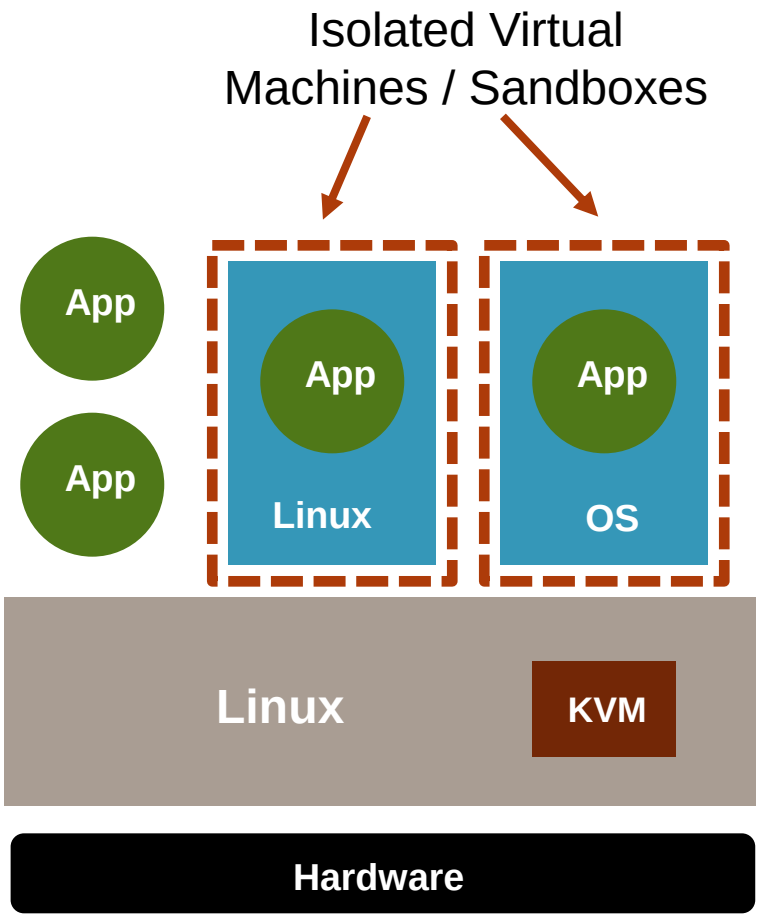
Dynamic Resource Management

Fail Over



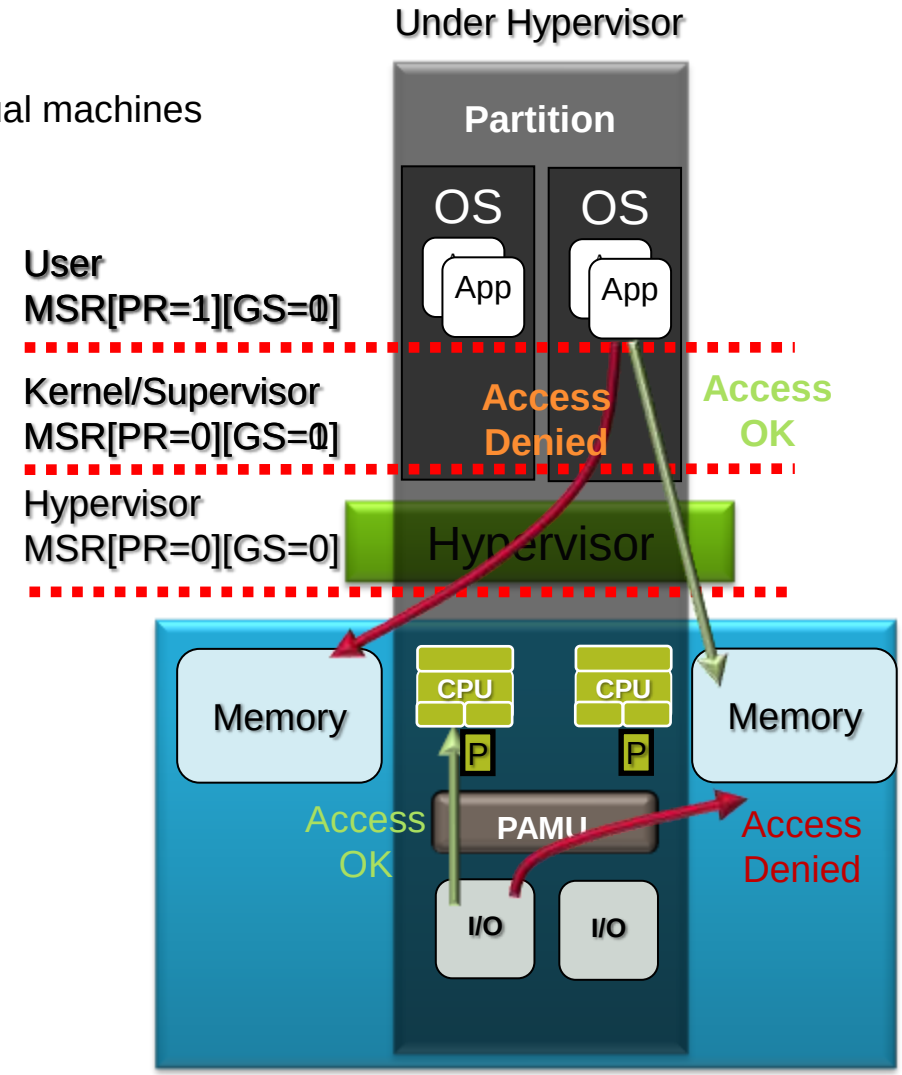
What Do Virtualization Technologies Enable?

- Sandboxing – allows untrusted software to be added to a system (e.g. operator applications)
- Run legacy software or OS on Linux
- Use different versions of the Linux kernel
- Improved hardware utilization
- Create/destroy VMs as needed
- Better management of resources
  - Allocation of physical CPUs
  - Manage allocation of % CPU cycles
- Migrate running VM to different system



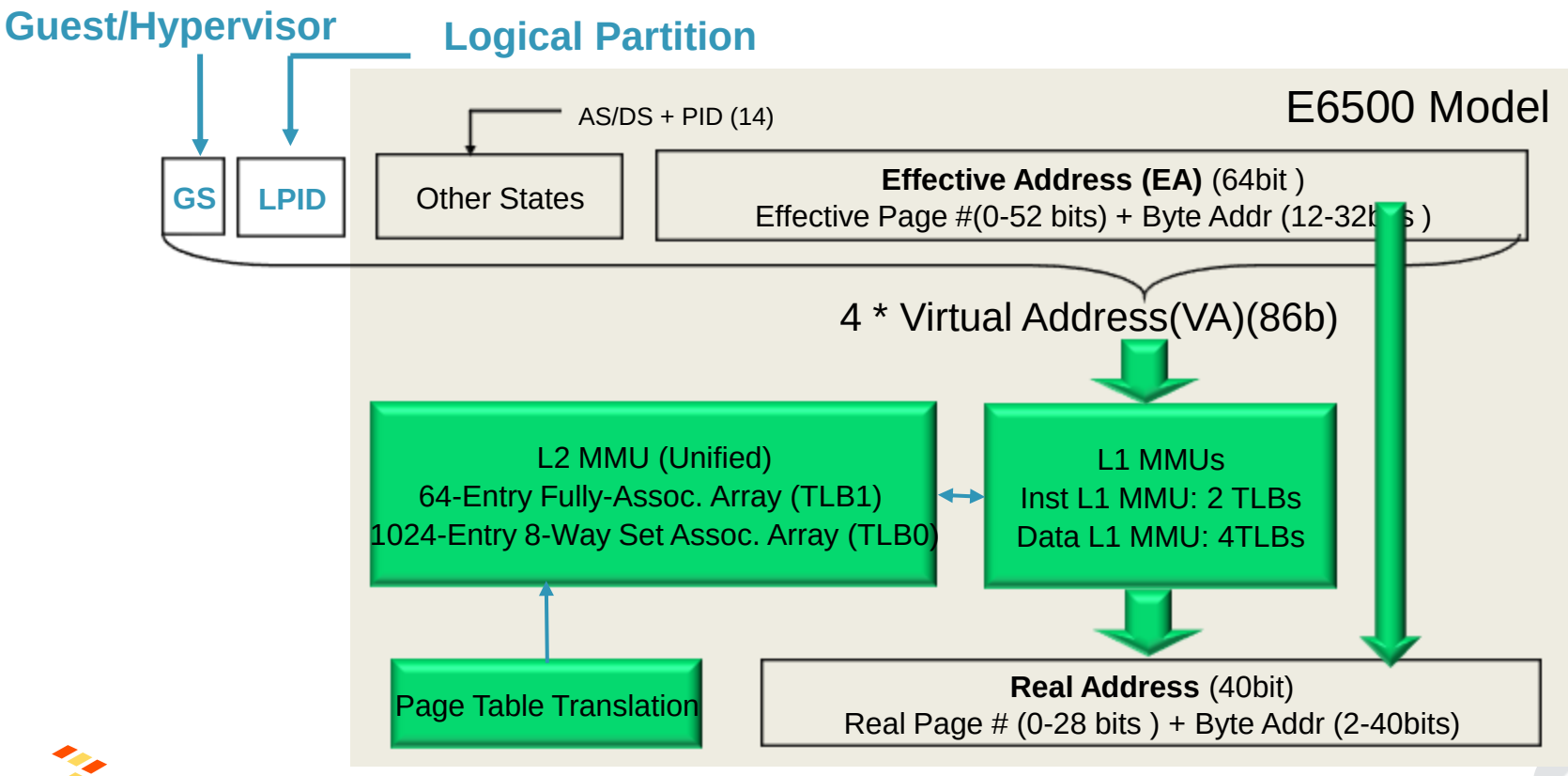
Virtualization Features in QorIQ Silicon

- Hypervisor (Topaz) runs “bare metal”
  - software component that creates and manages virtual machines
- CPU
  - e500mc / e5500 / e6500
  - 3rd privilege level
  - Partition ID / extended virtual address space
  - Shadow registers
  - Direct system calls
  - Direct external hardware interrupts to guest
- SoC
  - IOMMU (PAMU)
    - Provides isolation from I/O device memory accesses
  - Portal
    - Data path portal is assigned and dedicated to partition
- Software Ready Features:
  - virtio network and block
  - hugetlbfs support
  - Libvirt
  - in-kernel MPIC
  - QEMU debug stub
  - passthru of PCI devices (vfio-pci)



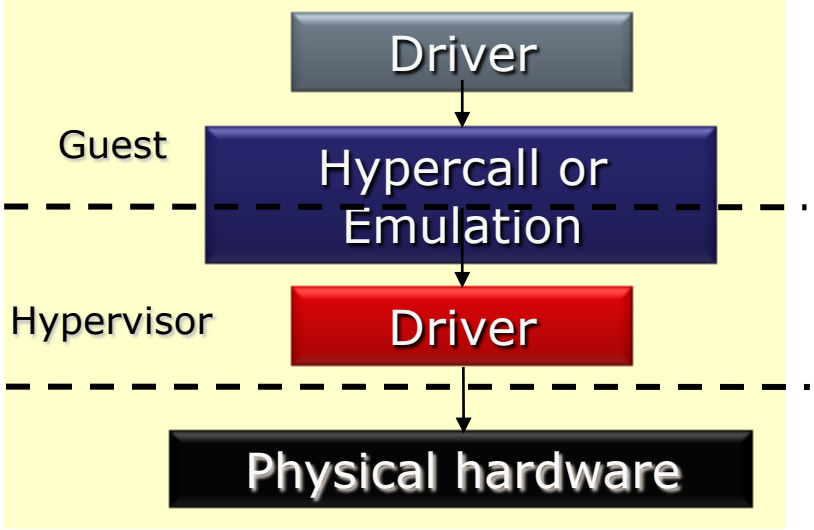
e6500 MMU Address Translation

- The fetch and load/store units generate 64-bit effective addresses.
- The MMU translates these addresses to 40-bit real addresses using an interim virtual address.
- In multicore implementations, such as the e6500, the virtual address is formed by concatenating MSR[GS] || LPIDR || MSR[IS|DS] || PID || EA,



Virtualized I/O

- PIC
- I2C
- GPIO
- UART and Byte-channels
- Supported through
  - Hypervisor hypercall API + I/O driver



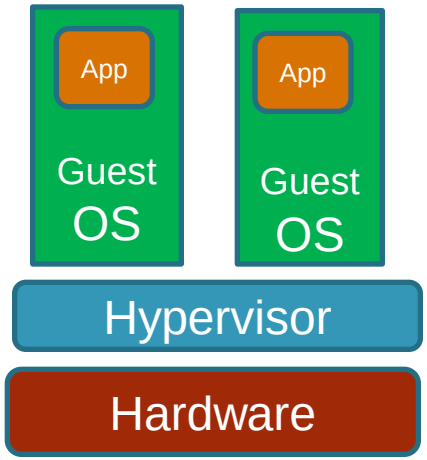
- External interrupts are processed by guest software in a partition. But MPIC hardware is not directly accessible by guest software
- Instead, a virtual MPIC (VMPIC) interface provides interrupt controller services
  - Guest accesses VMPIC via a hypercall interface
- All hardware interrupts that route to the MPIC node in the hardware device tree, will be routed to a VMPIC node in the guest device tree
- Direct end-of-interrupt (EOI)
  - An optional hypervisor mechanism by which in some cases an EOI can be performed with no hypercall

Sketch of Virtualization Technology on Power Architecture

- Enables the efficient and secure partitioning on a multi-core system

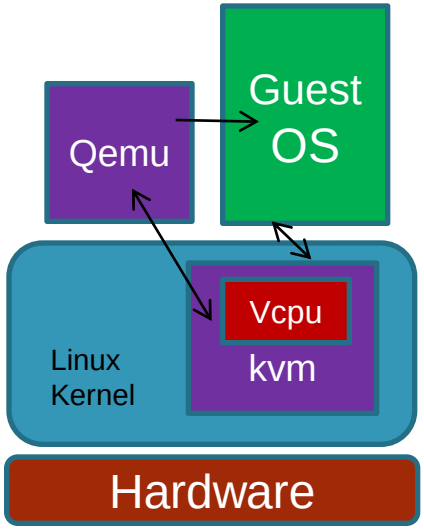
Topaz

- Enables the efficient and secure partitioning on a multi-core system



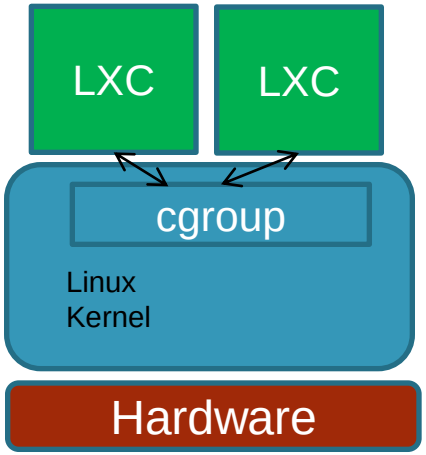
KVM

- A full isolated environment



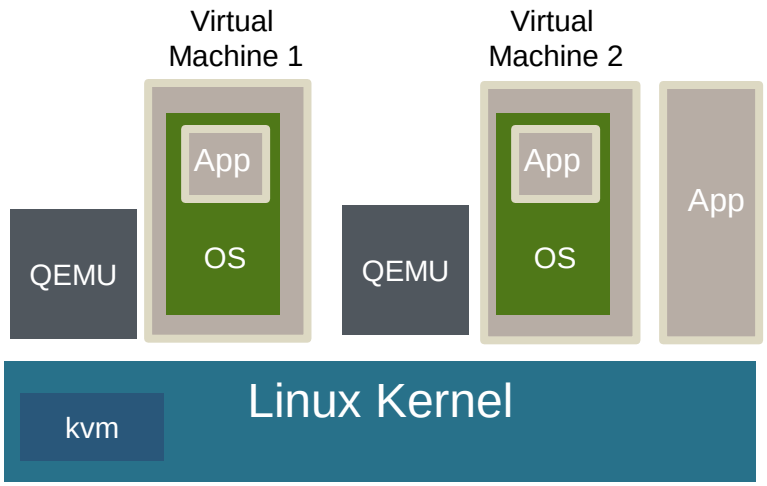
LXC

- OS-level virtual technology based on kernel



KVM Overview

- KVM/QEMU
  - open source virtualization technology based on the Linux kernel
  - Boot operating systems in virtual machines alongside Linux applications
  - No or minimal OS changes required
  - Virtual I/O – virtual disk, network interfaces, serial, etc.
  - Direct/pass thru I/O – assign I/O devices to VMs



- Scheduling / Context Switches
  - A QEMU/KVM virtual machine is sharing the CPU with other VMs and applications
  - Linux scheduler takes care of prioritization
  - When a guest is scheduled out/in there is overhead in saving/restoring guest state

Enable KVM

- Configure the Linux kernel to enable KVM-related features.
  - $ bitbake -c menuconfig linux-qoriq-sdk
  - From the main menuconfig window enable virtualization:
    - [*] Virtualization
  - In the virtualization menu enable the following options:
    - [*] KVM support for PowerPC E500MC/E5500/E6500 processors
  - Enable virto related interface
    - <*> PCI driver for virtio devices (EXPERIMENTAL) <*> Virtio block driver
    - <*> Universal TUN/TAP device driver support <*> Virtio network driver
- Add QEMU to the packages built by fsl-image-core
  - Edit the conf/local.conf file and append the following line which adds the QEMU package:
    - IMAGE_INSTALL_append = " qemu“
  - Build a guest root filesystem and add it to the host rootfs, then re-build the fsl-image-core image.
    - bitbake fsl-image-minimal; bitbake fsl-image-core
- Start QEMU
  - qemu-system-ppc -enable-kvm -m 512 -mem-path /var/lib/hugetlbfs/pagesize-4MB -nographic -M ppce500 -kernel /boot/uImage -initrd ./guest.rootfs.ext2.gz -append "root=/dev/ram rw console=ttyS0,115200" -serial tcp::4444,server,telnet
  - Connect to QEMU via telnet to start the virtual machine booting
- For detail information, refers to QorIQ SDK Documentation. KVM/QEMU > KVM for Freescale QorIQ Users Guide and Reference “.

KVM/QEMU Example

- A simple QEMU command line in a text file named kvm1.args:

  > cat kvm1.args /usr/bin/qemu-system-ppc -m 256 -nographic -M ppce500 -kernel /boot/uImage -initrd /home/root/my.rootfs.ext2.gz -append "root=/dev/ram rw console=ttyS0,115200" -serial pty -enable-kvm -name kvm1

- Converted QEMU command line to libvirt XML format:
  - > virsh domxml-from-native qemu-argv kvm1.args > kvm1.xml
- Define the domain:

  > virsh define kvm1.xml

  ```
  Domain kvm1 defined from kvm1.xml
  ```

- Start the domain. This starts the VM and boots the guest Linux.

  > virsh start kvm1

  ```
  Domain kvm1 started
  ```

  > virsh list

  ```
  Id Name State
  ---------------
  3 kvm1 running
  ```

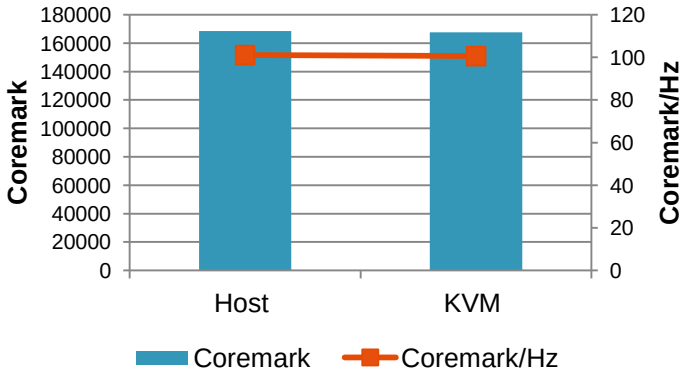
- The virsh console command connect to the console of the running Linux domain.

  > virsh console kvm1

  ```
  Connected to domain kvm1
  Escape character is ^]
  Poky 9.0 (Yocto Project 1.4 Reference Distro) 1.4 model : qemu ppce500 ttyS0
  model : qemu ppce500 login:
  Press CTRL + ] to exit the console.
  ```

T4240 CoreMark Results

Scenario		CoreMark Score	CoreMark / MHz
1	Host	168447	101.07
2	KVM	167619	100.57



Environment

- T4240 platform
- GCC4.7.3 (-O3 -mcpu=e6500 -m32 -mno-altivec)
- -DMULTITHREAD=24
- Linux SMP 24 cpus

Setup Comparison

1. Host (24 cpus)
2. KVM Guest with 24 vpcus

Conclusions

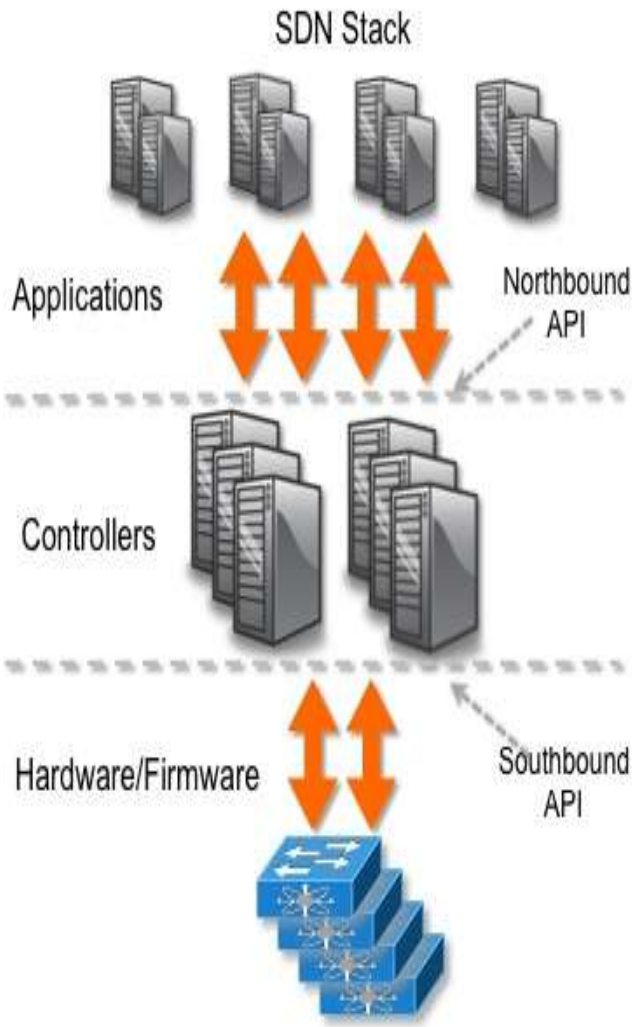
- Coremark scores in virtualized environments are close to bare metal
  - CPU operations are not impacted by virtualization
- Reasons
  - Limited memory management operations performed
  - All operations are tied to the core: matrix multiplication, list processing, CRC

Source: FTF-SDS-F0028-Benchmarking Virtualization Solutions for QorIQ Processors

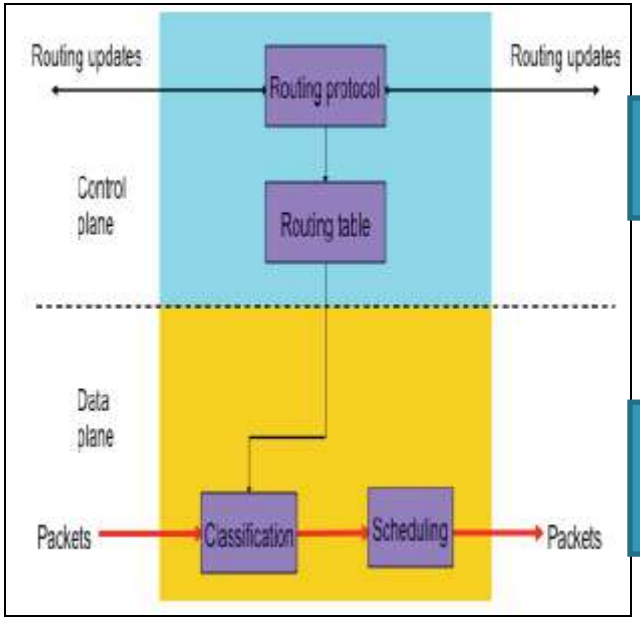
Software Defined Network (SDN)

Software Defined Networking (SDN)

- What is SDN?
  - SDN is a transformational networking paradigm that separates applications, control plane and data (forwarding) plane coupled with open standards based protocols, e.g. OpenFlow™
- What are the benefits?
  - Cloud services (any device, anywhere, any time)
  - Client-server moving to client-multiserver (M2M) to share server load
  - Push server virtualization into the routing network
  - Enterprise control over private cloud, public cloud and mix security policies
  - Networking equipment interoperability through a common protocol
- DN Strategy
  - SDN-optimized multi-core solutions
  - Contribute to OpenFlow standard
  - Lead OpenFlow Implementation

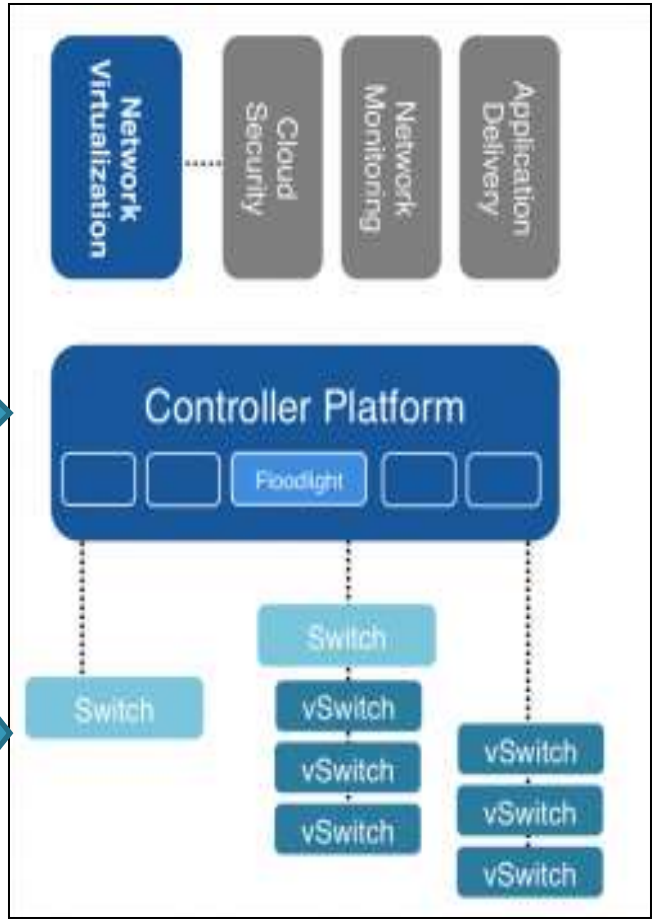


SDN, OpenFlow and Traditional Control Plane, Data Plane

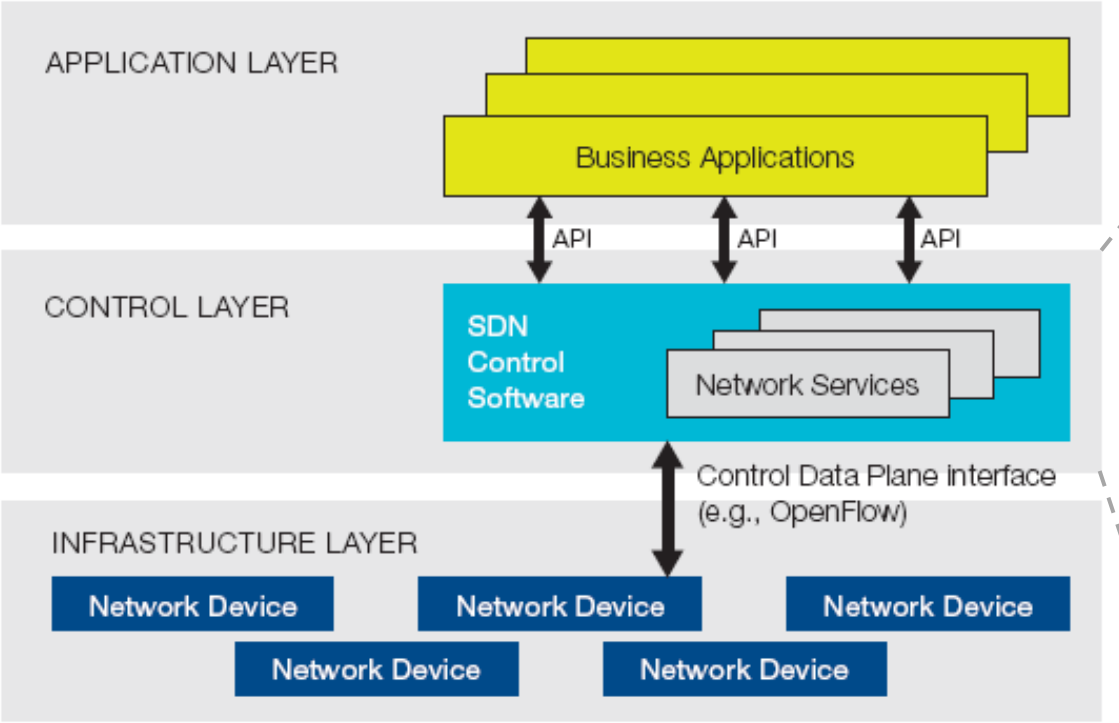


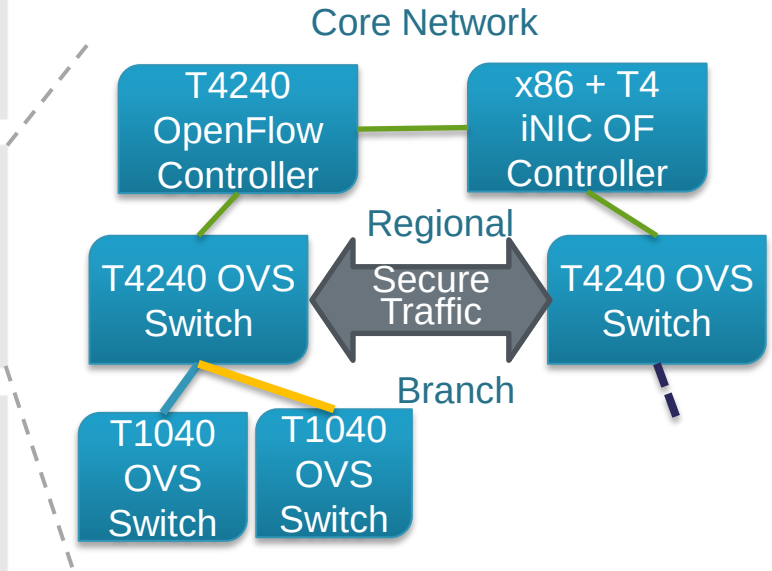
Centralizing Control Processor

Data Plane / Software Controlled Switching



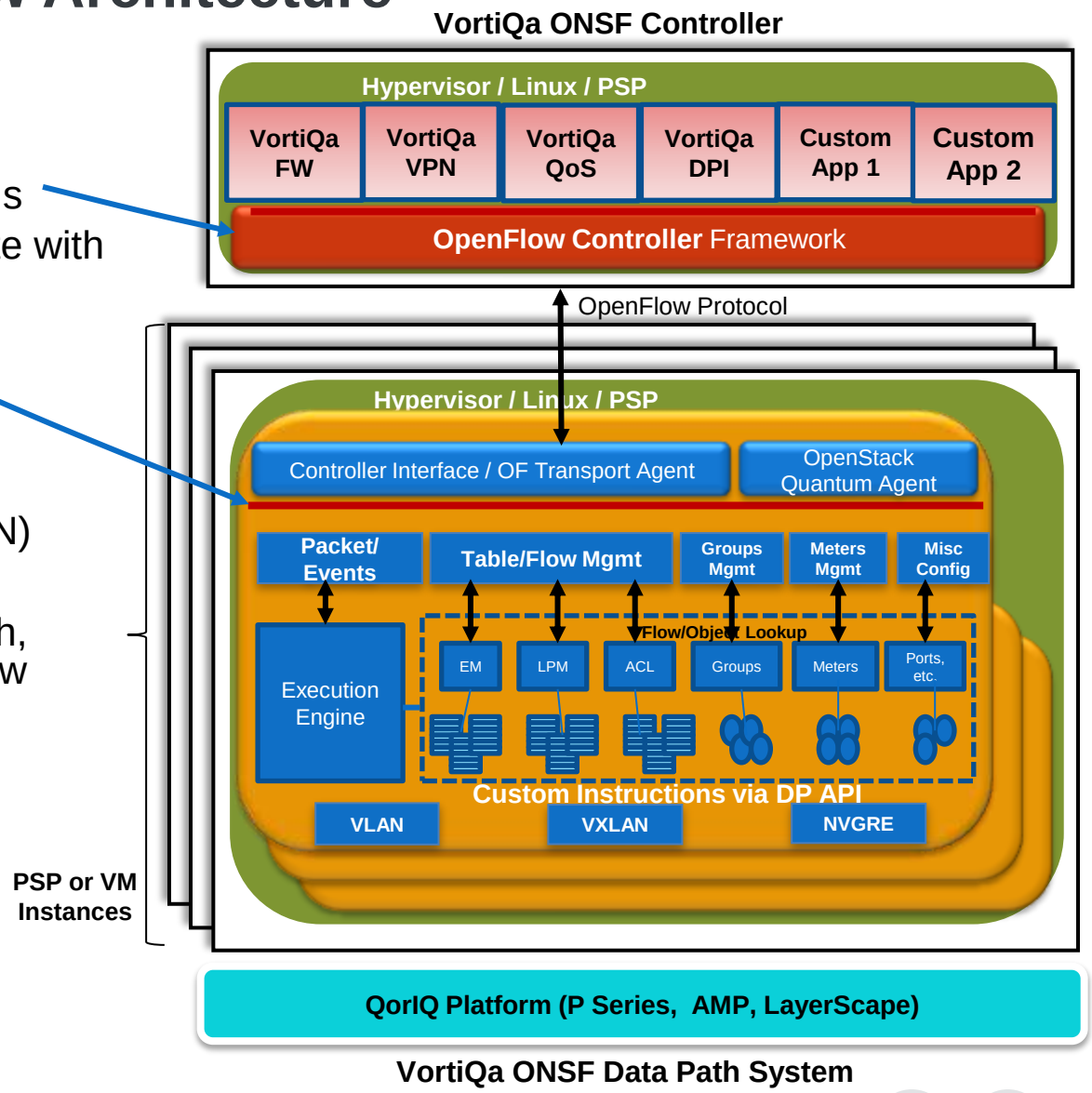
SDN Layers and OpenFlow(OF) Controller





VortiQa SDN OpenFlow Architecture

- ONSF Interfacing (VortiQa or Custom)
  - Apps mate with Northbound APIs
  - Custom instructions/actions mate with VortiQa DP API
- Data Plane Processing with OpenFlow Tables
  - Multiple Instances
  - Logical Interfaces (VLAN/VXLAN)
  - VortiQa APIs for DP mgmt
  - Search Algorithms - Exact Match, Radix Trie / LPM, Recursive Flow Classification
- OpenFlow Agents
  - DP management – uses VortiQa DP APIs
  - Quantum Agent for Network Virtualization

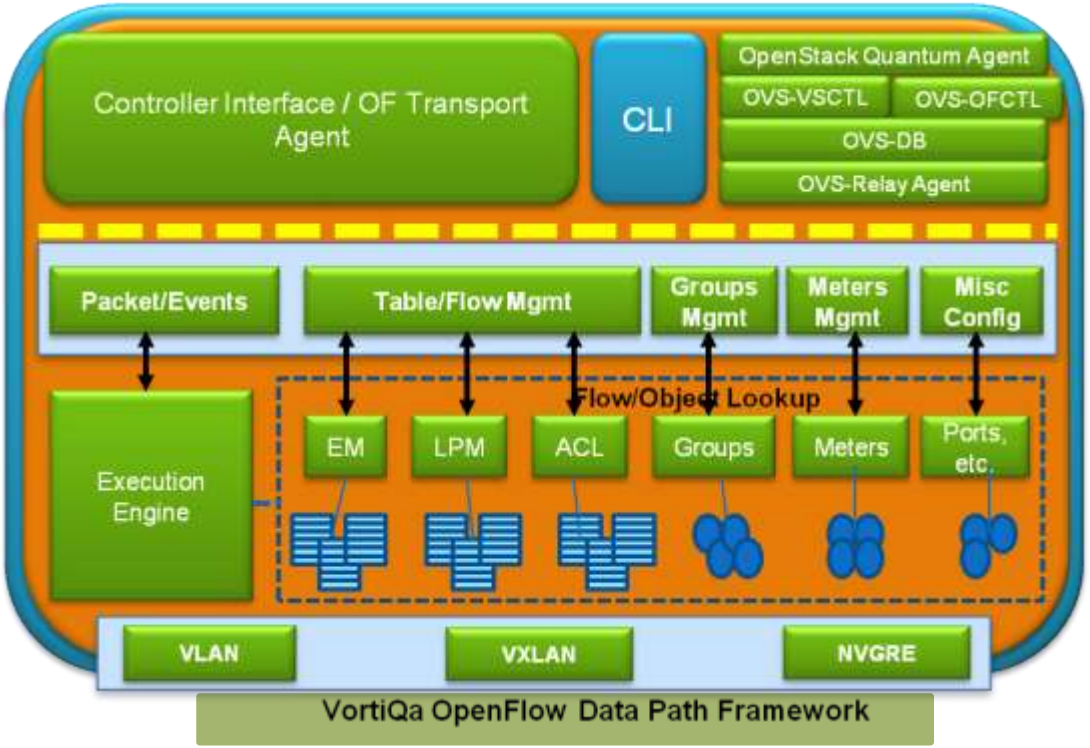


VortiQa ONSF Switch 1.0: Features

OpenFlow Data Path Support

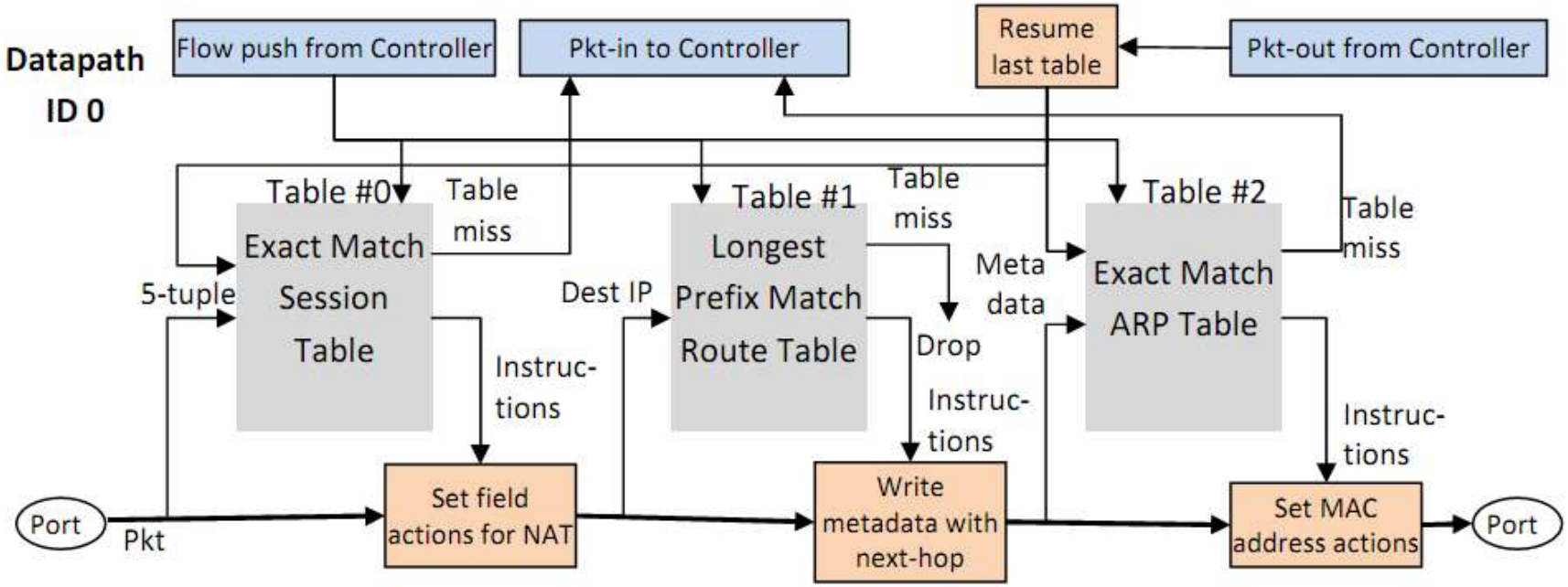
- Open Flow 1.3.x support
- Multiple Data Path instances
- Integration with OVS-DB
- Virtual Ports – VxLAN, etc.
- OpenStack Quantum integration

- Table Processing
  - Any number of tables per pipeline; custom extensions
  - Exact Match, LPM, ACL (RFC), DCFL
  - Flow indexing for fast flow search
  - Instruction / Action Extensions (L4-L7)
- Tags: MPLS, multiple MPLS, VLAN and multiple VLAN (QinQ)
- Groups, Meters, Queues object support
- Multipart messaging support including Tables features, Port Description
- Secure Transport Channel to Controller
- OpenStack Quantum Integration
- Auxiliary Connection support

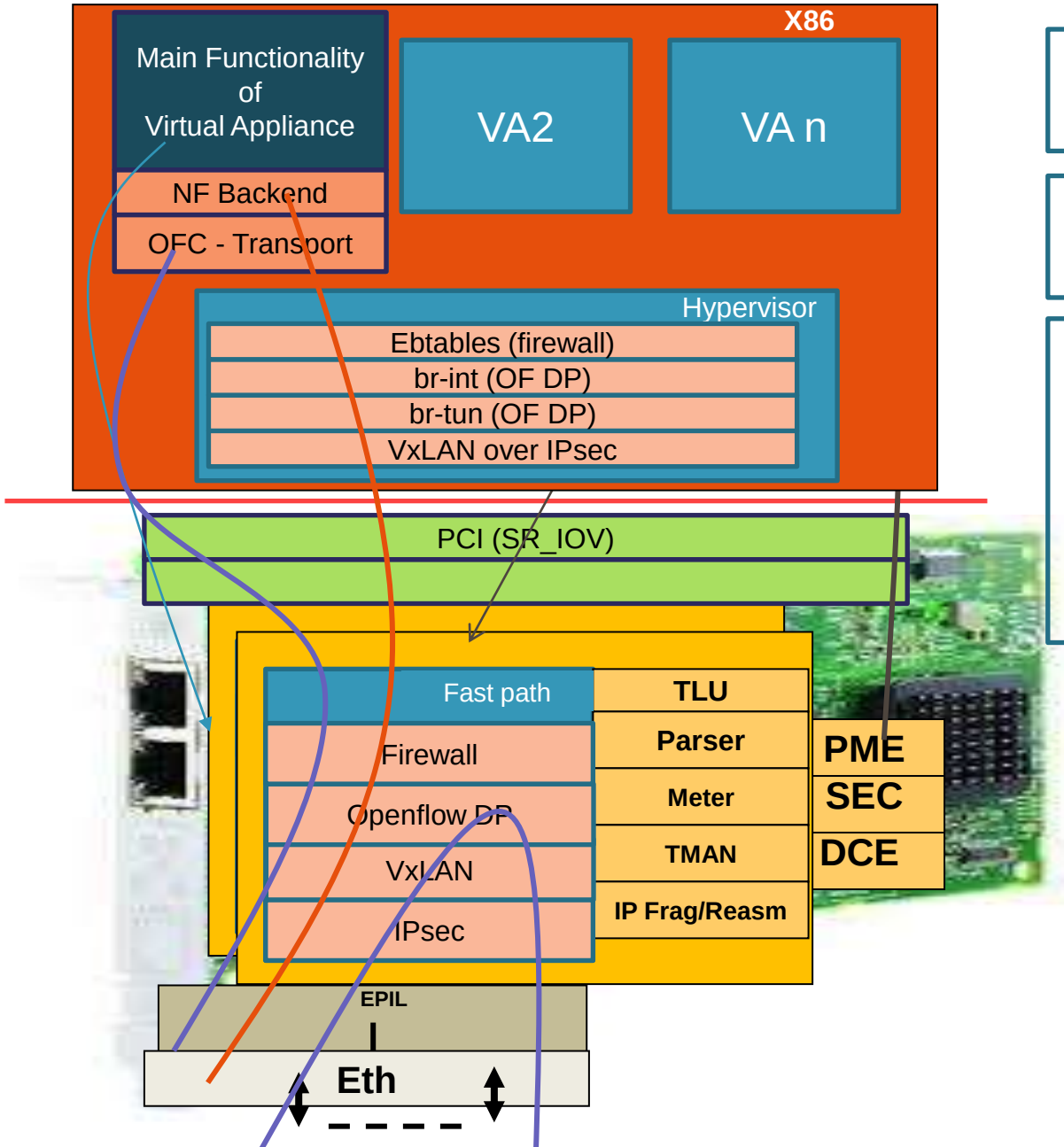


Freescale SDN Datapath Table Processing Diagram

- Most open source based SDN Switches support only L2 switching
- Freescale SDN switch intends to cover L4 and management
- Main features include SFW, NAT, ACL for router application
- Leverage DPAA datapath offload capability from FMan



Performance Optimization



Hypervisor FastPath

Partition of Accelerators
Direct Connectivity to VAs

Fast Path for VAs

IPv4/IPv6 Unicast forwarding
IPv4/IPv6 Multicast forwarding
IPv4/IPv6 Firewall
IPv4/IPv6 IPsec
IPv4/IPv6 QoS
GTP-U, PDCP, RoHC*
Openflow (for Offload)

Providing agility and elasticity with similar performance as in bare-metal appliances

T4240 QorIQ-Enabled VortiQa ON Switch

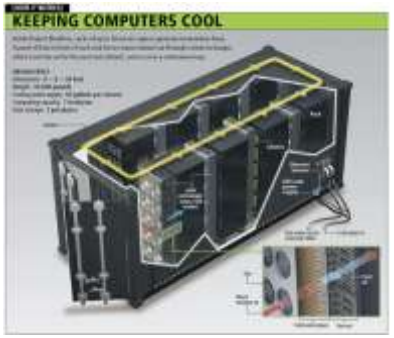
- Cryptography acceleration using SEC
- Complete Packet processing in Linux user space.
- Affinity to hardware cores/threads
- Egress hardware traffic conditioning and DCB support
- Ingress packet distribution to processes
  - Programmable hardware Parser for newer header detection (e.g. VxLAN)
  - Hardware Parse/Classify/Distribute on standard or proprietary header fields
  - Separate packet buffer pools per process (storage profiles)
- Faster table lookup using AltiVec

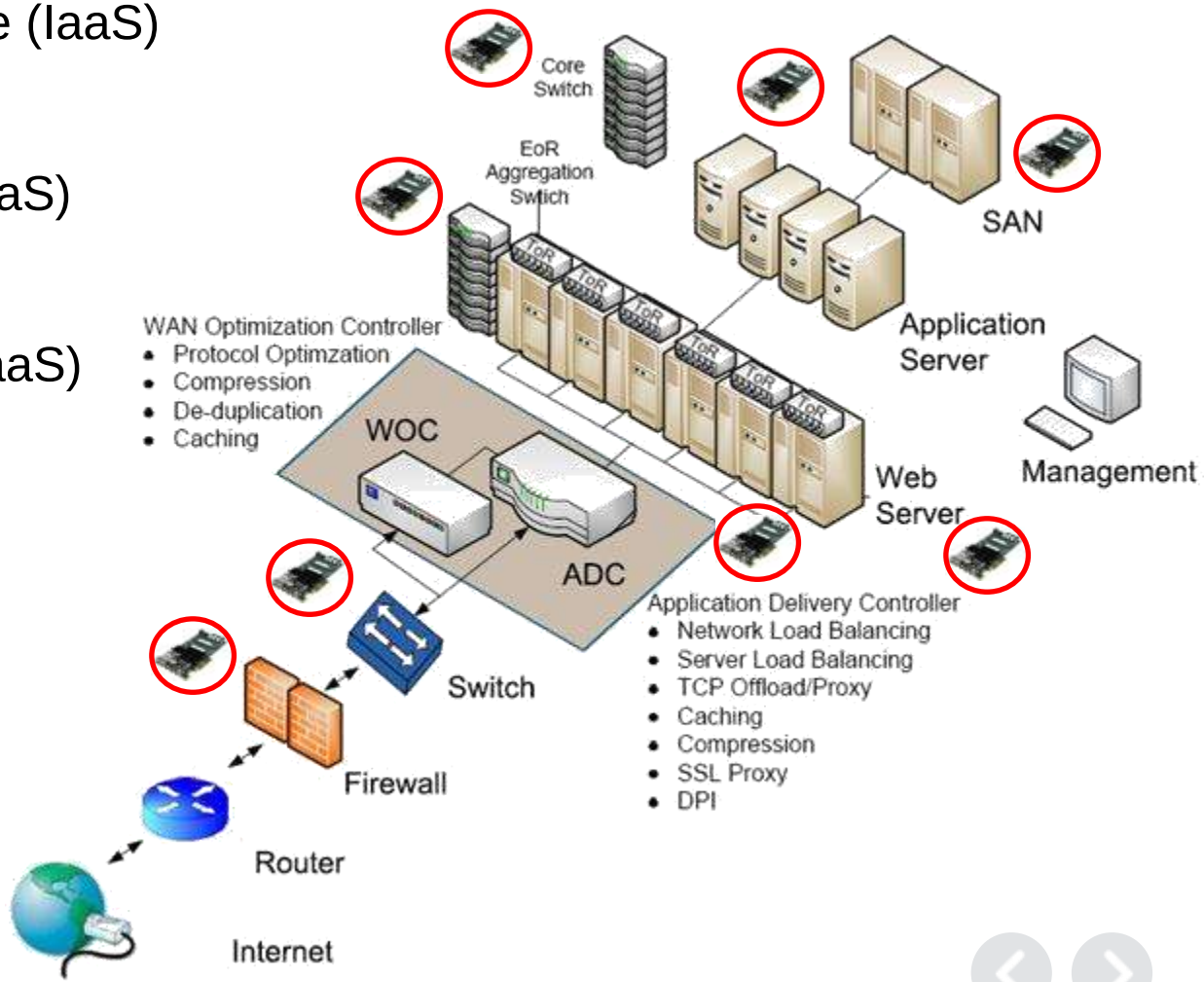
Intelligent Network Interface Controller (iNIC)

High Level Data Center Equipment Map

Data Center In-a-Box

- A central server administers the system, monitoring traffic and client demands
- Infrastructure as a Service (IaaS)
  - CRM
  - Mail Server, etc.
- Platform as a Service (PaaS)
  - Database Server
  - Web Server, etc
- Software as a Service (SaaS)
  - Load Balancer
  - storage server, etc.

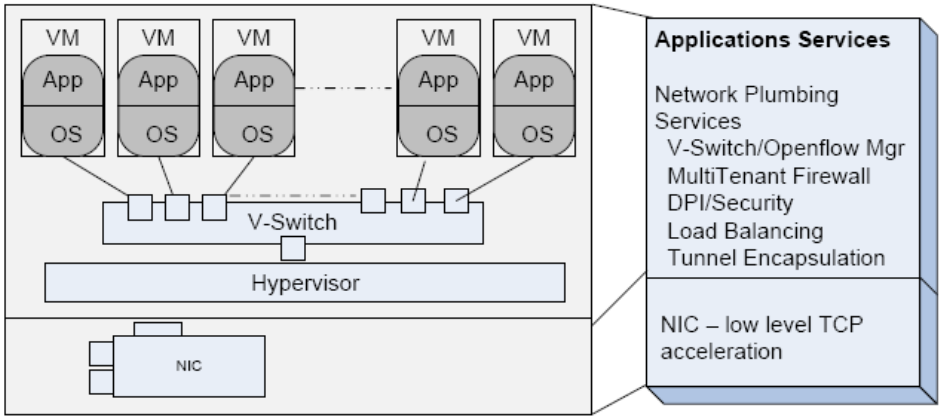




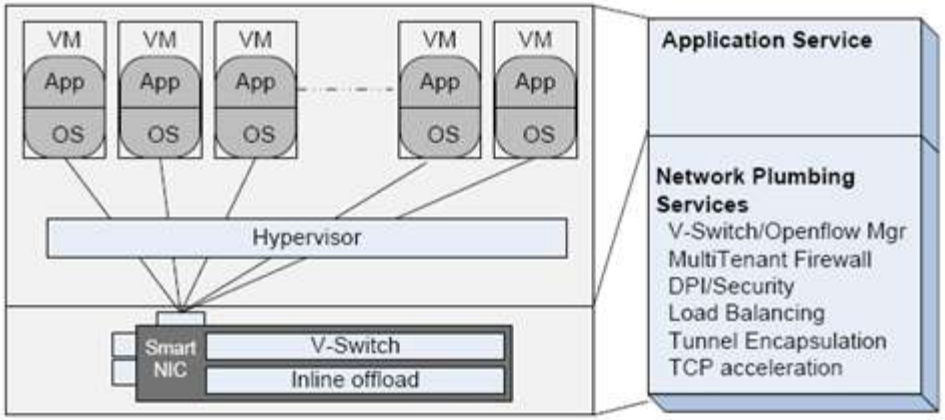
Networking Trend: More Performance, Less Power

- Moore's Law can't keep up with processing demands for exponentially increasing IP traffic
- Multicore processors need to balance number of cores with power consumption
- Need for scalability to build multiple products on a common architecture
- Reduce software complexity, improve productivity and speed implementation
- Network Virtualization is driving new Data Center architecture
- Multicore datapath adds flexibility and system-level performance advantage

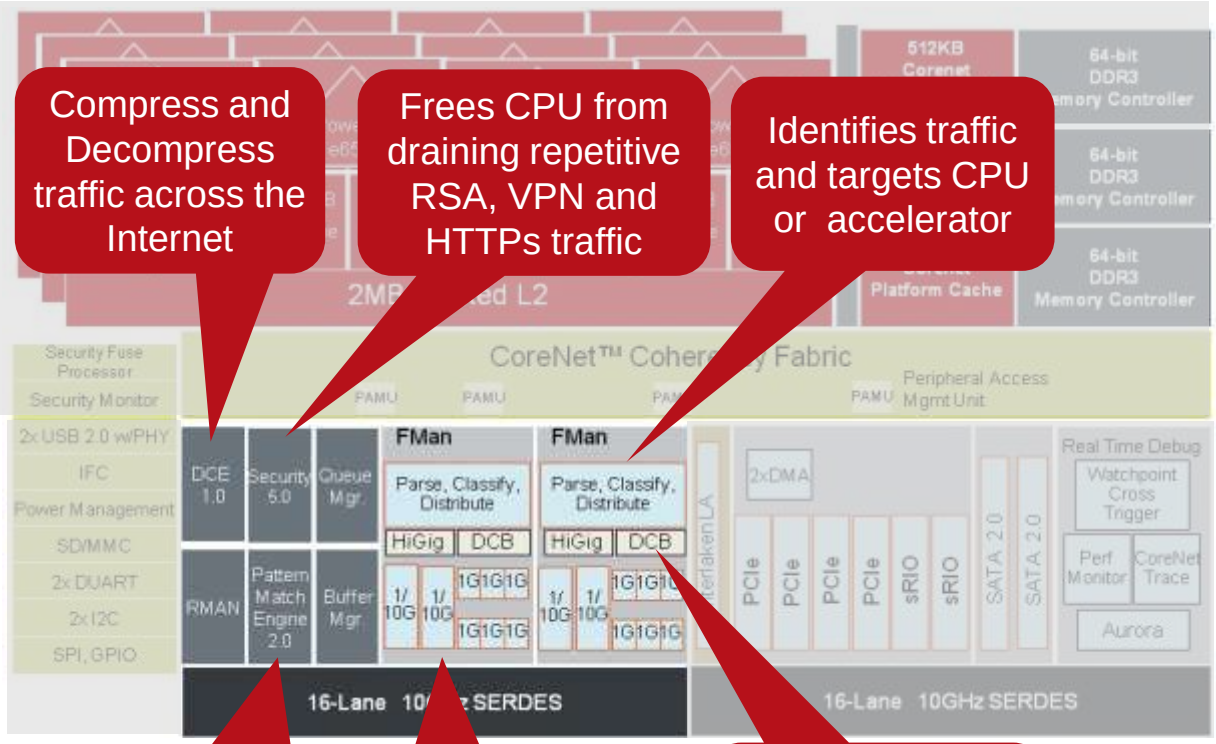
Simple NIC



QorIQ Datapath



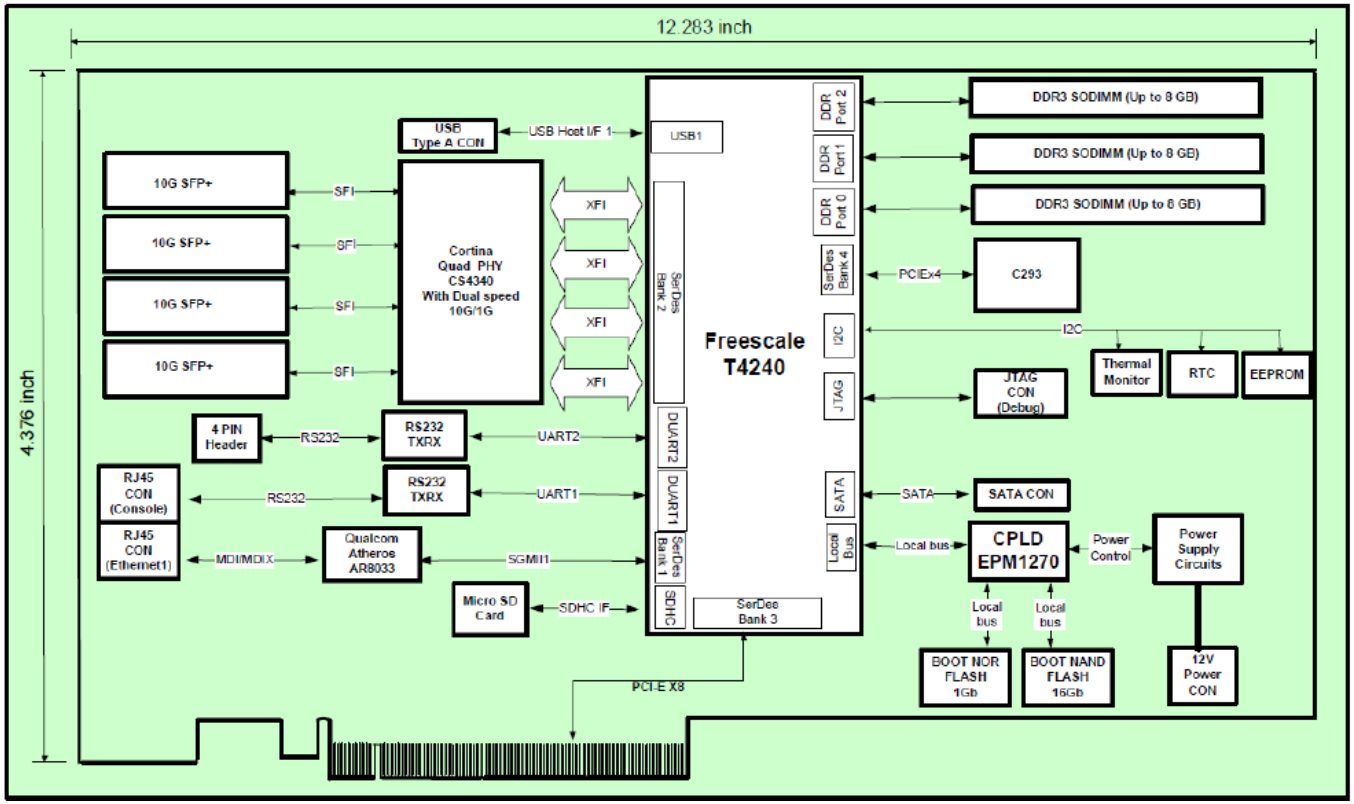
Enhancing Core Performance with Data Path Acceleration Architecture



Hardware Accelerators	
FMAN Frame Manager	50 Gbps aggregate Parse, Classify, Distribute
BMAN Buffer Manager	64 buffer pools
QMAN Queue Manager	Up to 2^{24} queues
RMAN Rapid IO Manager	Seamless mapping sRIO to DPAA
SEC Security	40Gbps: IPsec, SSL Public Key 25K/s 1024b RSA
PME Pattern Matching	10Gbps aggregate
DCE Data Compression	20Gbps aggregate
Saving CPU Cycles for higher value work	

New	Enhanced

T4240PCIe as Next-Generation Intelligent NIC (iNIC)



- Full size PCIe card
- T4240 Processor at 1.67GHz
- C293 Public Key acceleration
- 6GB DDR3 1867MT/s
- 4x 10G SFP+ cages
- x8 PCI Express Gen 2 EP
- x4 PCIe Express Gen 2 Root Complex
- 1Gb NOR and 1Gb NAND
- 2Gb Micro SD card
- USB Type A connector
- SATA connector
- JTAG connector
- 2x RS232 serial ports
- EEPROM
- Real Time Clock
- Avail Q1-14

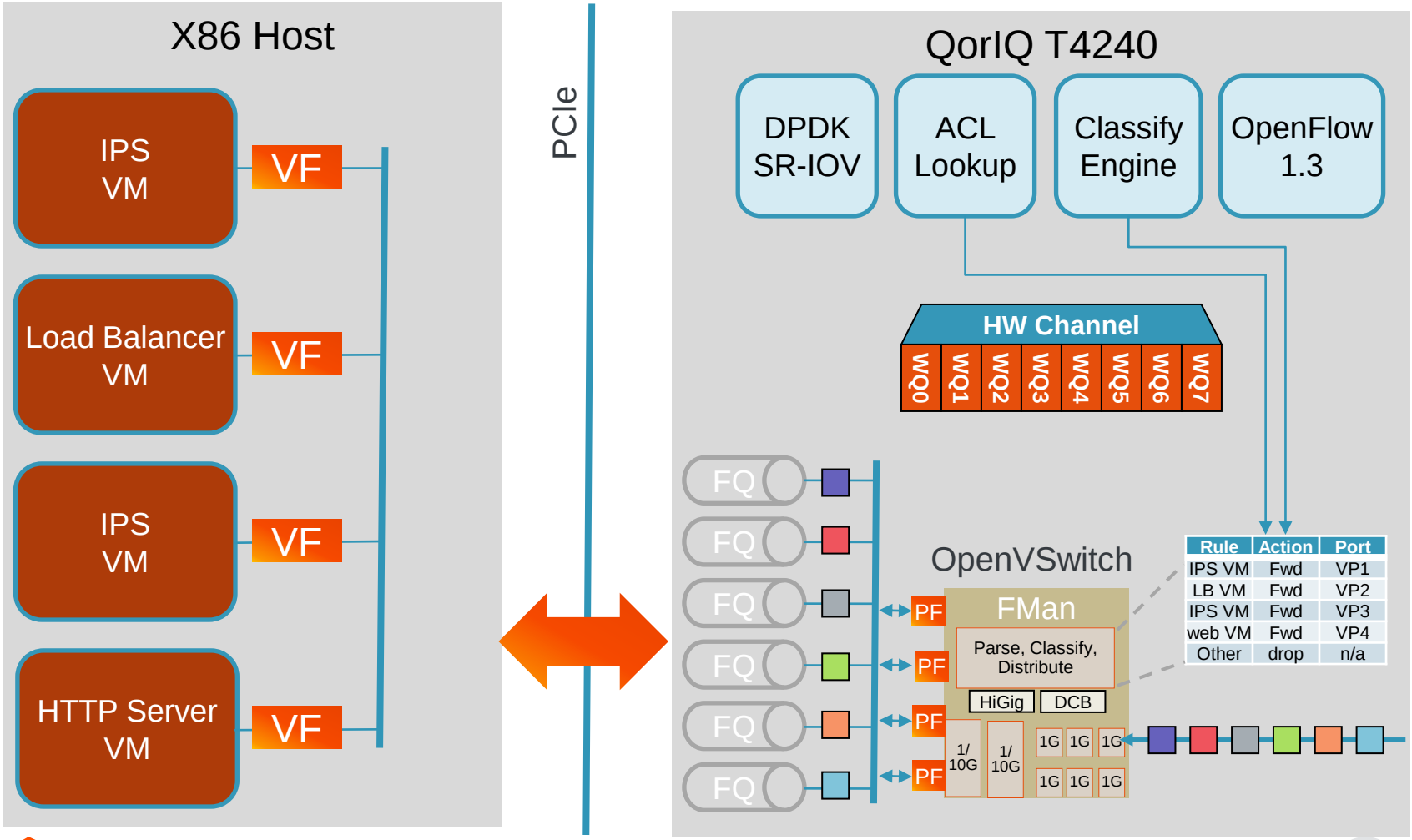
SR-IOV Support

- Single Root I/O Virtualization (SR-IOV) is a specification that allows a PCIe device to appear to be multiple separate physical PCIe devices
- SR-IOV works by introducing the idea of physical functions (PFs) and virtual functions (VFs)
  - Physical functions (PFs) are full-featured PCIe functions
  - Virtual functions (VFs) are “lightweight” functions that lack configuration resources
  - The PCI SIG SR-IOV specification indicates that each *device* can have up to 256 VFs

- QorIQ SR-IOV supports
  - PCI Express controller 1 supports end-point SR-IOV
  - Supports SR-IOV 1.1 spec version with 2 PFs and 64 VFs per PF (total of 128 VFs)
  - Each PF will have its own dedicated 8-Kbyte memory-mapped register space
  - Mapping of addresses into VF/PF space are through the ATMU translation

iNIC Using DPAA Accelerators

- Flow Classification with FMAN Classifier



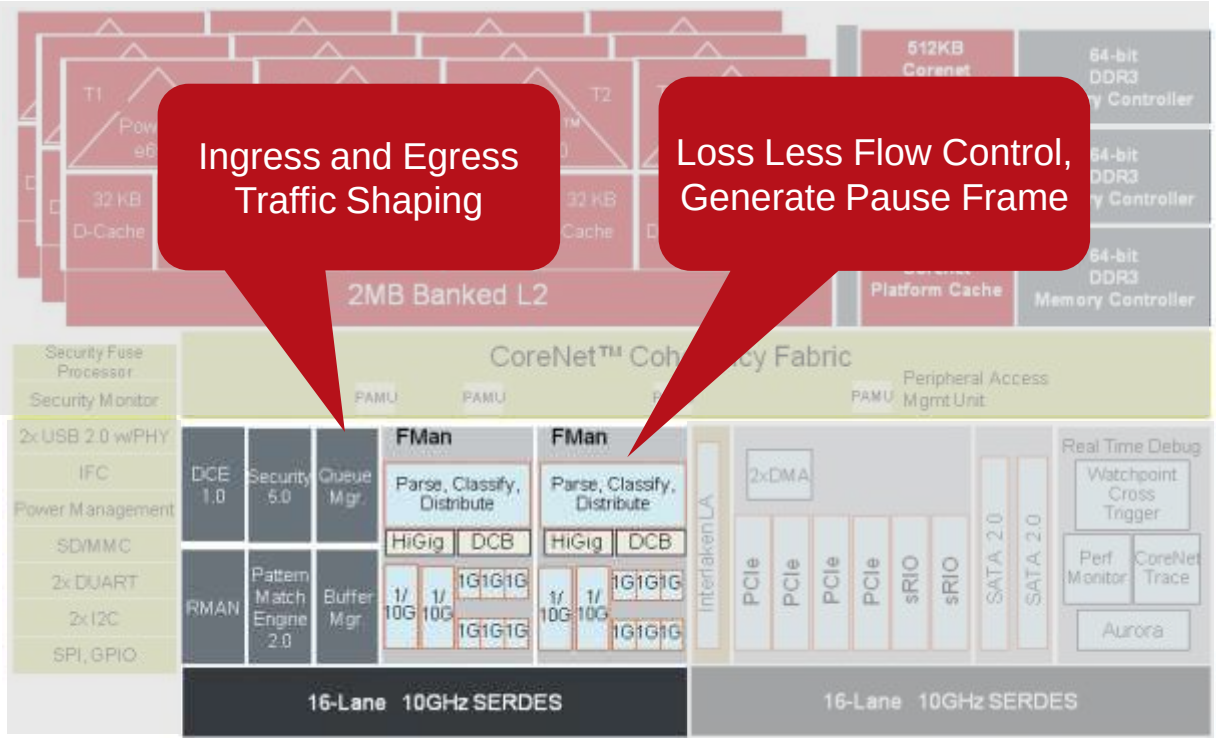
DPDK Compatibility

- Intel Data Plane Development Kit (Intel DPDK), a set of software libraries that can improve packet processing performance through the use of:
  - Memory Manager: A pool is created in huge page (2 MB/1 GB page) memory space and uses a ring to store free objects
  - Buffer Manager: pre-allocates fixed size buffers which are stored in memory pools
  - Queue Manager: instead of using spinlocks, implements safe lockless queues that allow different software components to process packets
  - Flow Classification: incorporates Intel Streaming SIMD Extensions to produce a hash based on tuple information so that packets can placed into flows
  - Poll Mode Drivers: The Intel DPDK includes Poll Modes which are designed to work without asynchronous, interrupt-based signaling mechanisms

- QorIQ is using Data Path Acceleration Architecture (DPAA) to implement the above functionality with hardware accelerators
  - SDK provided a shim layer to map the APIs

Data Center Server with DCB

Enhancing Performance with Data Path Acceleration Architecture

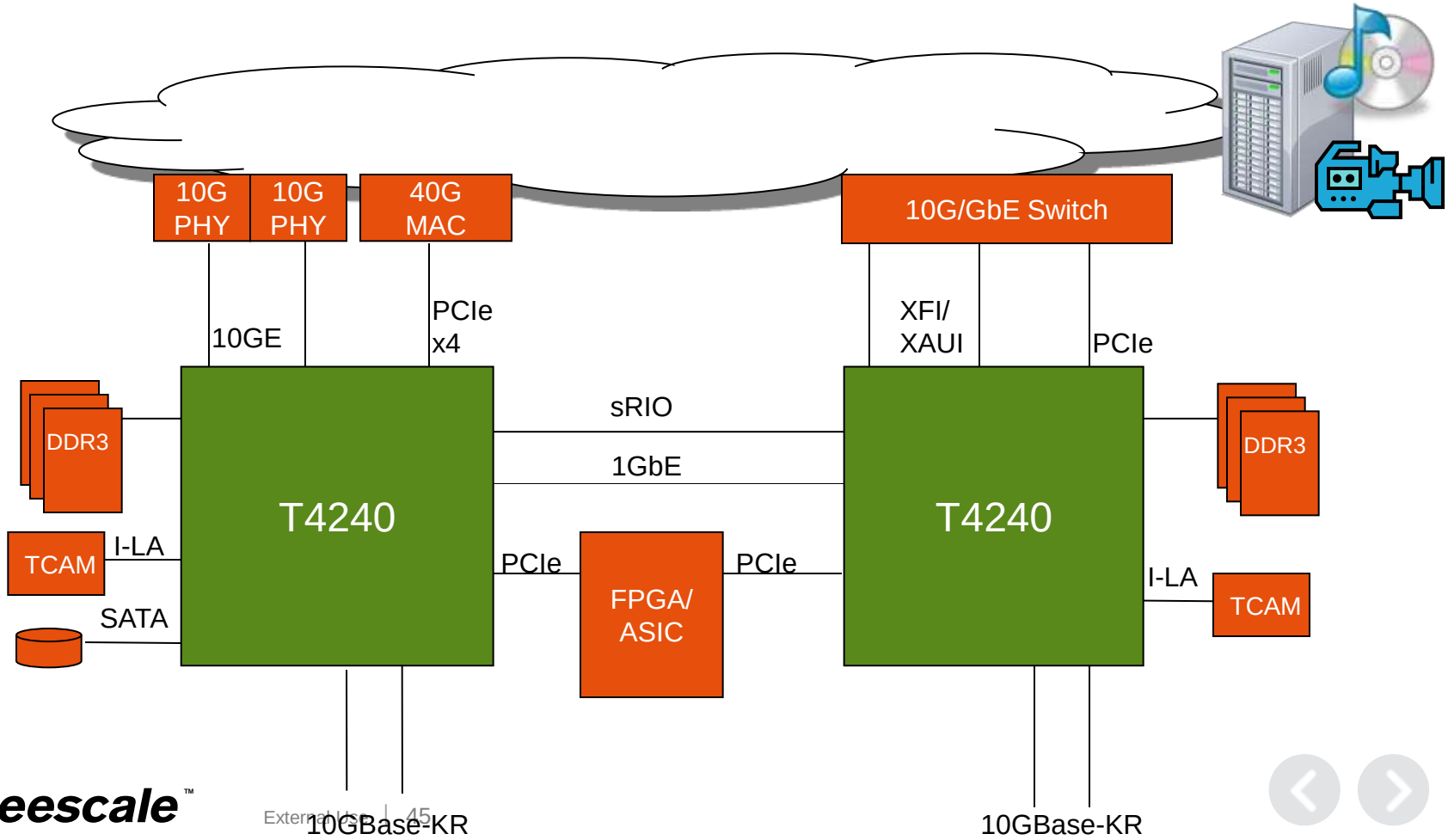


Hardware Accelerators	
FMAN Frame Manager	50 Gbps aggregate Parse, Classify, Distribute
BMAN Buffer Manager	64 buffer pools
QMAN Queue Manager	Up to 2^{24} queues
RMAN Rapid IO Manager	Seamless mapping sRIO to DPAA
SEC Security	40Gbps: IPsec, SSL Public Key 25K/s 1024b RSA
PME Pattern Matching	10Gbps aggregate
DCE Data Compression	20Gbps aggregate
Saving CPU Cycles for higher value work	

New	Enhanced

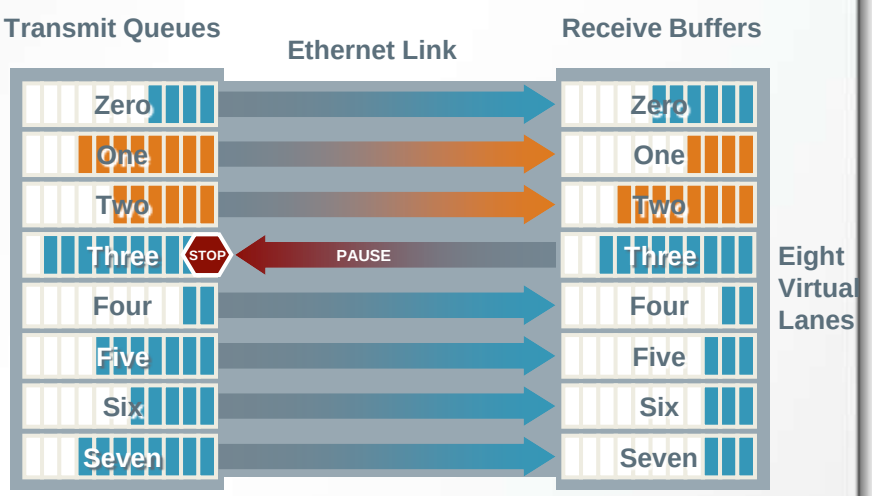
Network Appliance Blade Block diagram

- Network appliance connect to the cloud that offer Quality of Service base on subscription classes.



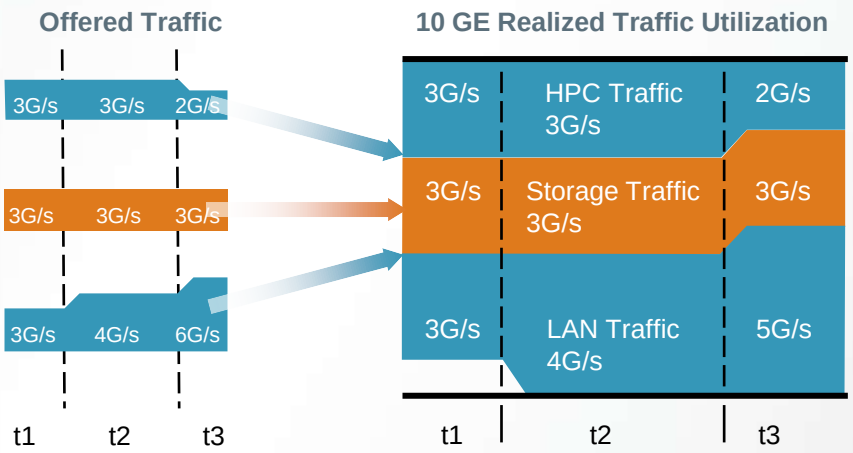
Data Center Ethernet: PFC and Bandwidth Management

Priority Flow Control



- *Enables lossless behavior for each class of service*
- *PAUSE sent per virtual lane when buffers limit exceeded*
- *IEEE 802.1Qbb*

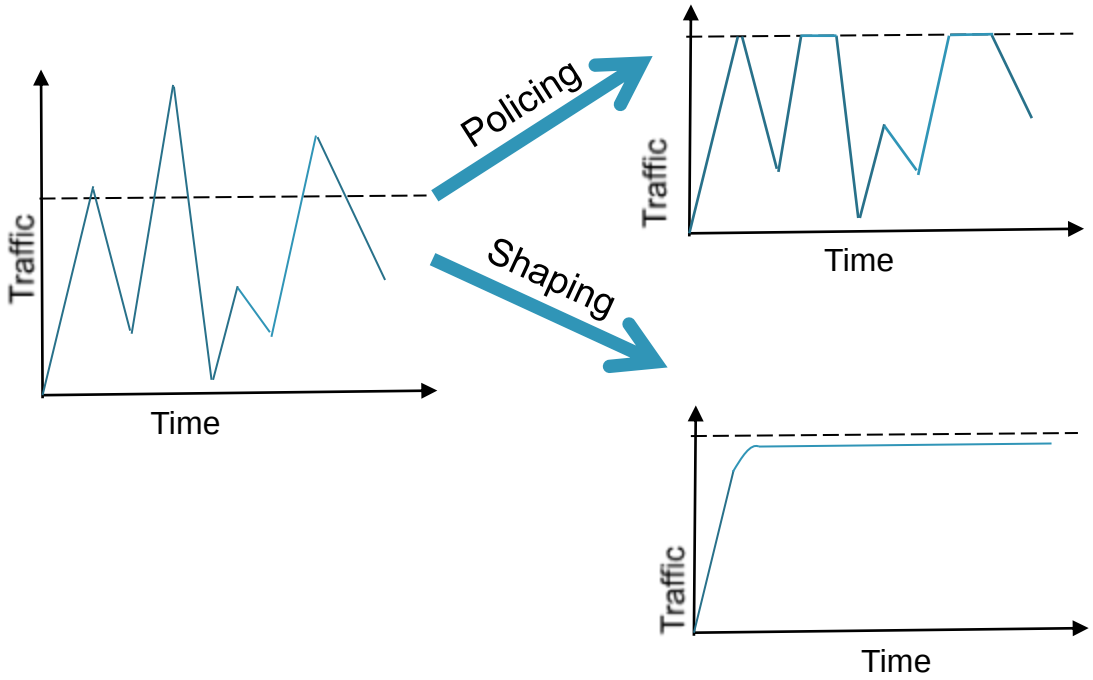
ETS CoS-based Bandwidth Management



- *Enables intelligent sharing of bandwidth between traffic classes control of bandwidth*
- *802.1Qaz*

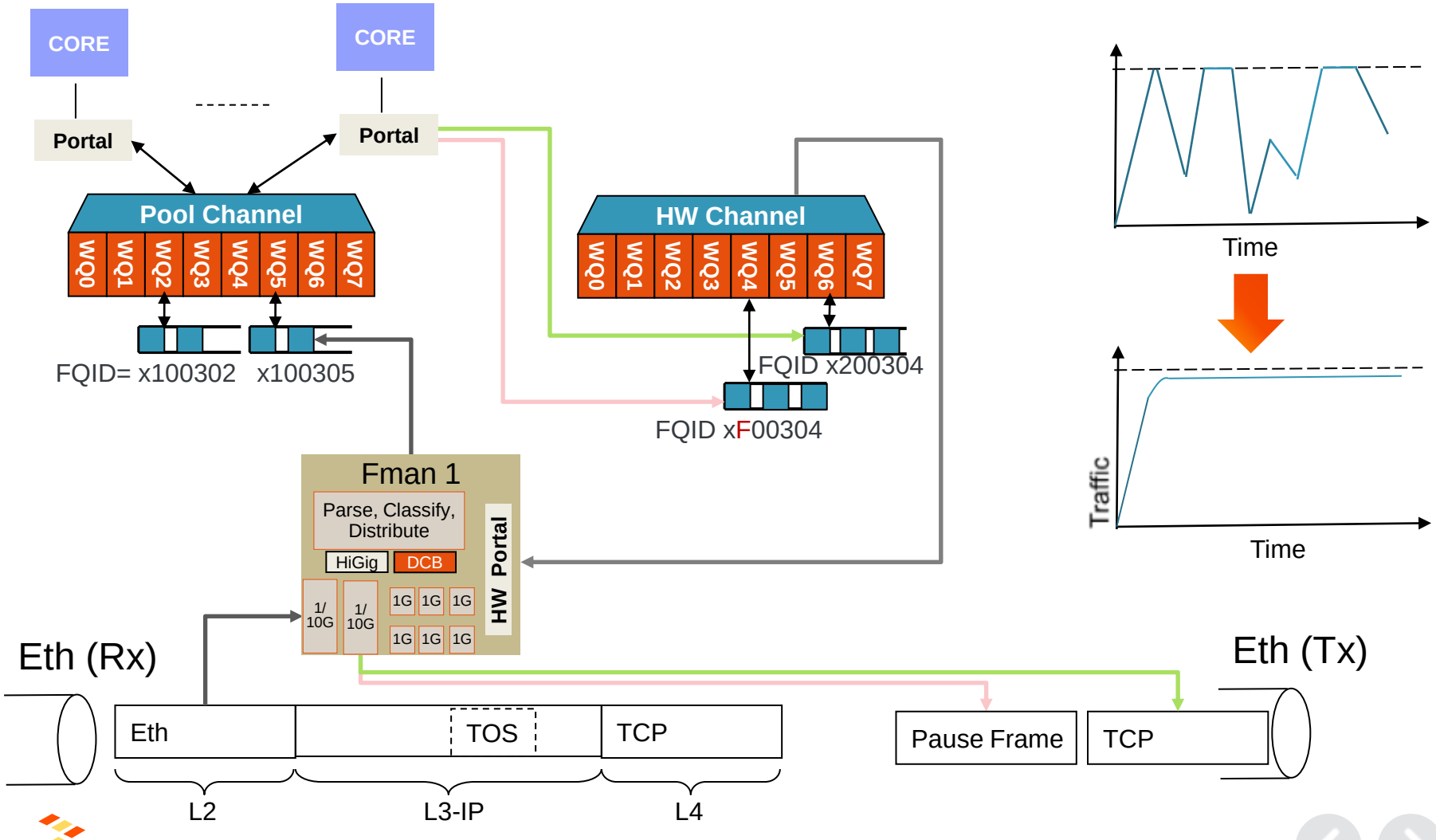
Policing and Shaping

- Policing puts a cap on the network usage and guarantee bandwidth
- Shaping smoothes out the egress traffic



Use Case: High-Level Application Mapping

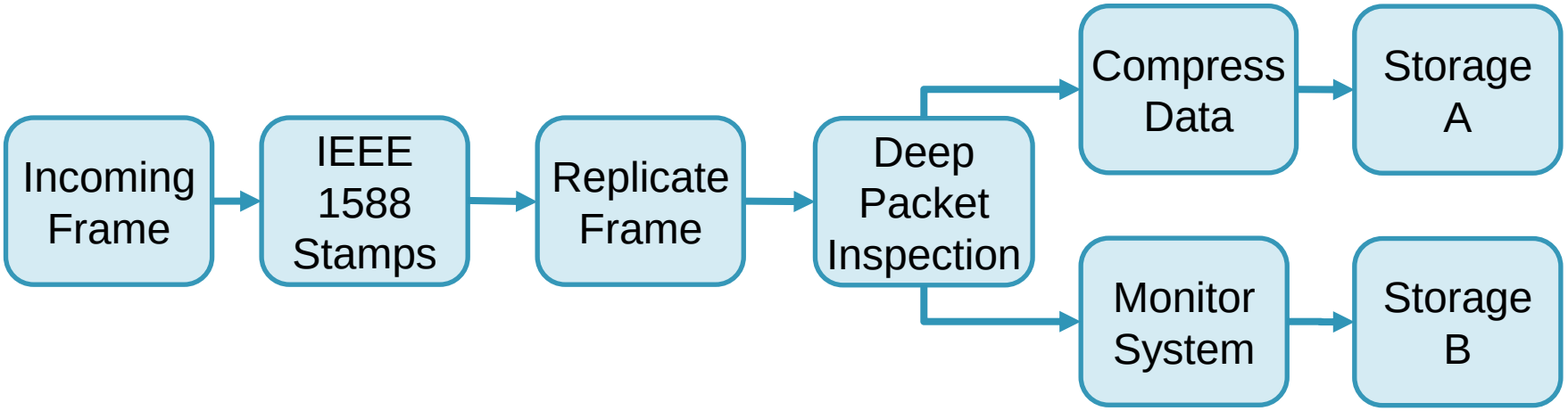
- Customer can decided to apply flow control or traffic shaping per flow/class



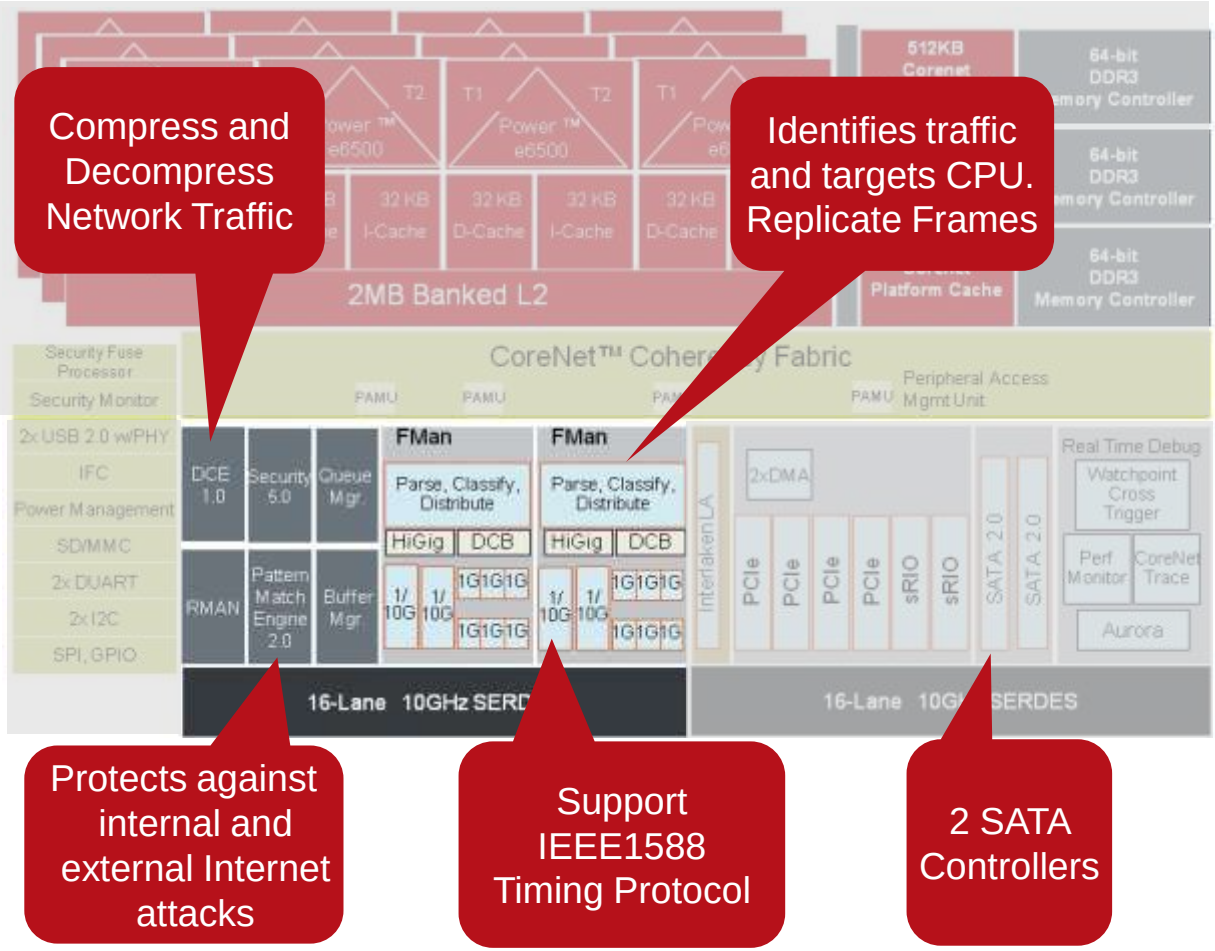
Smart Network Appliance: Data Replicator with DPAA Accelerator (FMAN/DCE/PME)

Smart Storage Using DPAA Accelerators

- Smart Storage Application with Compression and Deep Packet Inspection
- Payload inspection for Flow Classification
- Time stamp incoming packets for record keeping
- Replicate incoming traffic for forensic analysis



Enhancing Performance with Data Path Acceleration Architecture



Hardware Accelerators	
FMAN Frame Manager	50 Gbps aggregate Parse, Classify, Distribute
BMAN Buffer Manager	64 buffer pools
QMAN Queue Manager	Up to 2^{24} queues
RMAN Rapid IO Manager	Seamless mapping sRIO to DPAA
SEC Security	40Gbps: IPsec, SSL Public Key 25K/s 1024b RSA
PME Pattern Matching	10Gbps aggregate
DCE Data Compression	20Gbps aggregate
Saving CPU Cycles for higher value work	

New | Enhanced

New Frame Manager (FMan) Features

- FMan combines the Ethernet network interfaces with packet distribution logic to provide intelligent distribution and queuing decisions for incoming traffic at line rate.
- FMan key new features for QorIQ T4 processors
  - 1 Gbps/2.5Gbps/10Gbps
  - QMan interface: Supports priority based flow control message pass from Ethernet MAC to Qman
  - Comply with IEEE 803.3az (energy efficient Ethernet) and IEEE 802.1QBbb, in addition of IEEE Std 802.3, IEEE 802.3u, IEEE 802.3x, IEEE 802.3z, IEEE 802.3ac, IEEE 802.3ab, and IEEE-1588 v2 (clock synchronization over Ethernet)
  - Port Virtualization: Virtual Storage profile (SPID) selection after classification or distribution function evaluation
  - Rx port multicast support.
  - Offline port:
    - Able to dequeue and enqueue from/to a QMan queue
    - FMan (T series) is able to copy the frame into new buffers and enqueue back to the QMan
    - Use case: IP fragmentation and reassembly

Frame Manager BMI Features

- Storage Profile
  - Storage profile (including buffer pool allocation) for each received frame according to Rx port and frame length
  - Storage profile (including buffer allocation) for each received frame according to the results of classification (and frame length)
- Hardware assist for IEEE 1588-compliant timestamping
  - A high precision time measurement is provided by the FPM as a global utility to FMan modules which need a timestamp
  - Pass actual timestamp to host for received frames
  - Configurable pass of actual timestamp of transmitted frames to host
  - The IEEE1588 Timestamp (8 bytes) is written with the timestamp entry in the IC of the frame (if this feature is disabled in the MAC, the BMI writes a zero in this field)

Hardware Assist for IEEE 1588-Compliant Timestamping

- Support for IEEE 1588 can be done entirely in software running on a host CPU, but applications that require sub-10µSec accuracy need hardware support for accurate timestamping of incoming packets
- On Rx flow, the Ethernet MAC samples the 8 byte timestamp which is placed in the appropriate location in the Internal Context (IC). The user may configure the BMI to copy parts of the IC into a margin at the beginning of the first buffer of the frame. This is done by programming the FMBM_RICP register
- In this way, the timestamp is passed to the host CPU
- FPM Timestamp Register (FMFP_TSP) and Timestamp Fraction Register (FMFP_TSF)
  - The FPM timestamp register (FMFP_TSP) holds the timestamp integer value and the fraction value.

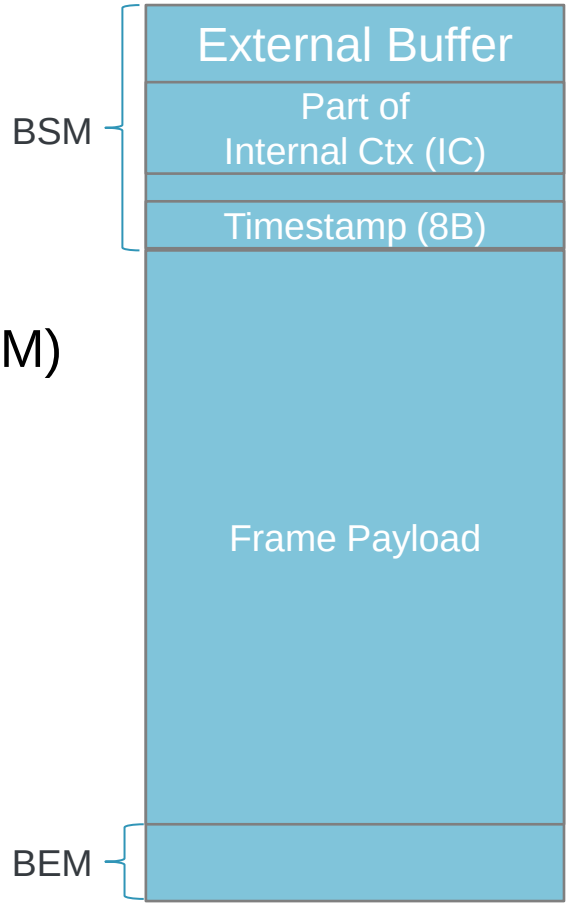
Internal Context

- The frame internal context (IC) is a data structure associated to every frame being processed
- For every new frame, the IC is automatically allocated in the FMan internal memory and is initialized with user-configurable, initial values

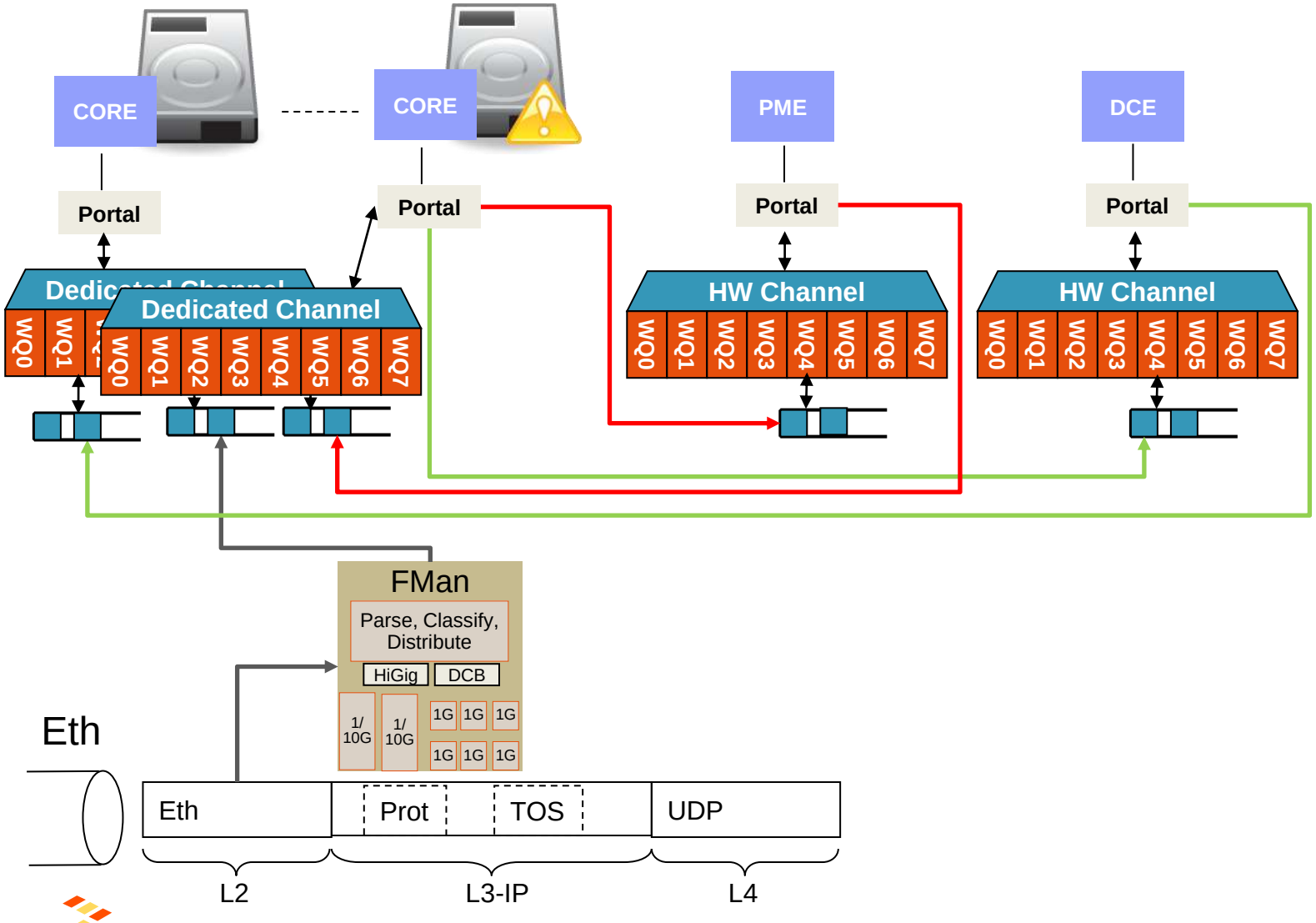
Offset	Size (B)	Name	Description
0x00	16	FD	Frame Descriptor (FD)
…			
0x40	8	Time Stamp	Rx: This is the timestamp captured by the Ethernet MAC (1G and 10G) when the frame is received. If the timestamp feature is disabled in the Ethernet MAC, this field is zeroed. Tx: this is the timestamp captured by the physical PHY when the frame is transmitted. If the timestamp feature is disabled in the Ethernet MAC, this field is zeroed
…			

Frame Buffer: Buffer Start Margin

- Frame is stored inside a Frame Buffer
  - Frame store after Buffer Start Margin (BSM)
  - Default BSM is 64B
  - Timestamp can be stored at offset 48B
  - There is no affect on the Buffer End Margin (BEM)



Use Case: High-Level Application Mapping



Summary

T4240 – Dense Processing For Demanding Applications

- Wireless infrastructure: control/transport, CRAN, RNC, EPC
- Microserver: high performance density
- Intelligent NIC: big data offload, SSL proxy, ADC, WOC
- Mil/Aero: 12 Altivec engines
- UTM: 40Gb/s crypto, 10Gb/s regex

- Highly efficient data path
- 2x better CoreMark/Watt than Xeon
- 4x 10GE integration – 1 chip solution compared to 4+ with Xeon
- SR-IOV with 128 VF for iNIC
- Datacenter bridging for lossless Ethernet
- Secure boot for IP protection

Other Sessions And Useful Information

- FTF2014 Sessions
  - FTF-NET-F0146_Introduction_to_DPAA
  - FTF-NET-F0070_QorIQ Platforms Trust Arch Overview
  - FTF-NET-F0157_QorIQ Platforms Trust Arch Demo & Deep Dive
  - FTF-SDS-F0028-Benchmarking Virtualization Solutions for QorIQ Processors
  - FTF-SDS-F0101 VortiQ ONSF
  - FTF-SDS-F0225_Vortiqa L1.pptx
  - FTF-SDS-F0016 - Software Defined networking (SDN) and IOT
  - FTF-SDS-F0218_Security_DN

- Hardware and Software Solution
  - T4240 Product Summary
    - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=T4240
  - VortiQa Open Network Director software
    - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=VORTIQA_OND
  - VortiQa Open Network Switch software
    - http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=VORTIQA_ONS

Introducing The QorIQ LS2 Family

Breakthrough, software-defined approach to advance the world's new virtualized networks

New, high-performance architecture built with ease-of-use in mind

Groundbreaking, flexible architecture that abstracts hardware complexity and enables customers to focus their resources on innovation at the application level

Optimized for software-defined networking applications

Balanced integration of CPU performance with network I/O and C-programmable datapath acceleration that is right-sized (power/performance/cost) to deliver advanced SoC technology for the SDN era

Extending the industry's broadest portfolio of 64-bit multicore SoCs

Built on the ARM® Cortex®-A57 architecture with integrated L2 switch enabling interconnect and peripherals to provide a complete system-on-chip solution

QorIQ LS2 Family

Key Features

SDN/NFV Switching

Data Center

Wireless Access

Unprecedented performance and ease of use for smarter, more capable networks

High performance cores with leading interconnect and memory bandwidth

- 8x ARM Cortex-A57 cores, 2.0GHz, 4MB L2 cache, w Neon SIMD
- 1MB L3 platform cache w/ECC
- 2x 64b DDR4 up to 2.4GT/s

A high performance datapath designed with software developers in mind

- New datapath hardware and abstracted acceleration that is called via standard Linux objects
- 40 Gbps Packet processing performance with 20Gbps acceleration (crypto, Pattern Match/RegEx, Data Compression)
- Management complex provides all init/setup/teardown tasks

Leading network I/O integration

- 8x1/10GbE + 8x1G, MACSec on up to 4x 1/10GbE
- Integrated L2 switching capability for cost savings
- 4 PCIe Gen3 controllers, 1 with SR-IOV support
- 2 x SATA 3.0, 2 x USB 3.0 with PHY

See the LS2 Family First in the Tech Lab!

4 new demos built on QorIQ LS2 processors:

- Performance Analysis Made Easy
- Leave the Packet Processing To Us
- Combining Ease of Use with Performance
- Tools for Every Step of Your Design

www.Freescale.com

© 2014 Freescale Semiconductor, Inc. | ***External Use***