Integrating Rack Level Connectivity into a PCIe Switch

Hot Chips 2013

Silicon without Software is Just Sand

Confidential

Authors and Acknowledgments

- Architecture Team
  - Jack Regula
  - Mani Subramaniyan
  - Jeff Dodson
- Executive Team
- Engineering Team

The Opportunity

- PCIe is to servers as Ethernet is to the rest of the world
  - It is used to connect everything
  - Therefore extend PCIe for use as a converged rack level interconnect for IPC as well as for I/O
- Questions
  - Where best to bridge to Ethernet?
  - How to share expensive endpoints (SSD)?
  - How to architect modular compute platform where CPU blade is just processor and memory
- Benefits of PCIe as a rack level fabric are:
  - Saves cost, space, and power within the blades by removing NICs, HBAs, and disk drives from them
  - Low latency, high throughput IPC at low cost/power/area

The State of the Competition Within the Rack

- Competition is Data Center Ethernet and IB
  - 10GigE LOM per server moving to 40G and 100G uplinks
  - QDR (40 Gbps) IB now available
- PCIe Gen 3 at x8 has 1.6x QDR IB link speed w/o needing HCAs

- Existing PCIe switches and fabrics:
  - Non-transparent bridging is problematic because of the change in programming model
  - Don't support redundant links well
  - Limited scalability due to limit of 256 BUS numbers
  - Marginal security
  - But do serve to prove the concept

Challenges

- Retaining PCIe compatibility
  - Only 100% standard PCIe packets on edge links
    - Existing CPU chipsets and endpoints can be used
  - Use Vendor Defined extension mechanisms on fabric links
    - Vendor Defined Message
    - Vendor Specific DLLP
    - Vendor Defined End to End Prefix
- Not changing the programming model
  - BIOS, drivers, OS, applications
- Non-technical challenges:
  - Added features must be marginal increase in switch cost or product isn't economically viable
    - PLX has been shipping 96 lane PCIe Gen3 switch for >9 mos.
  - Normal product development schedule and budget constraints must be met

Architectural Goals

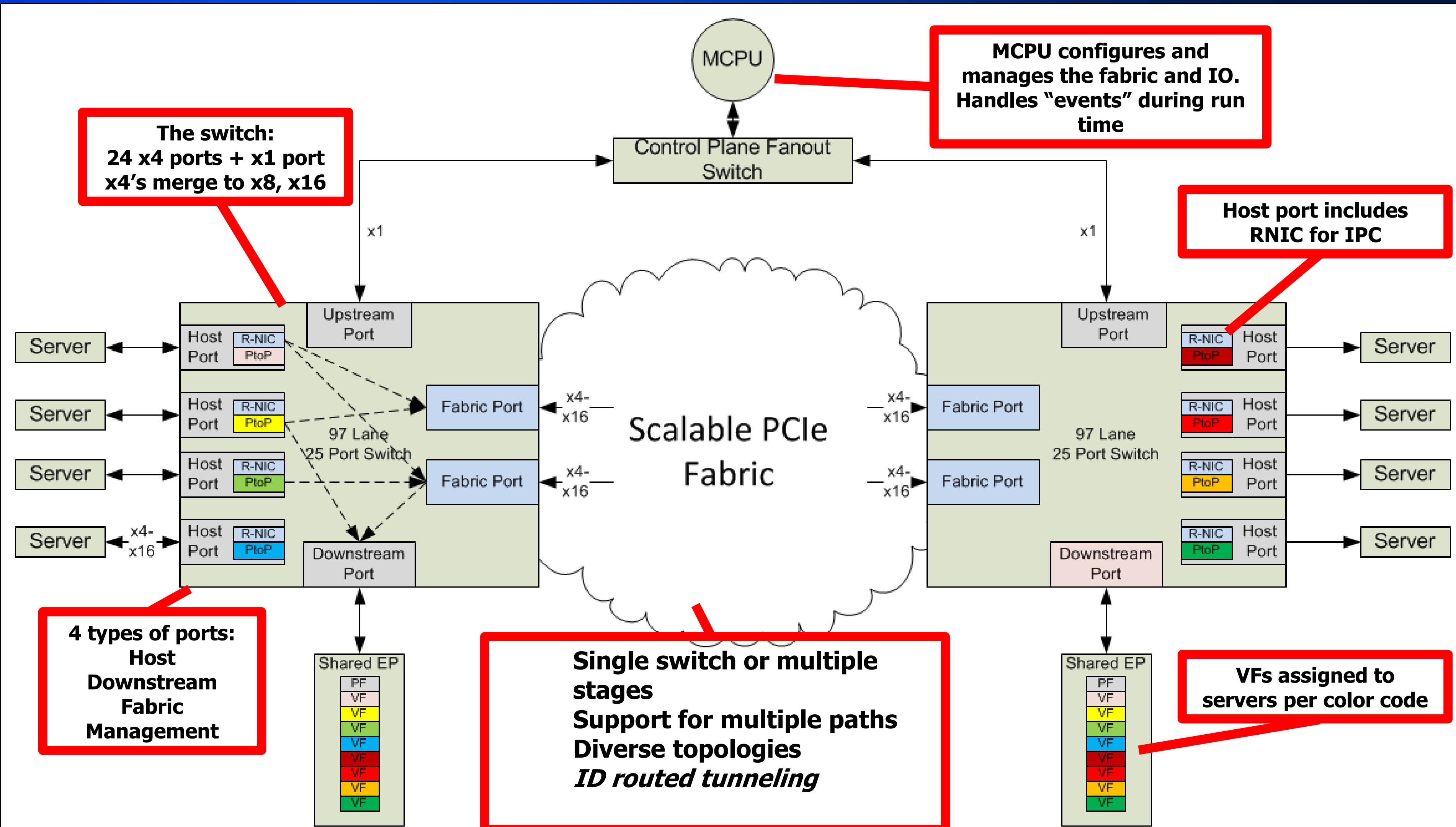
- The ability to share storage and communication end points among a large number of processors without driver changes
- High BW, low latency IPC using both Ethernet tunneling and RDMA over PCIe
- Remove PCIe scalability limits
- Support for diverse multi-stage fabric topologies
  - Load balancing over redundant paths
  - Congestion avoidance and management
  - Resiliency
- Complete compatibility with the PCIe specification
  - Use only standard PCIe TLPs on edge links
  - Use these plus PCIe Vendor Defined extension mechanisms on fabric links

System Solution Outline

Tell story via builds on next slide

- The Switch
  - 24 x4 ports plus an x1 port,
  - x4 ports combine to from wider ports
  - 4 types of ports:
    - Management port
    - Host ports that connect to servers
    - Fabric ports that connect to other switches
    - Downstream ports that connect to endpoints
- The Fabric
  - Single switch or multiple stages
  - Diverse topologies: Fat Tree, Mesh, Torus
  - Sweet spot = 3-5 stage fat tree >= rack scale
- The Management CPU
  - Manages and configures all so that standard server software (drivers, OS, applications) work w/o change
  - Handles only “events” during run time

System Block Diagram



ID Routing Solves Multiple Problems

- Scalability:
  - An 8-bit Domain ID added to the PCIe ID to created a 24-bit Global ID.
  - Each Domain is a separate PCIe BUS number space
  - Fabric scales to (65K-256) nodes, each with 256 FUNs
- Producer/Consumer Ordering over redundant paths
  - In PCIe, memory requests route by address while completions route by ID but stay ordered because there is only one path
  - In ExpressFabric™, both types route by ID. Same choice of redundant paths is made so ordering is maintained. Different ordered streams take different paths so all paths are used.
- ID Routing simplifies and in some cases eliminates need for mappings between global and host-local address spaces
- PCIe Vendor Defined End to End Prefix is added when packet type is not natively ID-routed or destination is in a different Domain

Congestion Avoided via Pull Protocol

- Pull protocol
  - Read completions of pull protocol can take any path, as can other unordered traffic
  - Unordered traffic is round robin spread over redundant paths absent congestion feedback (>70% of IPC traffic)
  - Avoids paths where congestion is indicated
  - Work Request and Remote Read Requests Outstanding limits function as end to end sliding windows flow control and manage congestion on source and destination host links, respectively
- BECN using VD DLLP used to balance loads on redundant paths
- Credit based flow control for DMA WR VDMs to avoid deadlock

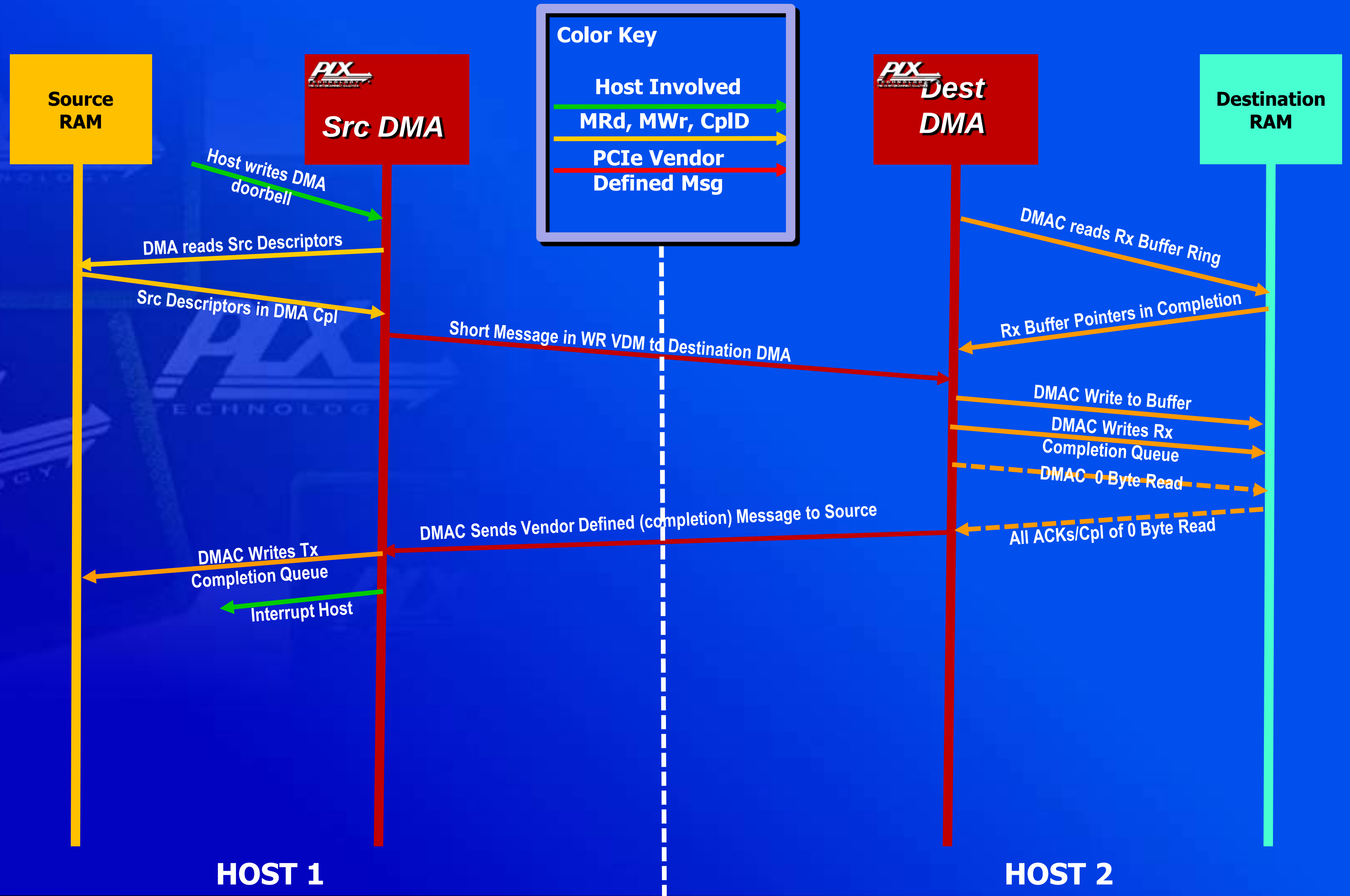
QoS

- TC Queuing on output queued switch
  - 4 TC Qs on x4, 8 on x8 or x16
  - Mix of priority and WRR scheduling
  - Interoperates and provides benefits with standard CPUs and endpoints as link partners, not just on fabric links

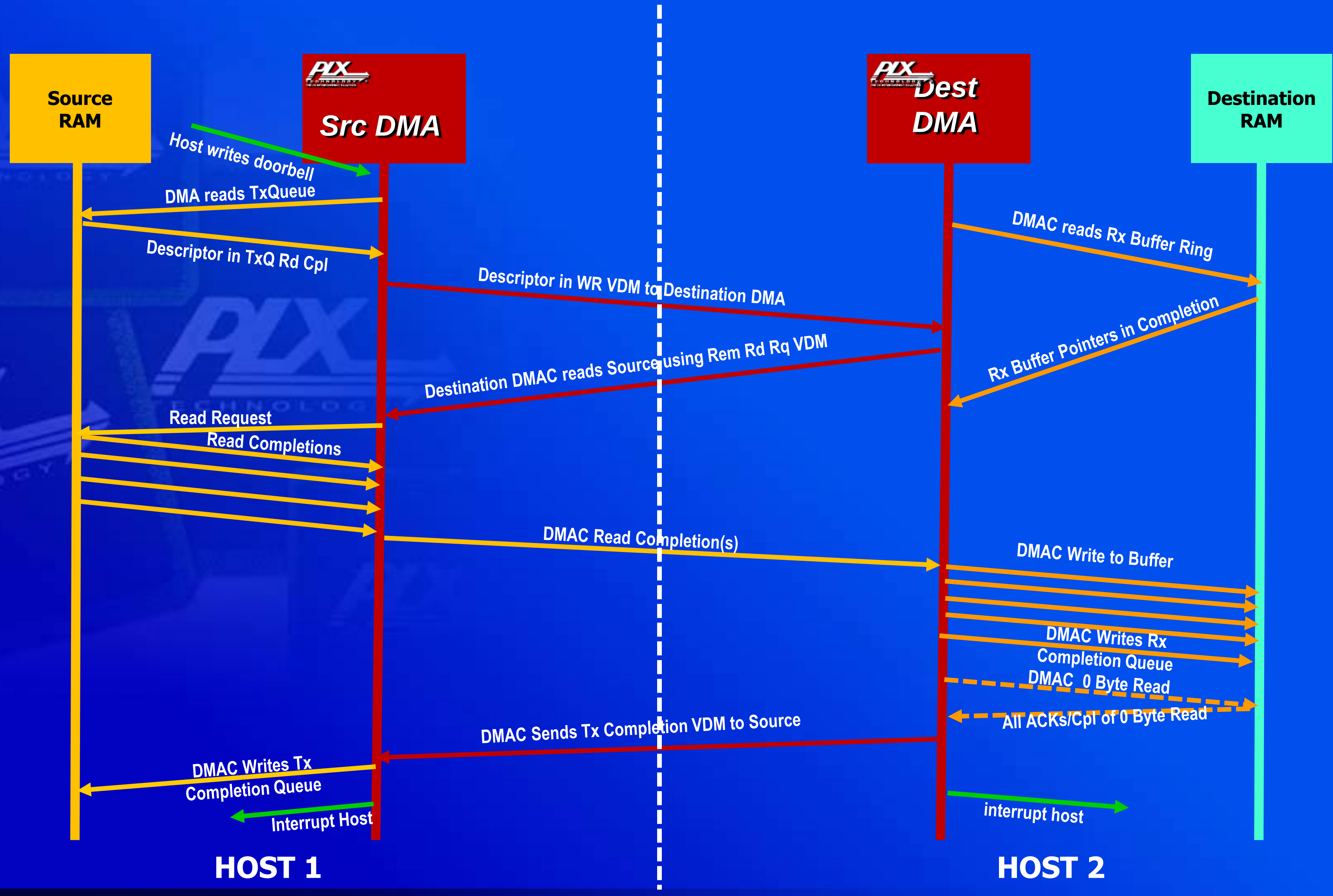
Integrated Messaging Engine

- Single HW messaging engine per 4_x4_port module of the switch
  - Virtualized and presented to hosts as multiple VFs
  - Each VF is an RDMA-NIC and can be assigned to an SI on the host
- Transfers can be done in two modes:
    - NIC mode tunnels Ethernet via standard TCP/IP stack
    - RDMA mode uses OFED stack and is a secure and reliable zero copy transfer from source application buffer to destination application buffer
    - Transfers use PCIe Vendor Defined Messages on fabric links
- Short packets are pushed; long packets are pulled
    - Congestion avoidance benefits vs 2nd pass thru fabric

Short Packet Push



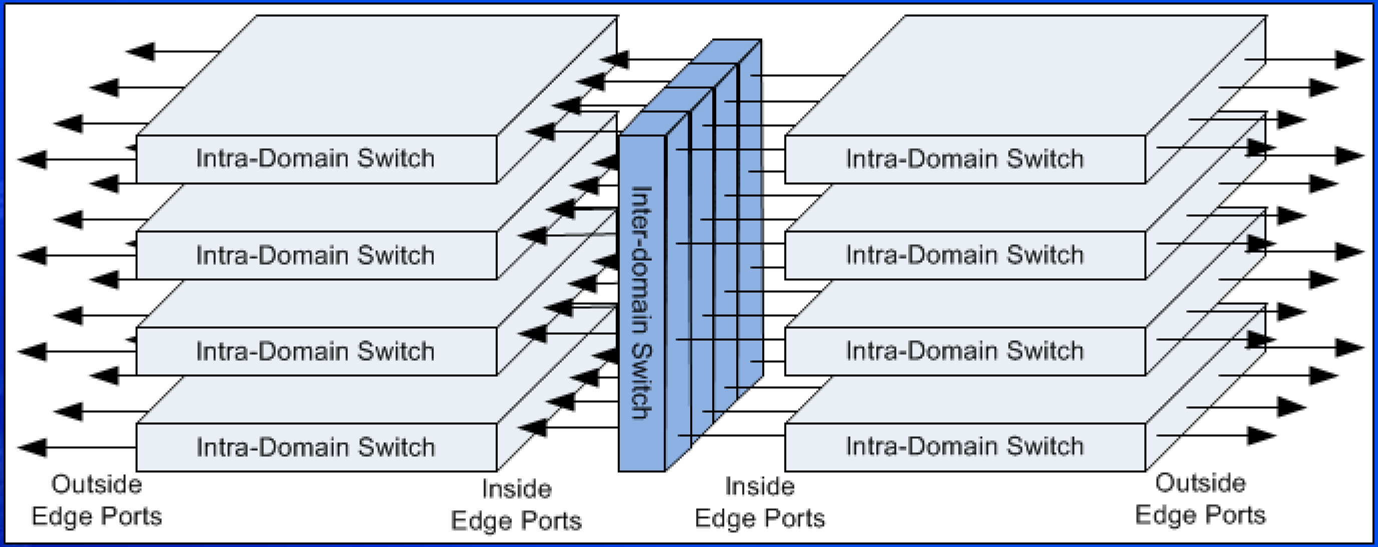
DMA Pull (NIC)


Source RAM
Src DMA
Dest DMA
Destination RAM
Host writes doorbell
DMA reads TxQueue
Descriptor in TxQ Rd Cpl
DMAC reads Rx Buffer Ring
Descriptor in WR VDM to Destination DMA
Rx Buffer Pointers in Completion
Destination DMAC reads Source using Rem Rd Rq VDM
Read Request
Read Completions
DMAC Read Completion(s)
DMAC Write to Buffer
DMAC Writes Rx Completion Queue
DMAC 0 Byte Read
All ACKs/Cpl of 0 Byte Read
DMAC Sends Tx Completion VDM to Source
DMAC Writes Tx Completion Queue
Interrupt Host
interrupt host
HOST 1
HOST 2

Fabric Topologies

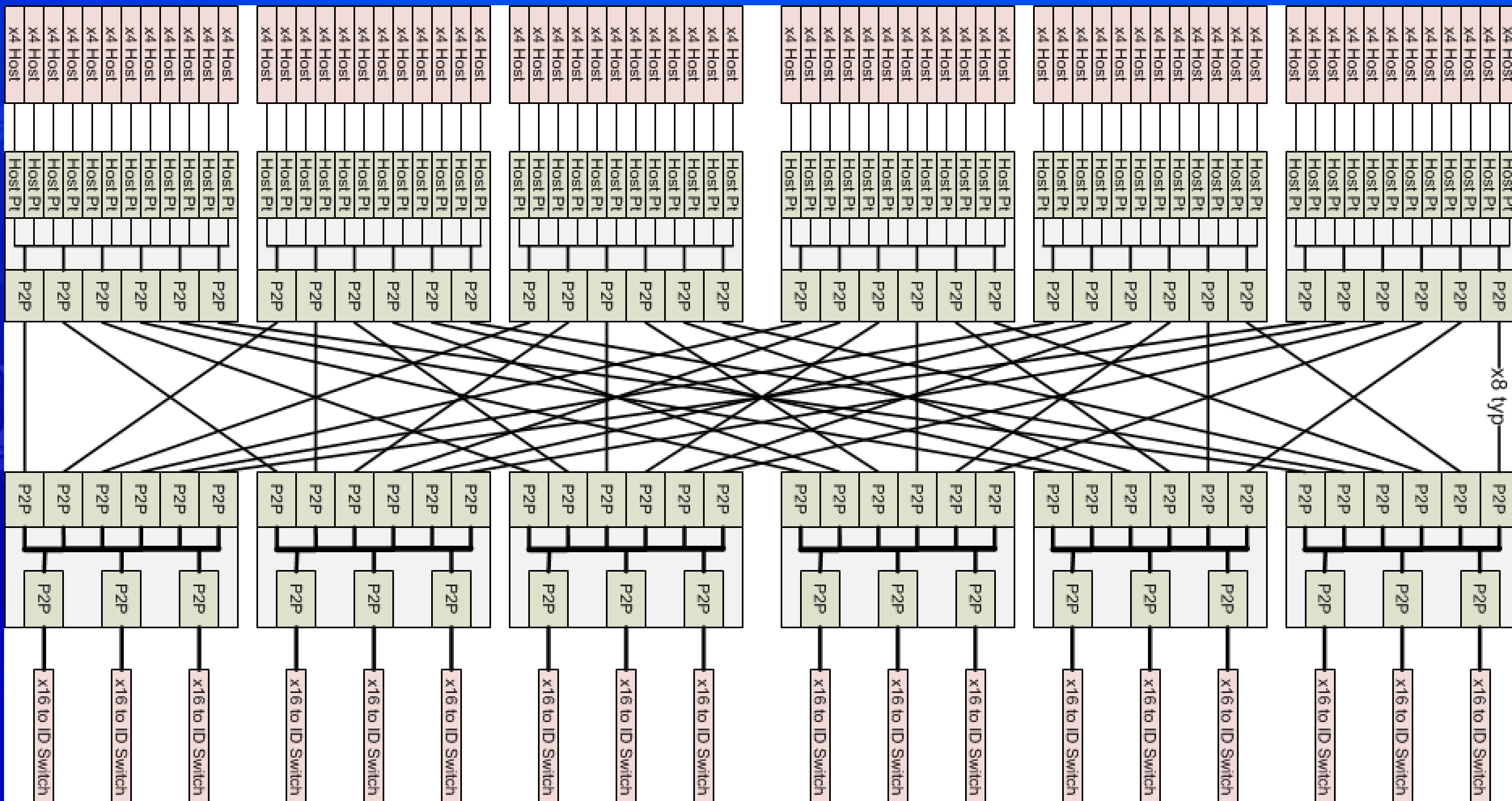
- Support for multi-stage switch fabrics includes
  - Routing and load balancing over redundant paths
  - Diverse topology support
    - Fat tree
    - 3D fat tree
    - Mesh
    - 2D and 3D Torus
- 3D Fat tree
  - Illustrating Domains
    - Each stands alone with its own MCPU
    - Share I/O within Domain
    - Communicate host to host across Domains
  - Align domain boundaries with packaging boundaries
    - Domain per cabinet
    - Domain per rack

3D Fat Tree

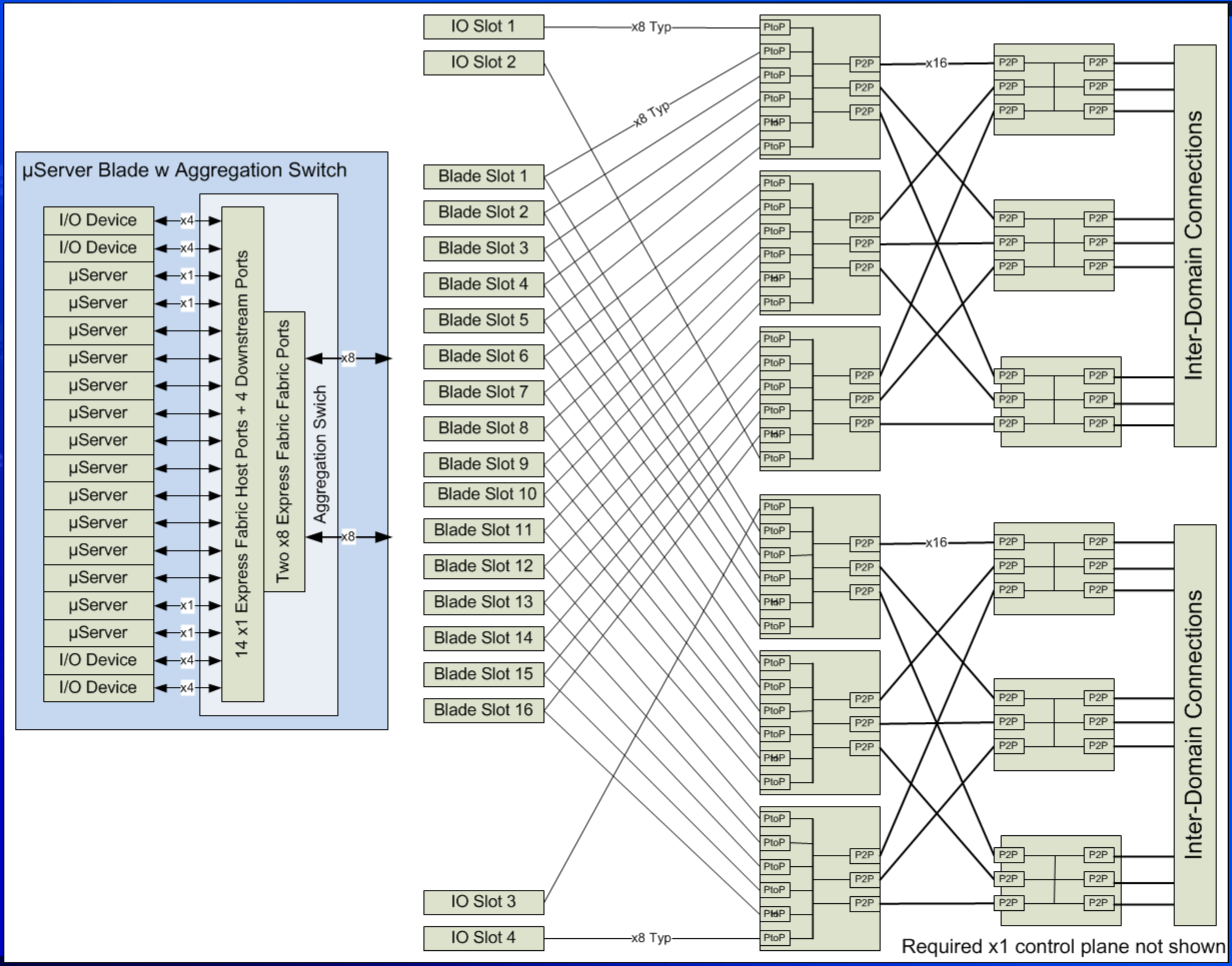


- Each grey box and its outside edge nodes is a standalone Domain & BUS# space w MCPU
- Domain fabric derived from 2n+1 stage fat tree by deleting n+1 columns of switches:
  - 3 stages => 1 stage; 5 stages => 2 stages
  - The former central rank forms inside edge rank
- Each Inter-domain switch can be a single chip or multiple stage fabric

Domain with 72 x4 Ports



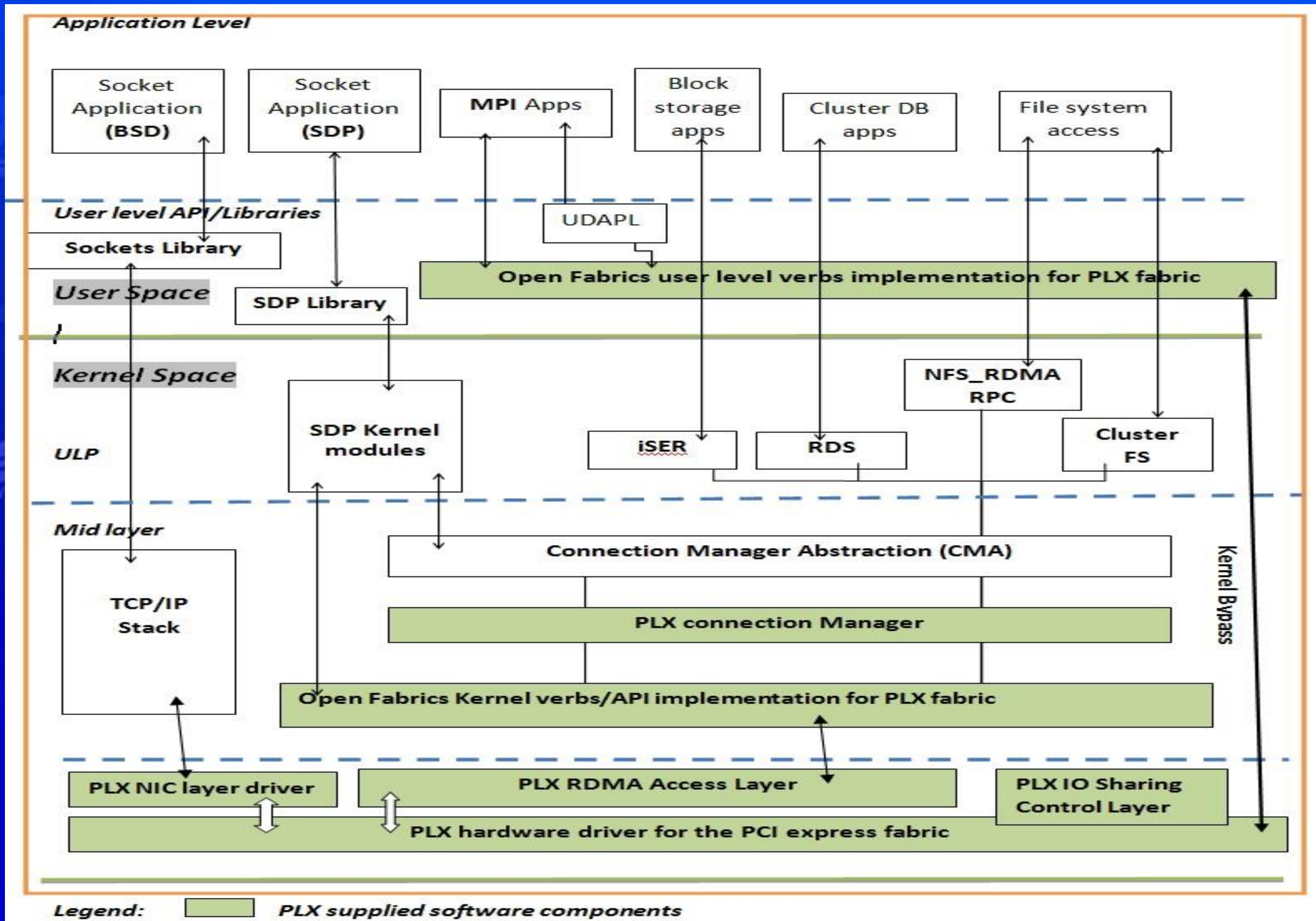
µServer Aggregation Fabric



Overcoming the SW Challenge

- Management CPU sleight of hand
  - MCPU manipulates fabric and IO configuration space to match standard enterprise IO programming model
  - MCPU configures mapping and routing tables in switches transparently to host software
  - Fabric and host ports are hidden from MCPU OS and BIOS so management application can manage them directly
- Leverage the existing SW infrastructure
  - TCP/IP Stack for Ethernet Tunneling
  - OFED Stack for RDMA
  - Supporting applications that use the APIs that feed these stacks
- PLX developed SW (SW layer diagram next slide)
  - Management application
  - Host to host drivers and adaptation layers

Host SW Layer Diagram

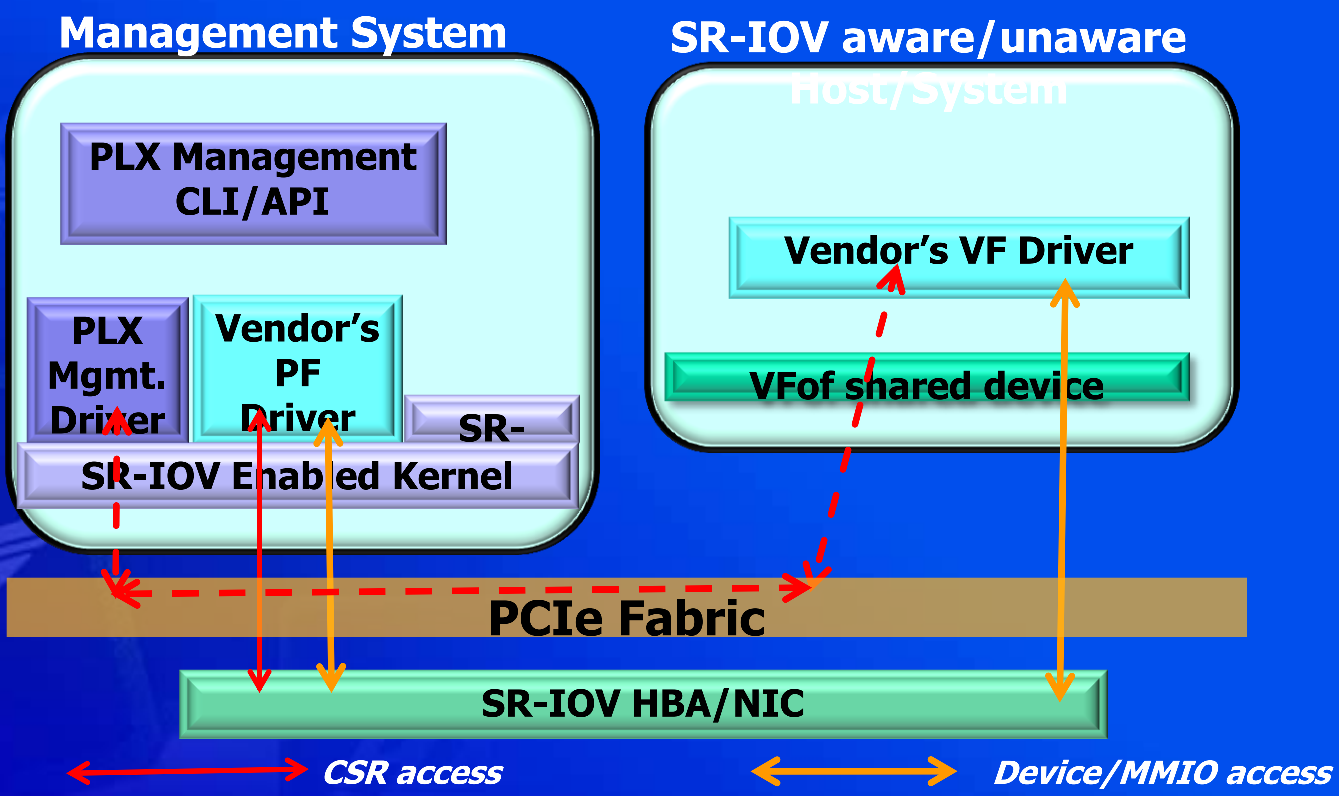


ExpressIOV™

- Sharing of I/O by the assignment of the VFs of SR-IOV endpoints, functions of multi function endpoint, and single function endpoints connected to the fabric to hosts within the same fabric Domain

- Applications:
  - Shared SSD within fabric scale cluster
  - Shared NICs or CNAs to connect ExpressFabric hosts into the “cloud”
  - Fabrics in which GPUs are dynamically assigned to hosts and communicate with hosts and each other either directly in memory space or via RDMA

Model for Sharing an SR-IOV VF



- MCPU configures ID-routed tunnels between hosts and the I/O VFs assigned to them
- Vendors' PF drivers run on MCPU
- Vendors' VF drivers run on hosts

Performance Benchmark

- Throughput
  - Results from an IPERF benchmark on 2, 8 core Xeon servers connected back to back, Linux RHEL 5.6
  - Gen 2 x8 and Gen3 x4 using memory copy DMA

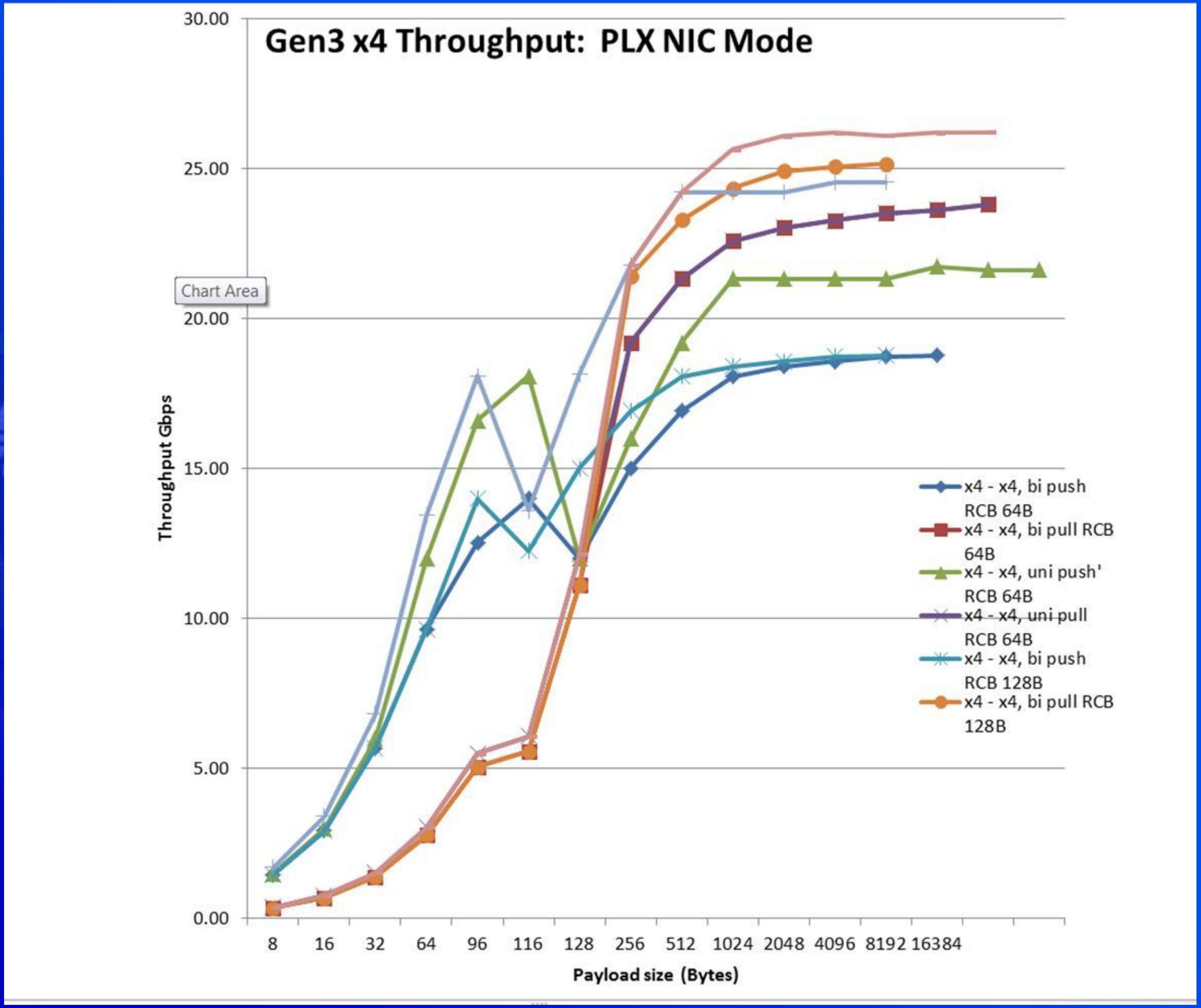
	PLX NIC software on PLX NT	10Gbe Server Adapter
Throughput	22 Gb/sec	9.8 Gb/sec
Receive CPU % overhead	11%	11.5%

- Total Latency (including HW & SW)
  - 25 µsec PCIe
  - 30 µsec on 10GbE server adapter

System HW Latency Estimation

DMA and TWC Latency Comparison vs Number of Fabric Stages								
	Number Fabric Stages					Number Fabric Stages		
TWC PIO Write	1	3	5		**Short Packet Push DMA**	1	3	5
Posted write thru fabric	150	450	750		DMA doorbell write	150	150	150
Land in memory	77	77	77		DMA reads TxQ	230	230	230
Total (ns)	227	527	827		WR VDM through fabric	0	300	600
					Dest DMA writes to RxQ, RxCQ	150	150	150
					RxCQ Interrupt Received	150	150	150
					Total (ns)	680	980	1280
	Number Fabric Stages					Number Fabric Stages		
TWC PIO Read	1	3	5		**NIC Mode Pull DMA**	1	3	5
forward path	150	450	750		DMA doorbell write	150	150	150
Read of host/RC memory	230	230	230		DMA reads TxQ	230	230	230
Return path	150	450	750		WR VDM through fabric	0	300	600
Land in register	77	77	77		Remote Read Req thru Fabric	0	300	600
Total (ns)	607	1207	1807		Read of host/RC memory	230	230	230
					Read Completion thru Fabric	0	300	600
Parameters in ns					Dest DMA writes to RxQ, RxCQ	150	150	150
Host Read round trip read latency	230				RxCQ Interrupt Received	150	150	150
Switch fall through latency	150				Total (ns)	910	1810	2710

Simulated Throughput vs Payload Size



Questions

- JRegula@plxtech.com