IOV hardware and software plumbing for VMware ESX - journey to uncompromised I/O Virtualization and Convergence

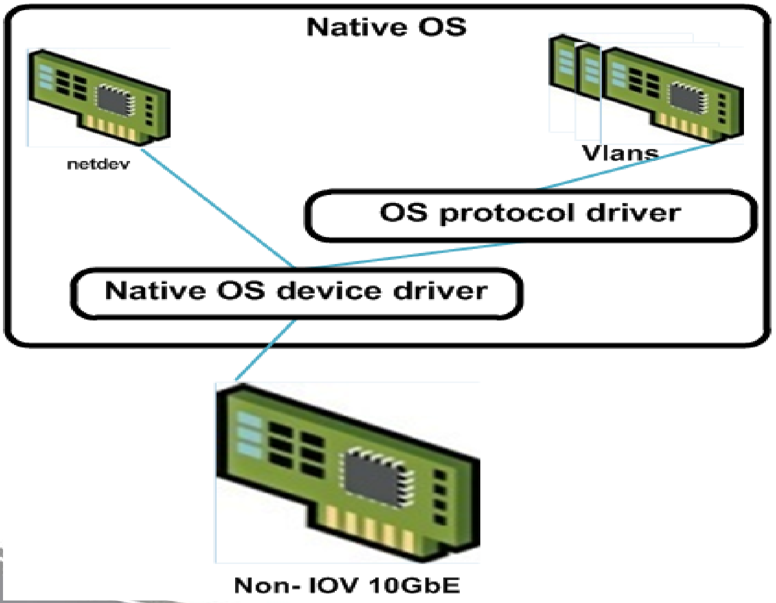
leonid.grossman@exar.com

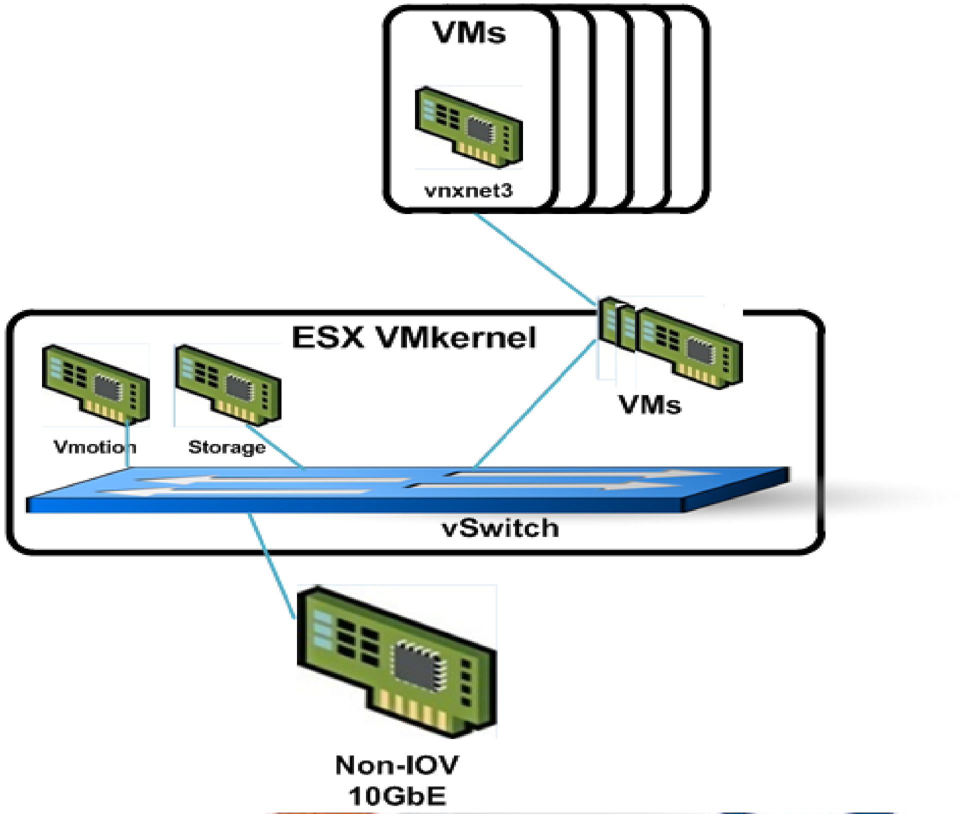
Major trends for I/O plumbing in datacenter and cloud

- **I/O virtualization**
- **10GbE and a “Perfect storm” of I/O innovation**
- **Server I/O virtualization on volume servers**
  - Software “para-virtualized” IOV in hypervisors
  - Hardware IOV and PCI-SIG SR IOV standard
- **I/O fabric convergence - network, storage and cluster controllers in a single pci-e slot**
- **Dedicated hardware I/O offload engines for running I/O protocols like iSCSI, FCoE and IPSec**

Evolution of software IOV

- **Virtual I/O in "Native" and Guest OS**
- **Limitations of sharing I/O device between multiple "tenants"**
  - Virtualization overhead
  - Loss of "native" features
  - *Lack of SLA guarantees*





Traditional virtualized networking has poor SLA, and reminds plumbing in older Silicon Valley house - once low priority users come in, high priority users see their service degraded.

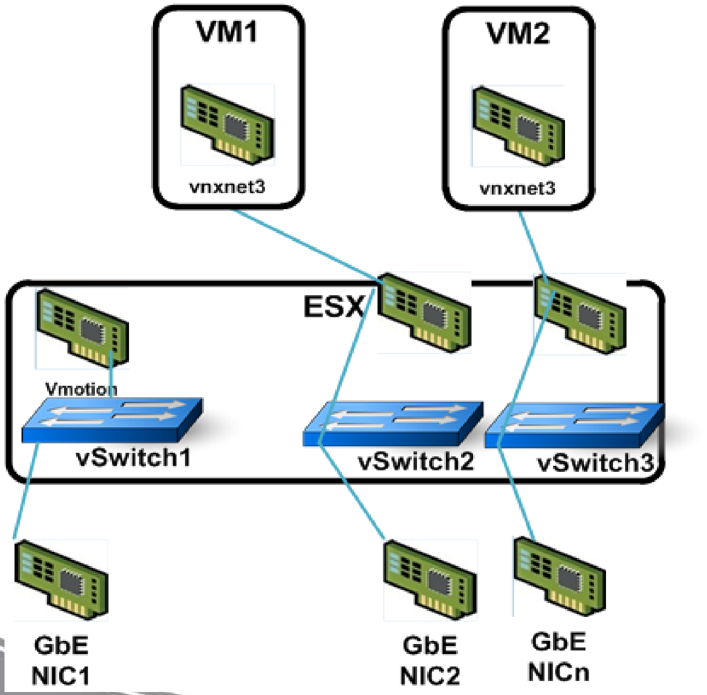
Low water pressure for morning showers
Sprinklers flow is unchecked.
Laundry flow is unchecked.

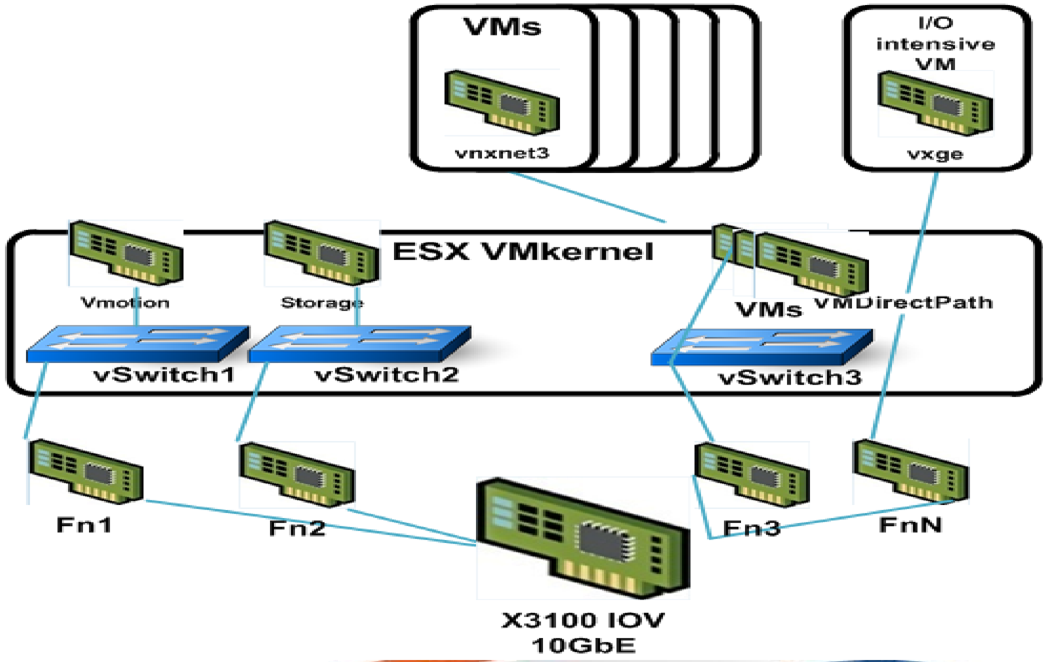
IOV solutions enable performance SLA guarantees for most demanding I/O applications, isolating them from low priority “tenants” sharing the same device.

Water pressure for shower is guaranteed.
Sprinklers flow is limited if shower is on.
Laundry flow is limited if shower is on.

Evolution of hardware IOV

- **Pci-e multi-port and multi-function solutions**
- **Single-Root (SR IOV) PCI-SIG specification**
  - Many products are implemented as “poor man” multi-function
  - Not supported in Microsoft and VMware hypervisors
- **Direct Hardware Access (passthru)**

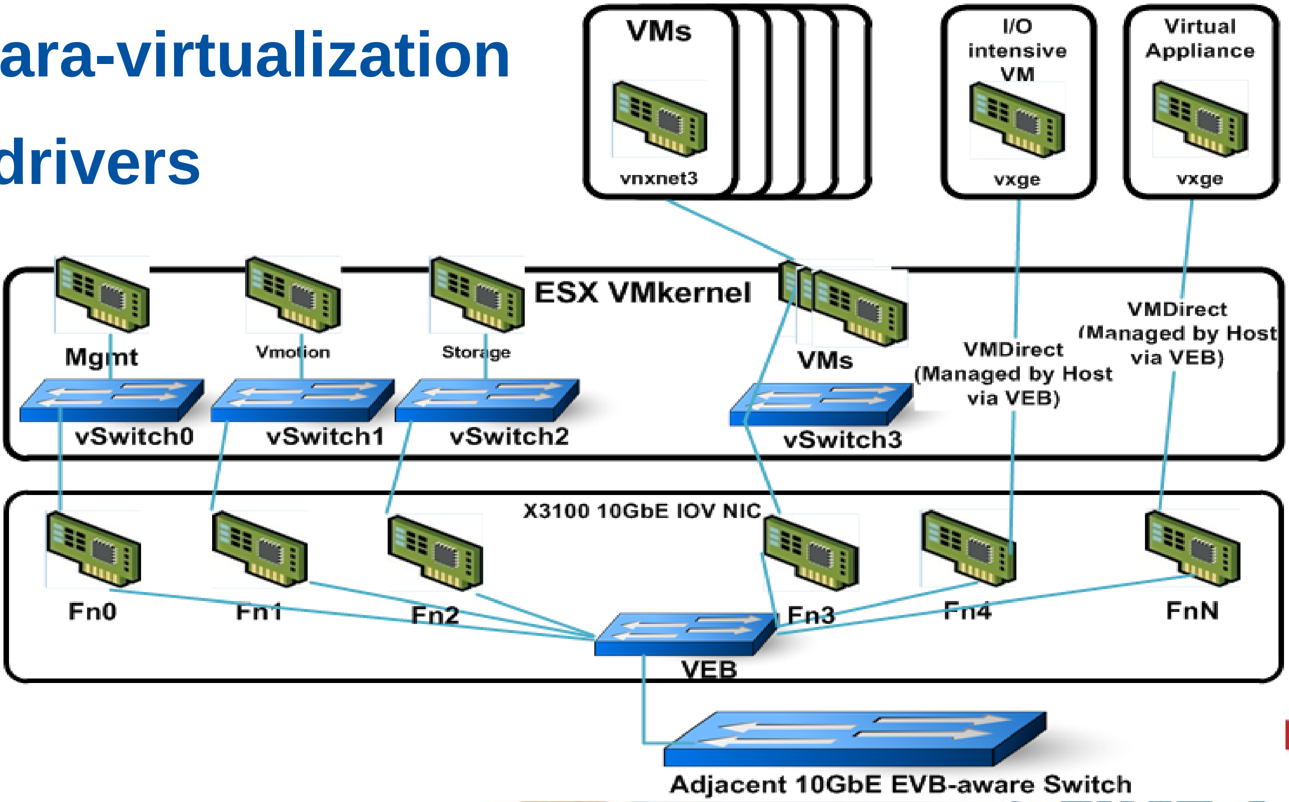




ESX direct hardware access with MF IOV

- Guest OS can bypass hypervisor and access I/O devices (including iSCSI, etc engines) directly, potentially getting performance and features customary in standalone OS
- Complements para-virtualization
- Can use native drivers
- Allows GOS

to support offloads like iSCSI, IPSec etc - even if Host doesn't



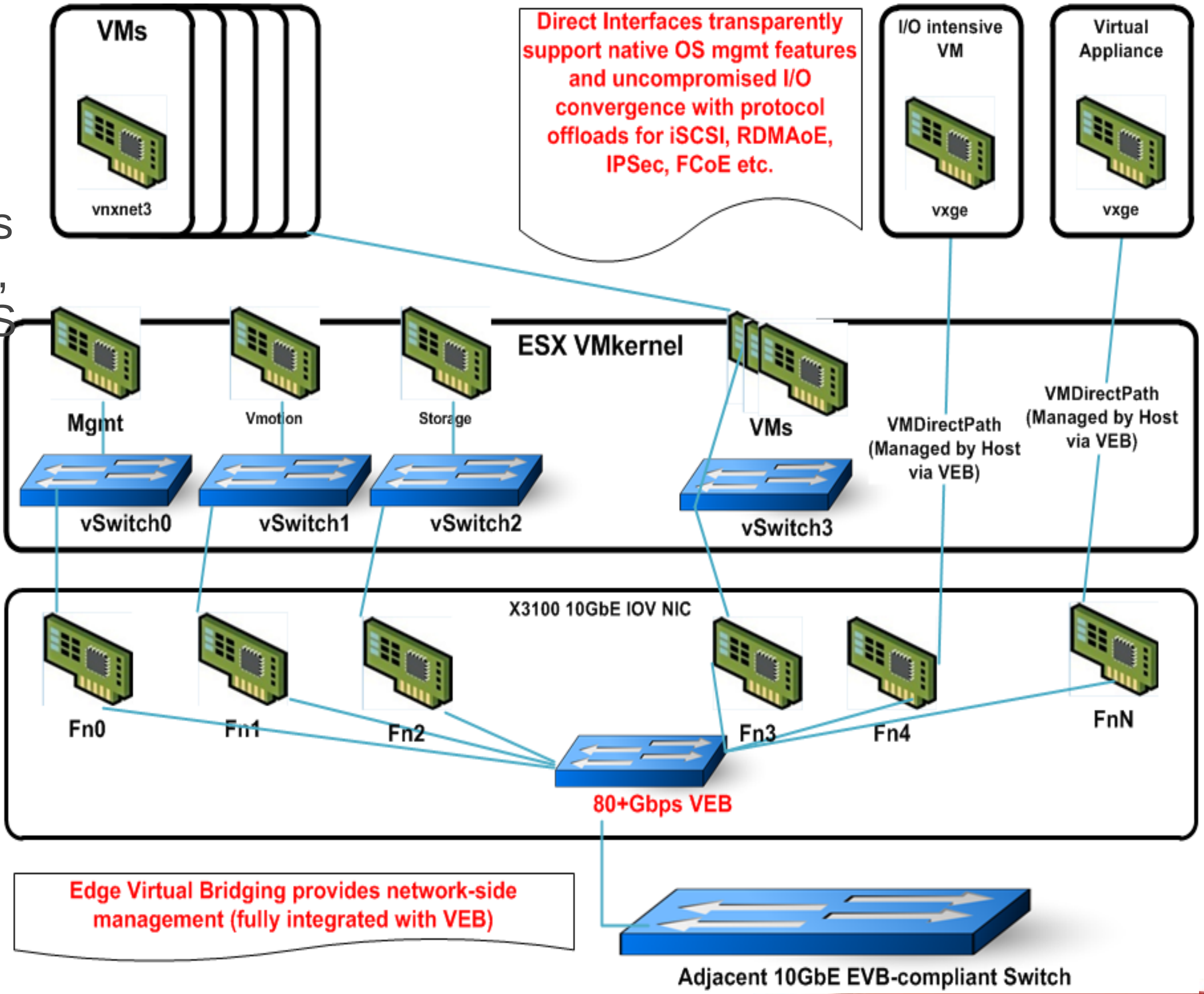
Summary of hardware IOV features used in “best ESX practices”

Best IOV practices …	Backed by 10GbE IOV features
Use separate pci-e function as a pnic behind each “differentiated” hypervisor vnic (example – iSCSI, vmotion, ESX console) Share one of the functions between “non-differentiated” vnics (example – VDI traffic)	Large number of pci-e functions, both SR IOV and traditional pci-e multifunction (for host OSes like ESX 4.x and Hyper-V)
Assign SLA (bandwidth, priority, latency etc) for each “differentiated” traffic type like iSCSI.	SLA support in hardware (throughout all ASIC blocks), to compliment software facilities like IOControl
Use Guest direct hardware access if “native” performance and feature set is required (example – iSCSI Virtual Appliance)	Each pci-e function is a complete device that can run “native” OS driver. Embedded L2 switch/VEB to support hypervisor control over privileged operations on passthru interfaces.
Use both host and network side mgmt tools as appropriate	Support for upcoming Edge Virtual Bridge IEEE standard

EXAR Powering Connectivity™

Destination – uncompromised I/O Virtualization and Convergence

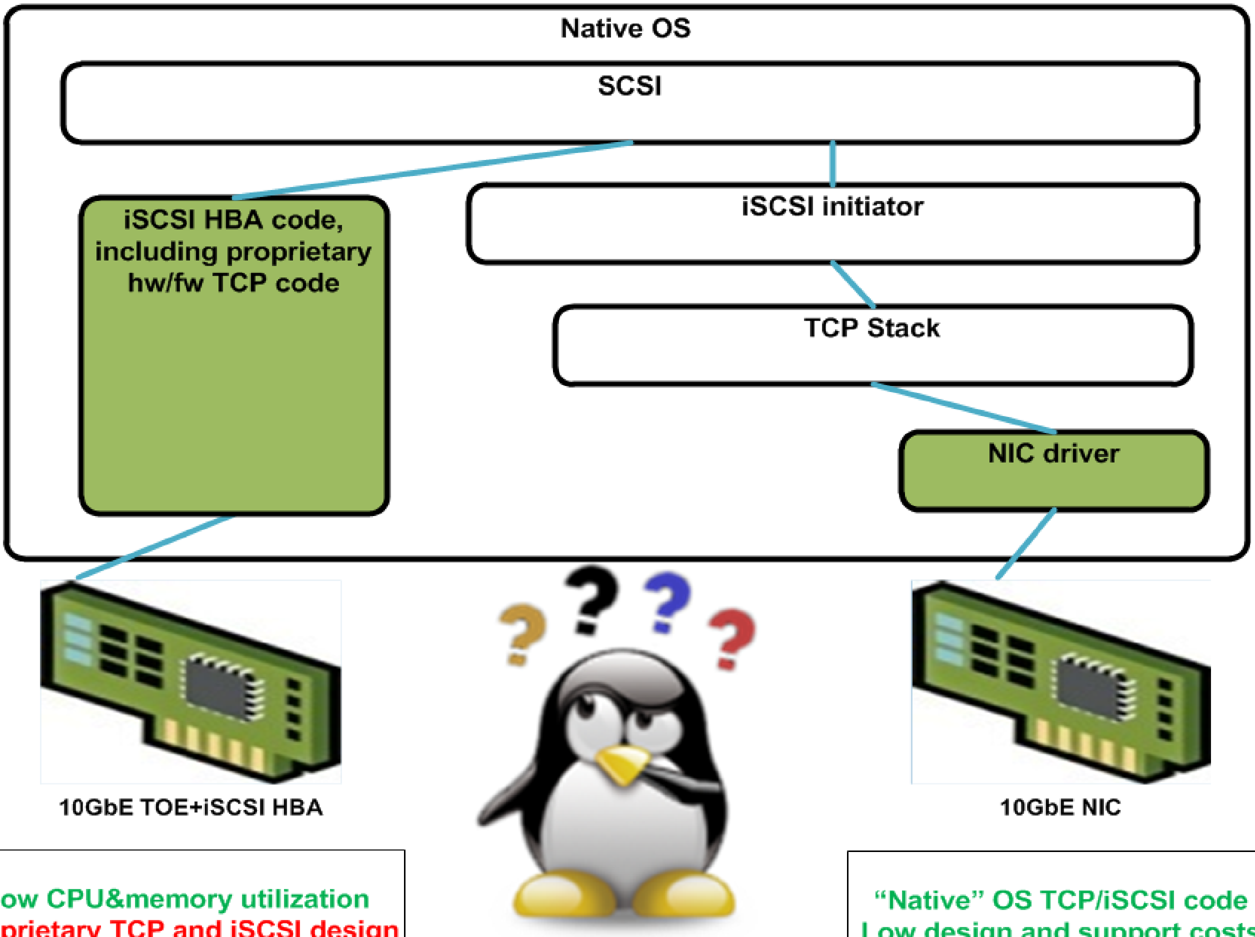
- **SR-IOV**
  - High number of 10GbE interfaces in a slot
- **Native Converged I/O Fabric**
  - Hardware protocol offloads available on each function, to both Host and Guest OS
  - iSCSI, RDMA over Ethernet, IPSec, FCoE
- **PCI-E Gen.3**
  - 60+Gbps of additional bandwidth for Inter-VM and Virtual Appliance traffic via VEB (without moving to 40/100Gbps)
- **IEEE Edge Virtual Bridging**
  - Complimentary to VEB
  - Improves Host mgmt
  - Supports Network mgmt



VMs
vnxnet3
Direct Interfaces transparently support native OS mgmt features and uncompromised I/O convergence with protocol offloads for iSCSI, RDMAoE, IPSec, FCoE etc.
I/O intensive VM
vxge
Virtual Appliance
vxge
ESX VMkernel
Mgmt
Vmotion
Storage
VMs
VMDirectPath (Managed by Host via VEB)
VMDirectPath (Managed by Host via VEB)
vSwitch0
vSwitch1
vSwitch2
vSwitch3
X3100 10GbE IOV NIC
Fn0
Fn1
Fn2
Fn3
Fn4
FnN
80+Gbps VEB
Edge Virtual Bridging provides network-side management (fully integrated with VEB)
Adjacent 10GbE EVB-compliant Switch

An example of IOV-friendly offload engine – Exar TOE-less iSCSI 10GbE project.

- Traditional iSCSI engines are hard to integrate with native OS iSCSI….
- Much harder to integrate with hypervisor iSCSI…
- Almost impossible to use directly with multiple VM/VA GOS iSCSI



Low CPU&memory utilization
Proprietary TCP and iSCSI design
High design and support costs
Doesn't leverage host advances
No migration capabilities

"Native" OS TCP/iSCSI code
Low design and support costs
Leverages host advances and migration capabilities
High CPU&memory utilization

Motivation for TOE-less iSCSI project – issues with iSCSI HBA

- **Most issues with iSCSI engines are related to TCP offload**
- **Problems with TOE are well documented, some/most are applicable to traditional iSCSI HBAs as well**

http://www.linuxfoundation.org/collaborate/workgroups/networking/toe

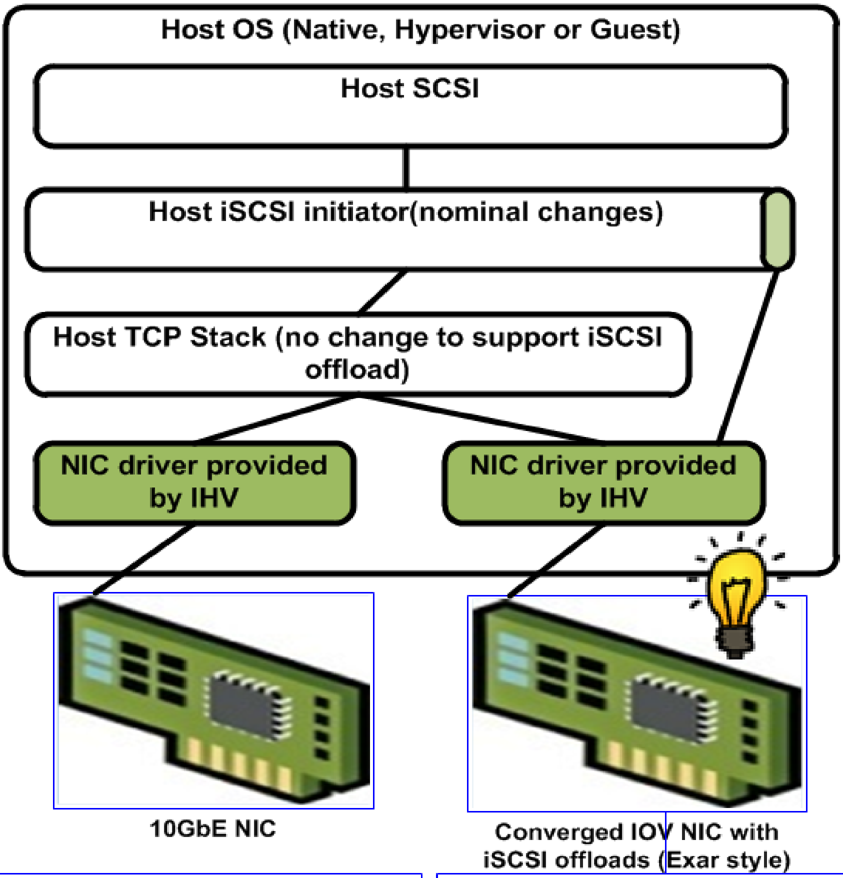
- **Integration of iSCSI HBA with host iSCSI software (like Open iSCSI) results in massive amounts of proprietary code**
- **Additional high-speed hardware design problems associated with TOE**
  - Vast amount of memory for buffering and context
  - Complex design and verification
  - Complex field support
  - Costs become visible and problematic as 10GbE prices drop
- **Virtualization makes all these issues much worse….**

Motivation for TOE-less iSCSI project – in search of a better offload engine

- **I/O Offload History**
  - Some I/O offloads (LSO, LRO, checksum) became widely adopted and integrated into Operating systems.
  - Other I/O offloads (TOE) never really “made it”, or became obsolete with software and hardware advances.
- **Virtualization amplified issues with “stateful” I/O HBAs**
  - HBAs accelerate/terminate a single instance of an I/O protocol. Little support for SLA and state migration; no support for offload on each IOV function.
  - Limited support for I/O offload (iSCSI, but also RDMA over Ethernet, IPSec, compression etc) in Host OSes
- **Conclusion – to be viable part of the virtualization and converged I/O fabric, iSCSI offload engine needs to:**
  - Be available on each IOV function
  - Accelerate software iSCSI stack, rather than replacing it with a parallel hardware stack.

TOE-less iSCSI offload – it can be done!

- **A TOE engine is NOT a pre-requisite for iSCSI engine.**
- **Offloading most intensive parts of iSCSI while running over unmodified host TCP code is not only possible, but superior from every angle (design, support, performance)**
  - Avoids all the problems associated with two TCP stacks in the system
  - Reuses existing "tried and true" system TCP and iSCSI code for slow path and exception handling
  - Offloads most intensive parts of TCP and iSCSI processing
  - Provides lower design, acquisition and support costs relative to TOE-based HBA – with a possibility to outperform
    - Will continue to benefit from future host hw/sw advances
    - All networking accelerators are still at work on the slow path
  - Highly desirable for virtualized platforms that require iSCSI acceleration and SLA on each IOV unction


Host OS (Native, Hypervisor or Guest)
Host SCSI
Host iSCSI initiator(nominal changes)
Host TCP Stack (no change to support iSCSI offload)
NIC driver provided by IHV
NIC driver provided by IHV
10GbE NIC
Converged IOV NIC with iSCSI offloads (Exar style)

"Native" OS TCP/iSCSI code Low design and support costs Leverages host advances and migration capabilities High CPU&memory utilization	Low CPU&memory utilization "Native" OS TCP/iSCSI code Low design and support costs Leverages host advances and migration capabilities

Backup slides

Details of TOE-Less iSCSI Offload

- **Uses the "Native" Host TCP Stack "as is" – including all offloads like LRO, TSO, checksum, etc**
  - No TCP termination
  - No TCP connection setup/termination
  - No custom utilities/ARP/ND/ICMP
  - No IP space partitioning
  - No responsibility for TCP security stack
  - No convincing customers that it is done right ☺
  - Networking administration doesn't change
- **Supports much more efficient hardware implementation**
  - Design
  - Verification
  - Support

Details of TOE-Less iSCSI, continued

- **Reuse of Existing iSCSI Implementations like Open iSCSI**
- **Host iSCSI "Slow Path" is entirely reused**
  - Connection setup/teardown
  - iSCSI Parameter Negotiation Process
  - Error recovery, like placing out-of-order data
    - TOE-less design makes OOO recovery painless – it switches between offload and non-offload paths in one iSCSI PDU, and can take it's time to recover iSCSI header
- **Host iSCSI "Fast Path" needs only minor modifications**
  - To support CRC offload on Tx and Rx side
  - To support direct placement for iSCSI Initiator Read
  - Overall, only about 10% of iSCSI HBA Open iSCSI code

Details of TOE-Less iSCSI, continued

- **Details of Open iSCSI modifications:**
  - Patch iscsi_tcp_recv() in iscsi_tcp.c to look for pre-posted SGL completion
  - Patch iscsi_tcp_conn_create() in iscsi_tcp.c to let HW know about a context of ISCSI session
  - Patch iscsi_tcp_conn_release() in iscsi_tcp.c to let HW know the previously created context of ISCSI session is destroyed
  - patch libiscsi.c to post SGL to HW
  - disable local digest computation
- **Total code Size: About 10% of the typical iSCSI HBA Open iSCSI code size**

EXAR
Powering Connectivity™

Thank you!
Q&A