iSCSI Extensions for RDMA
Updates and news

Sagi Grimberg
Mellanox Technologies

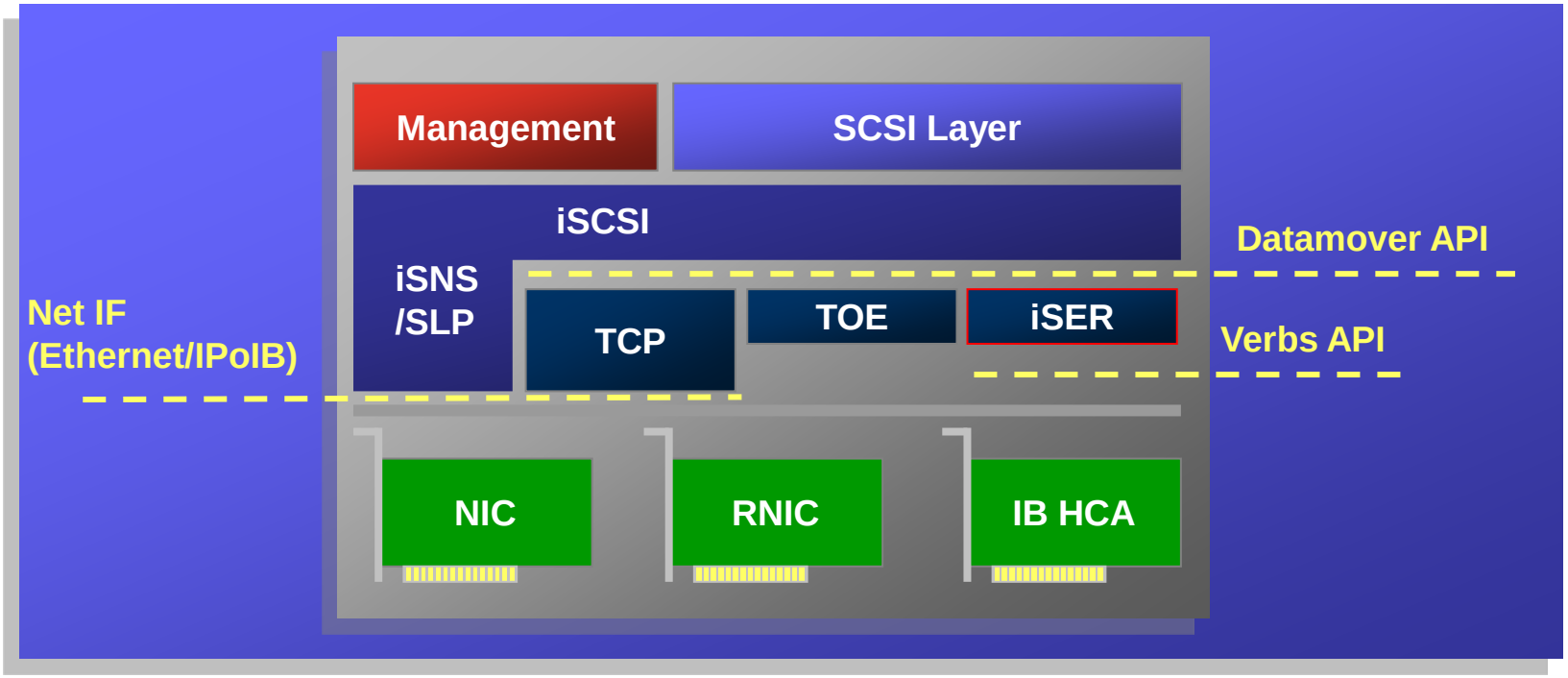
Agenda

- iSER (short) Overview
- Linux Updates and Improvements
- Data Integrity Offload (T10-DIF)
- Performance
- Future Plans
- Applications & Deployments

iSER (short) Overview

iSCSI and iSER Architecture

- Part of IETF: RFC-7147

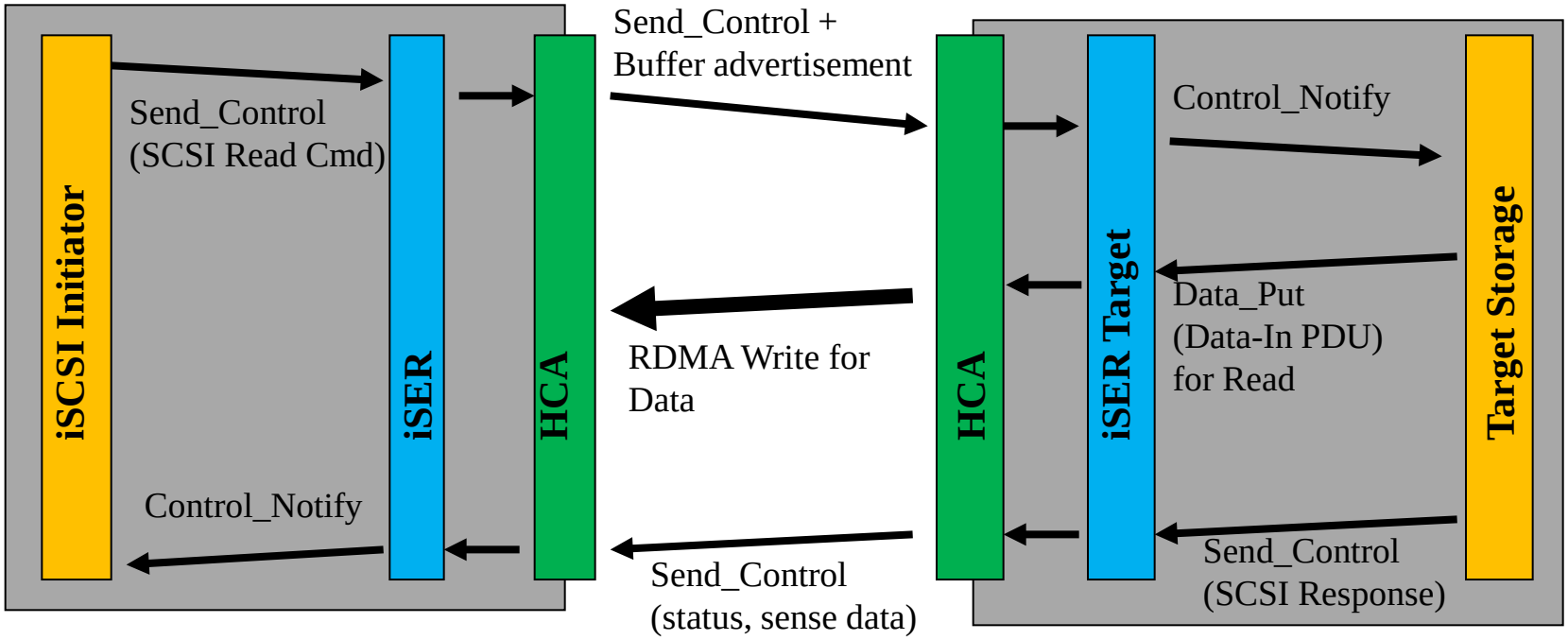


- The transport layer iSER and/or iSCSI/TCP are transparent to the user. Just need a simple configurable to decide

iSER Benefits

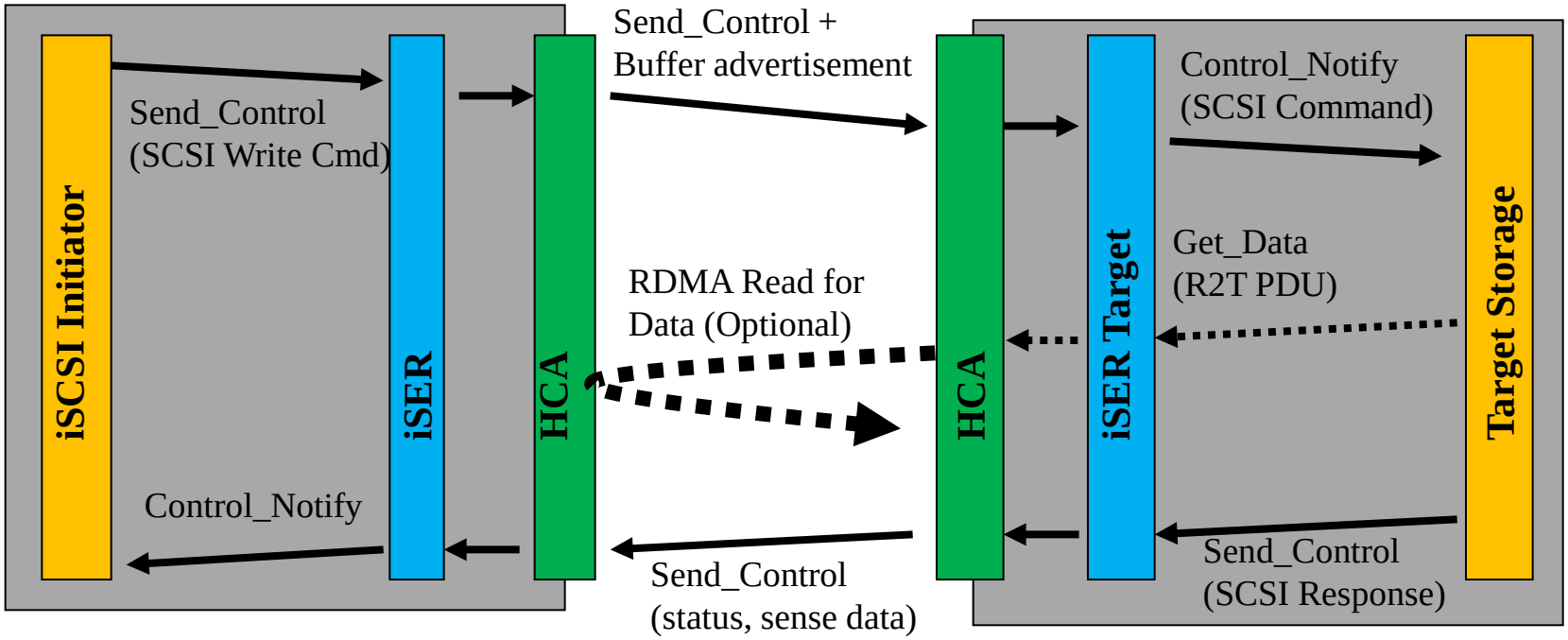
- Zero-Copy
- CPU offload
- Fabric reliability (lossless medium)
- High IOPs, Low Latency
- Inherits rich iSCSI management
- Link aggregation (bonding)
- Fabric consolidation (Same fabric for storage, networking and management)
- Works over Infiniband and Ethernet (converged) fabrics

iSER Protocol Overview (Read)



- **SCSI Reads**
  - **Initiator Send Command PDU (Protocol data unit) to Target**
  - **Target return data using RDMA Write**
  - **Target send Response PDU back when completed transaction**
  - **Initiator receives Response and complete SCSI operation**

iSER Protocol Overview (Write)



- **SCSI Writes**
  - **Send Command PDU (optionally with Immediate Data to improve latency)**
  - **Map R2T to RDMA Read operation (retrieve data)**
  - **Target send Response PDU back when completed transaction**

Available Targets

Coming up…

Coming up…

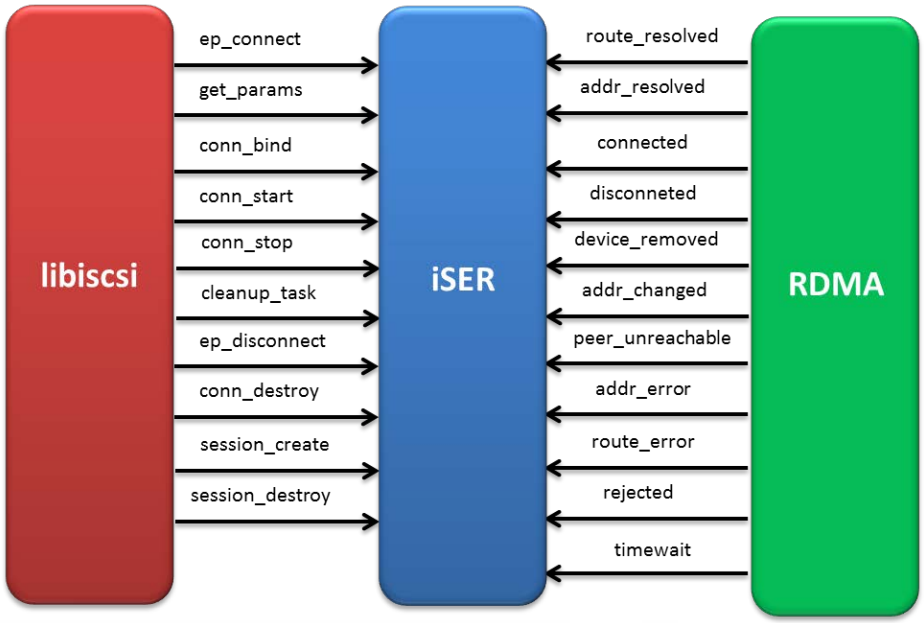
Coming up…

Linux Updates and Improvements

What have we been up to lately…

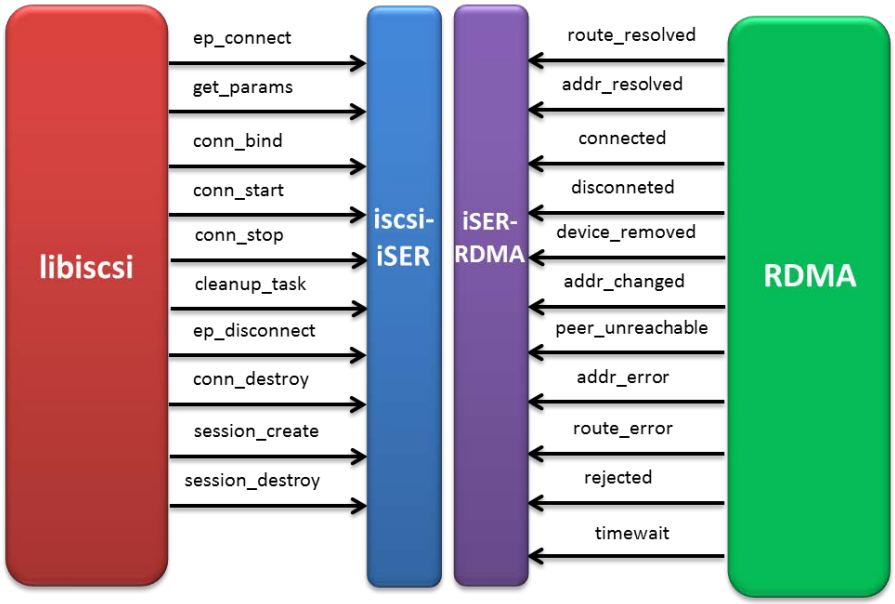
Re-Design iSER Initiator Control Plane

- iSER layer mediates between two connection management layers
  - iSCSI
  - RDMA-CM
- The former design:
  - Allow competetions
  - reference counters
  - Establish & Teardown dependency



Re-Design iSER Initiator Control Plane

- New Design: **Divide & Concur**
  - iSCSI-iSER layer: logical connection
  - iSER-RDMA layer: RDMA resources
  - Minimal dependencies



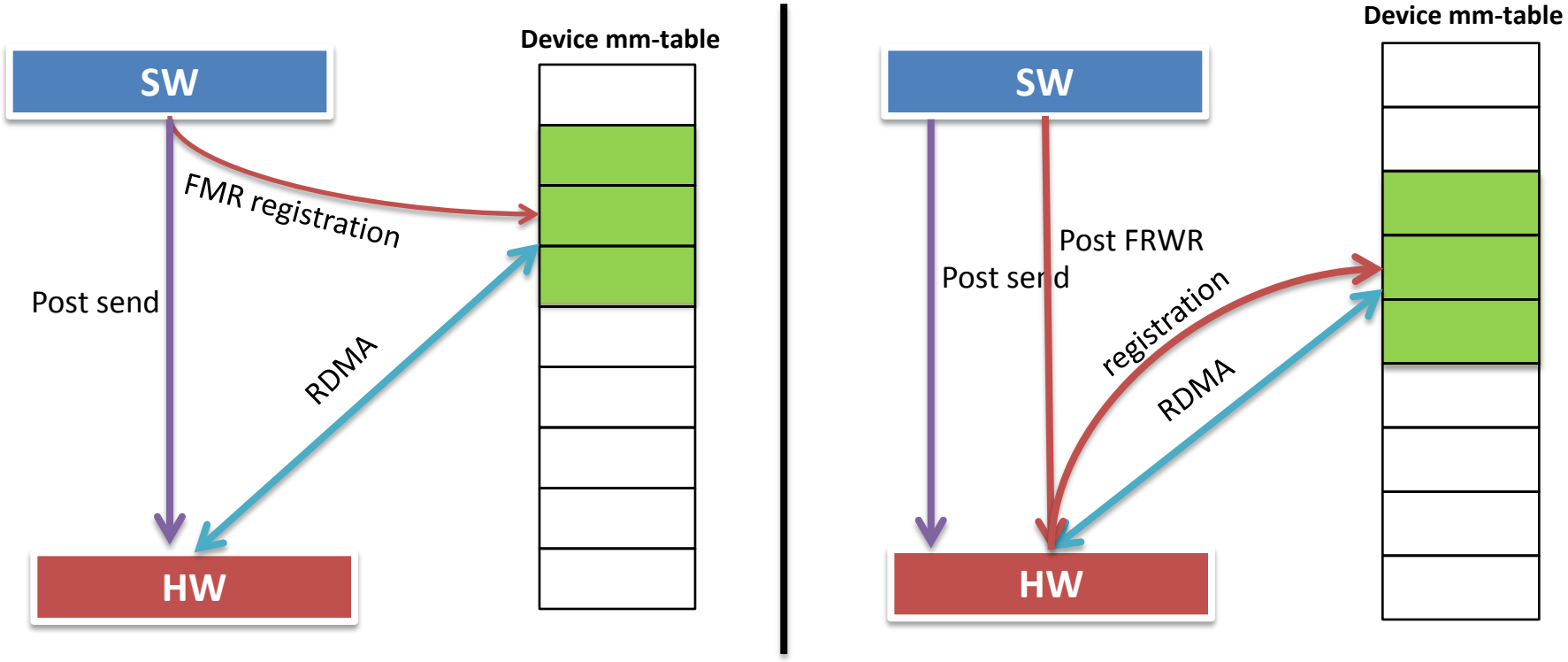
- iSER initiator passes long duration of test suites:
  - Target reset/kill/shutdown at random stages in login/logout sequence
  - Large scale fabric login/logout loops - hundreds of targets
  - Random Device removal.
  - And more…

Target Discovery over RDMA

- Originally, no discovery via iSER
  - Discovery was done using TCP/IP
- Embedded target may not have a TCP stack
- Since kernel 3.14 iSER supports discovery
  - Extended text negotiation capability to support ‘sendtargets’
- Added to open-source targets (TGT, LIO, SCST)

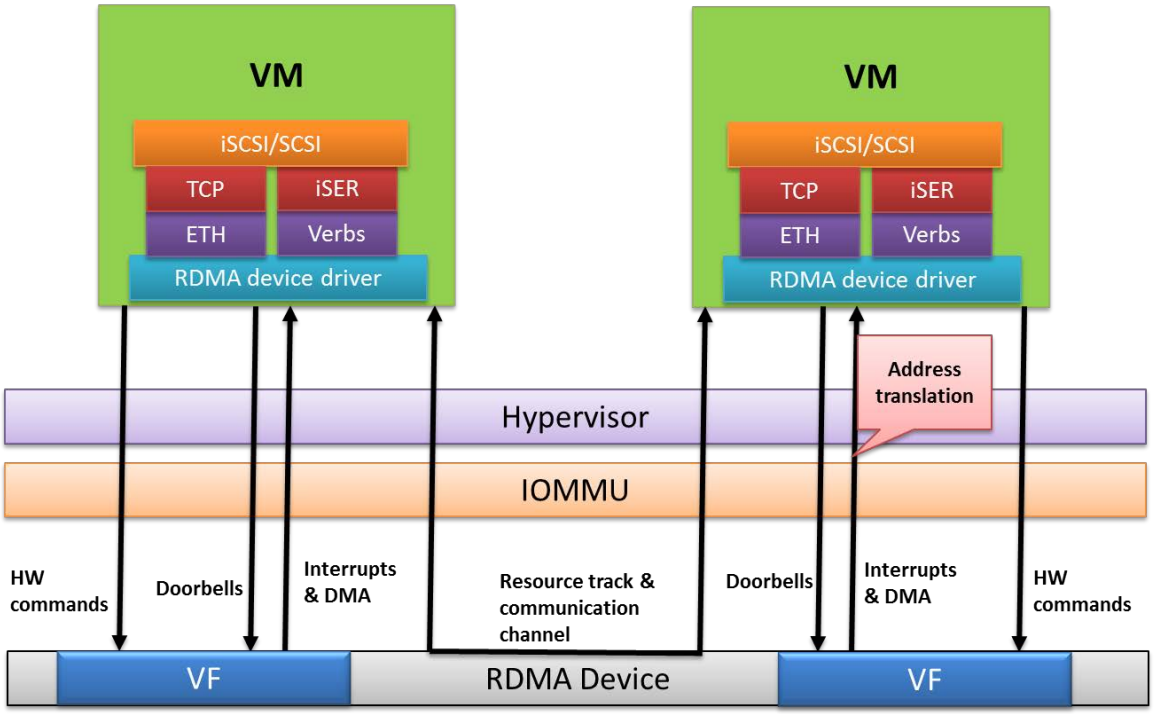
Fast Memory Registration in SRIOV

- Legacy Memory registration scheme (FMR pools) is not supported for Virtual functions and also in next generation RDMA Devices



Fast Memory Registration in SRIOV

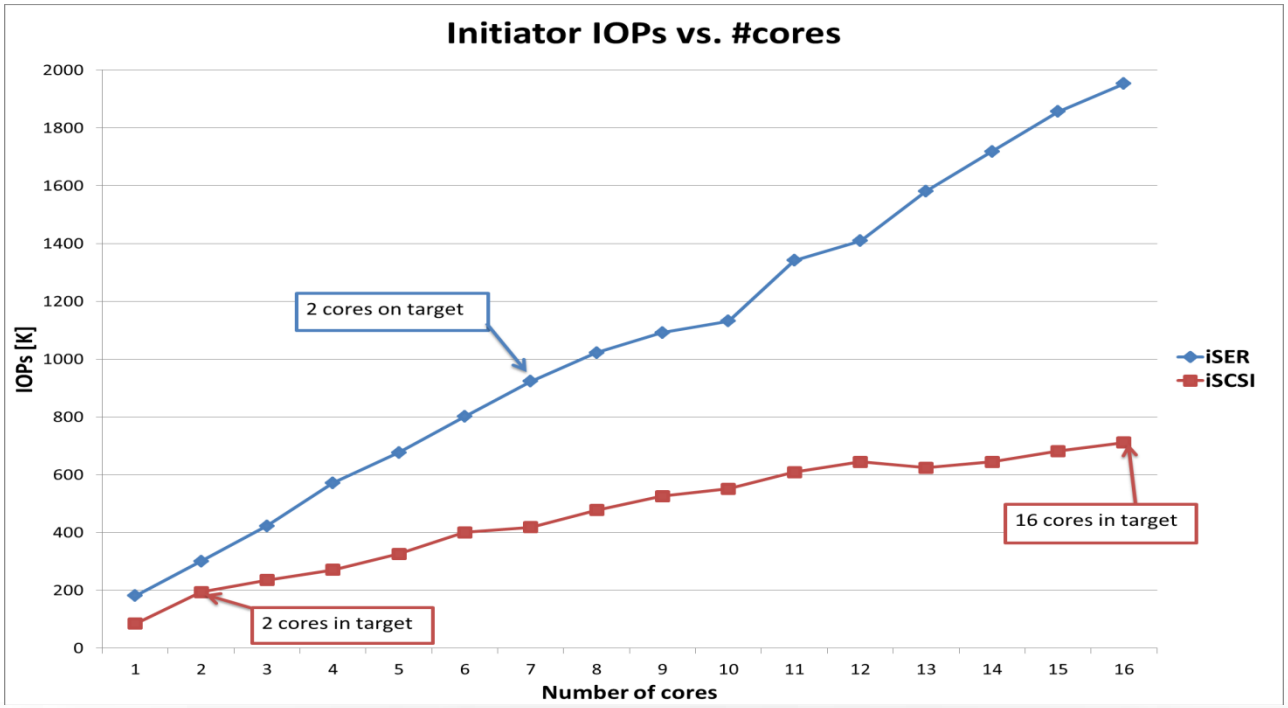
- Since kernel 3.13 iSER initiator supports Fast registration work requests (FRWR) scheme to allow efficient memory registration also in virtual functions



Improved iSER performance over virtual functions (VMs with SRIOV)

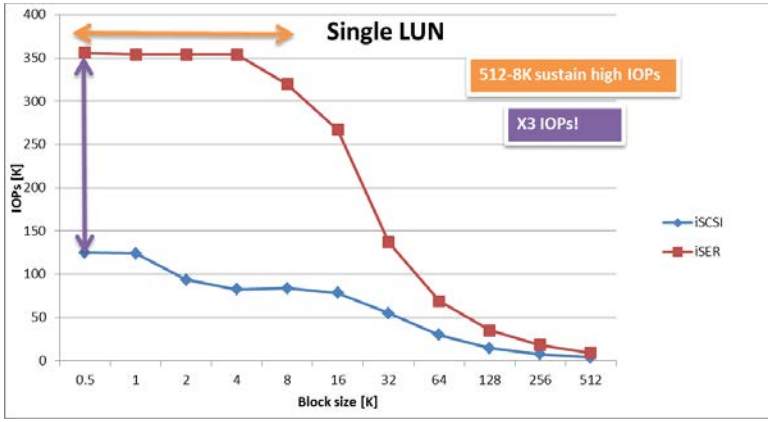
Performance Enhancements

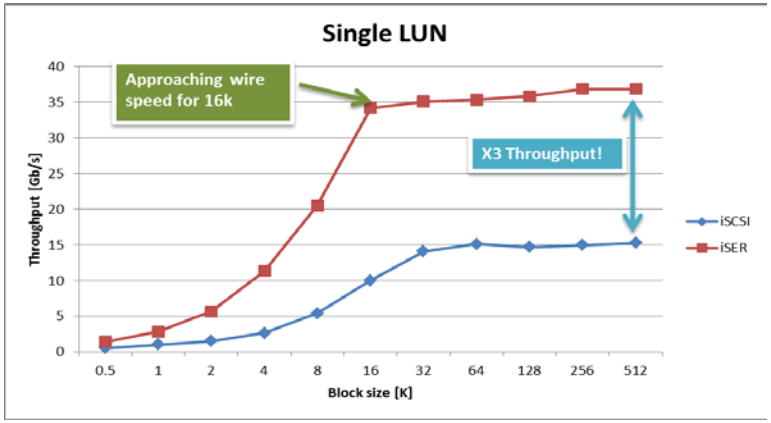
- Per-CPU Completion contexts
- Maintain internal polling budget for soft-irq completion processing fairness
- Interrupt moderation config options
- Still we have a lot more to do…

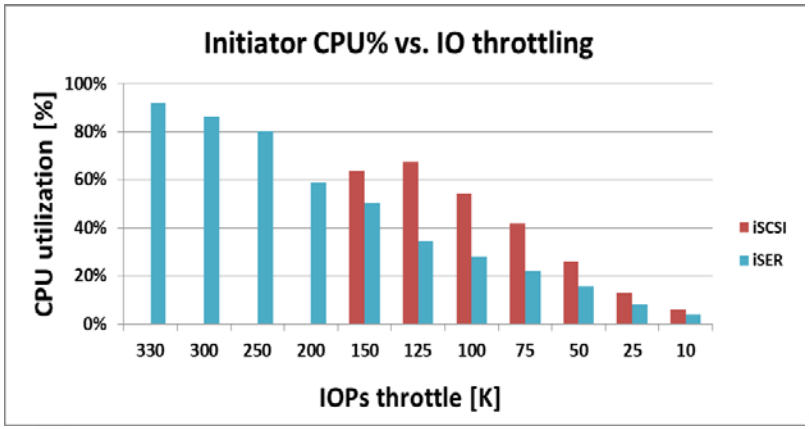


Performance Enhancements

- Some more numbers (Single LUN)…





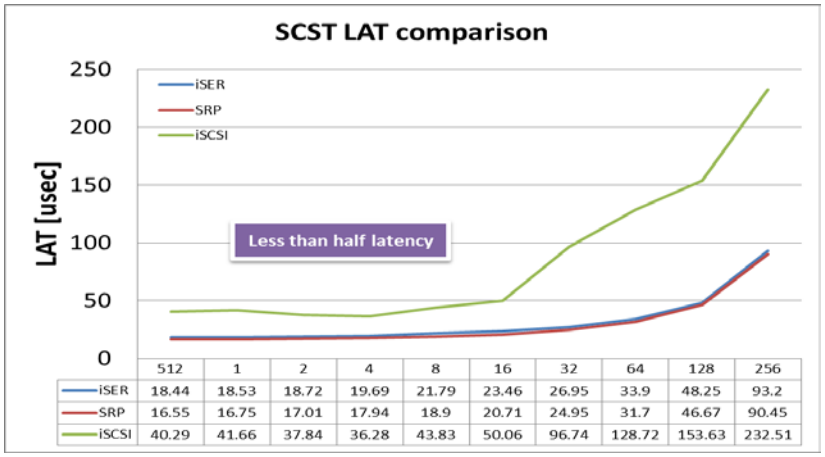


iSER in SCST (New!)

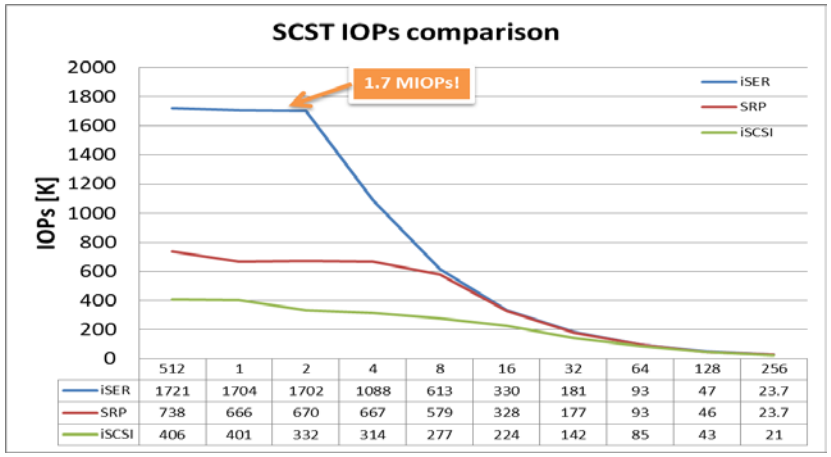
- A common SCSI target implementation in Linux
- Added a transport abstraction framework to fit RDMA extensions as well as TCP
- Achieves high IOPs & Throughput
- Stable!
- Available at http://scst.sourceforge.net/target_iser.html

iSER in SCST - Performance

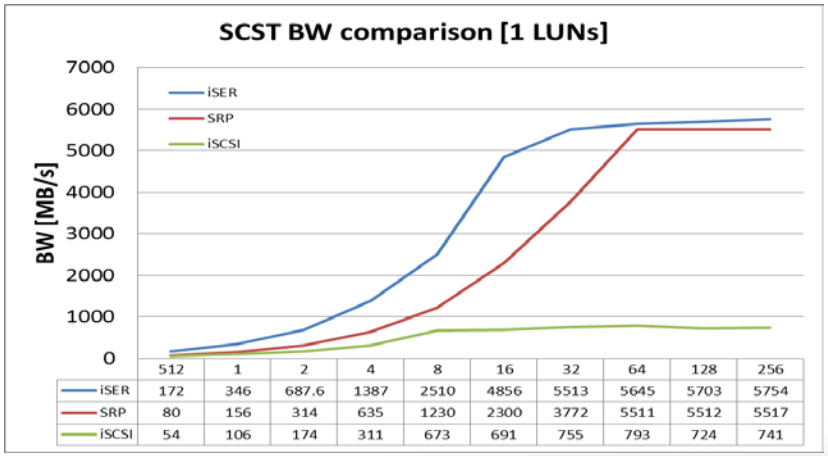
Single Initiator to Single Target
ConnectX-3 Adapter (FDR)



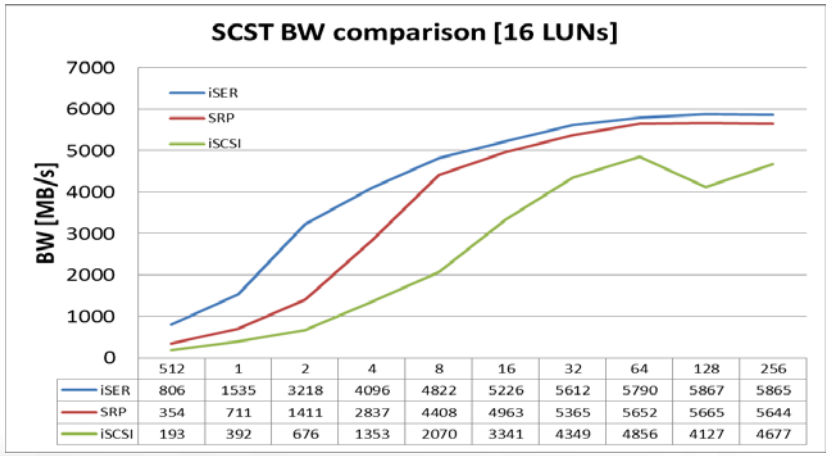
	512	1	2	4	8	16	32	64	128	256
iSER	18.44	18.53	18.72	19.69	21.79	23.46	26.95	33.9	48.25	93.2
SRP	16.55	16.75	17.01	17.94	18.9	20.71	24.95	31.7	46.67	90.45
iSCSI	40.29	41.66	37.84	36.28	43.83	50.06	96.74	128.72	153.63	232.51



	512	1	2	4	8	16	32	64	128	256
iSER	1721	1704	1702	1088	613	330	181	93	47	23.7
SRP	738	666	670	667	579	328	177	93	46	23.7
iSCSI	406	401	332	314	277	224	142	85	43	21



	512	1	2	4	8	16	32	64	128	256
iSER	172	346	687.6	1387	2510	4856	5513	5645	5703	5754
SRP	80	156	314	635	1230	2300	3772	5511	5512	5517
iSCSI	54	106	174	311	673	691	755	793	724	741



	512	1	2	4	8	16	32	64	128	256
iSER	806	1535	3218	4096	4822	5226	5612	5790	5867	5865
SRP	354	711	1411	2837	4408	4963	5365	5652	5665	5644
iSCSI	193	392	676	1353	2070	3341	4349	4856	4127	4677

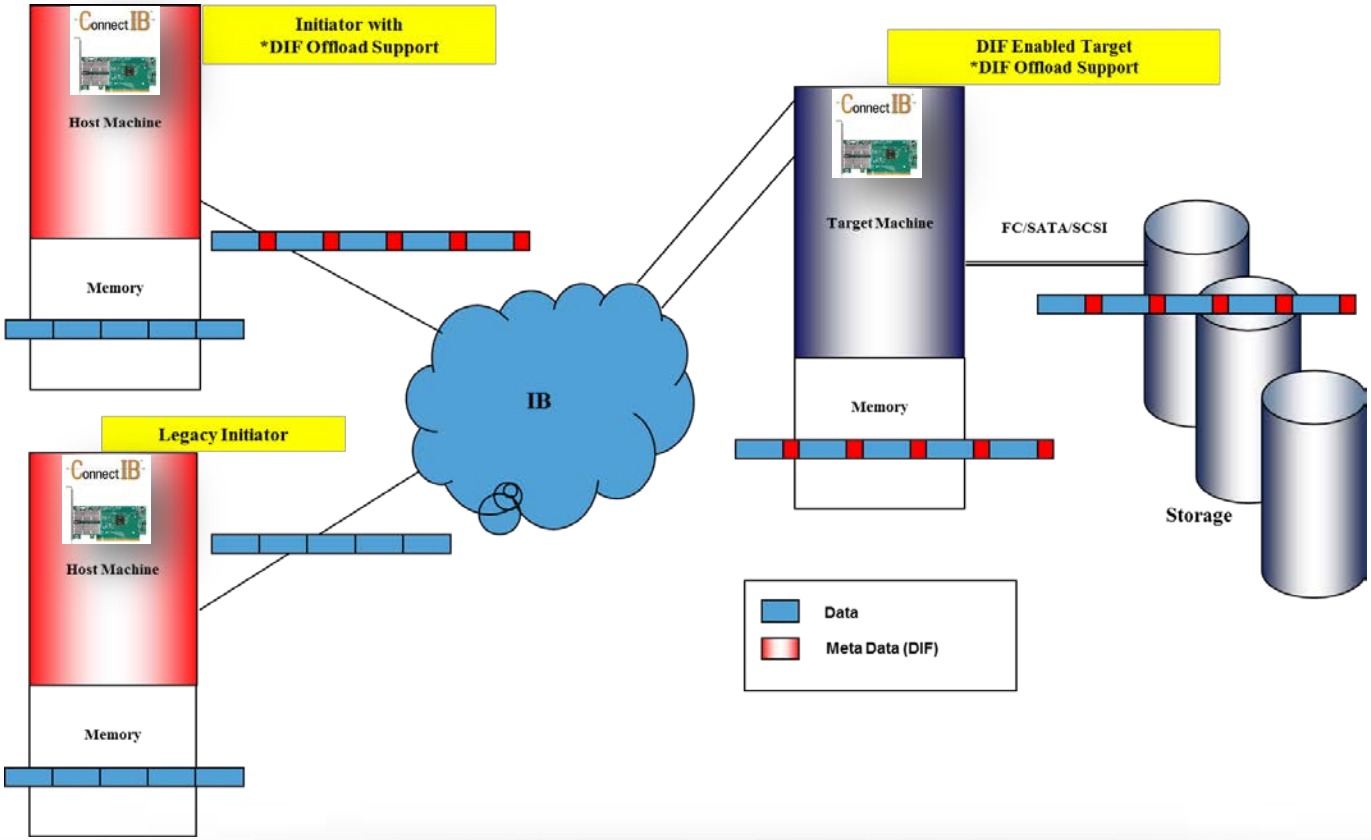
Major Stability Fixes in LIO iSER Target

- LIO iSER target is becoming attractive for Cloud/SDS
- Recent work allows LIO iSER target to support large scale fabrics
  - Rework parallel initiator login requests
  - Rework RDMA CM events handling
  - Rework memory management and fast registration
- Next: optimizations…

End-to-end Data Integrity Offload (T10-DIF)

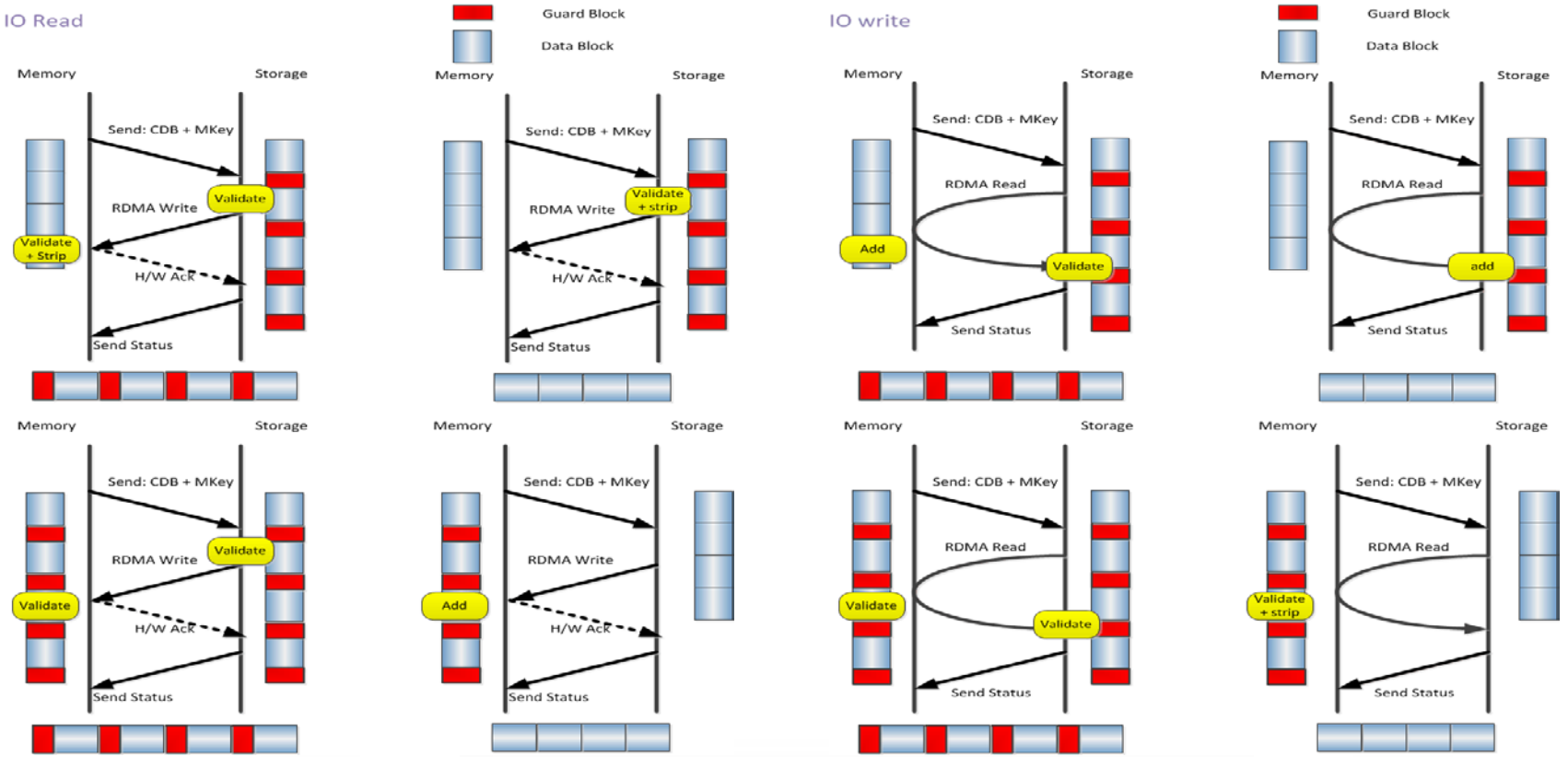
RDMA Signature Feature

- Mellanox ConnectIB HCA introduced data integrity offload support over RDMA communication



RDMA Signature Feature

- RDMA verbs layer was extended to support “signature handover” operations
- The “signature handover” operation is handing over data from memory to wire (and vice-versa) while verifying/passing/stripping/generating data integrity



RDMA Signature feature

- An RDMA application that wants to use data-integrity will need to take 5 simple steps:

1. **Allocate Signature enabled memory regions (session startup)**

```
mr_init_attr.flags |= IB_MR_SIGNATURE_EN;
sig_mr = ib_create_mr(pd, &mr_init_attr);
```

2. **Set QP as Signature enabled (session startup)**

```
qp_init_attr.create_flags |= IB_QP_CREATE_SIGNATURE_EN;
sig_qp = ib_create_qp(pd, &qp_init_attr);
```

3. **Register Signature MR (send work request `IB_WR_REG_SIG_MR`)**

```
sig_wr.opcode = IB_WR_REG_SIG_MR;
sig_wr.sg_list = data_sge;                      /* Data buffer */
sig_wr.wr.sig_handover.prot = prot_sge;         /* protection buffer */
sig_wr.wr.sig_handover.sig_attrs = &sig_attrs;  /* signature attributes struct */
sig_wr.wr.sig_handover.sig_mr = pi_ctx->sig_mr; /* Signature enabled MR */
ret = ib_post_send(qp, sig_wr, &bad_wr);
```

4. **do RDMA (data-transfer)…**

5. **Check Signature status**

```
ret = ib_check_mr_status(sig_mr, IB_MR_CHECK_SIG_STATUS, &mr_status);
```

End-to-End T10-DIF Support - iSER

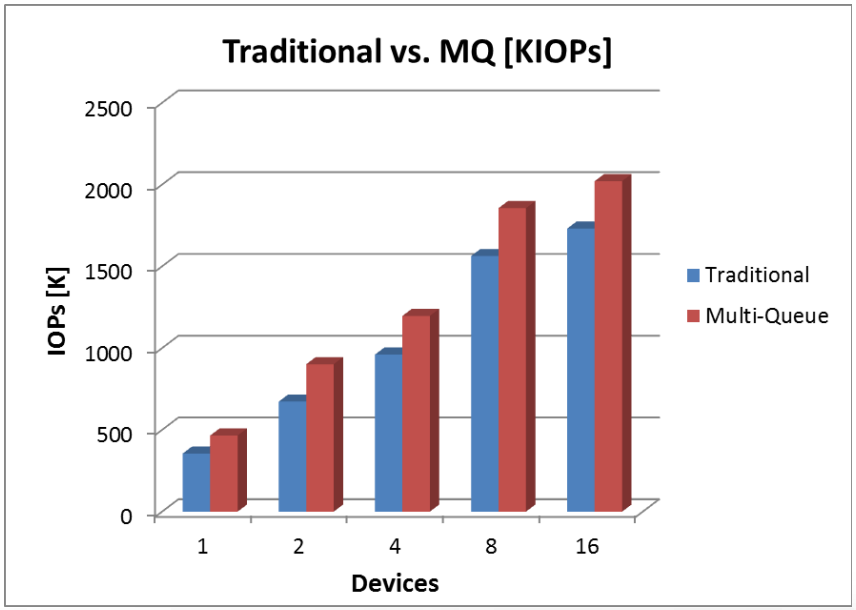
- The first RDMA signature API consumer is iSER
  - Added T10-DIF + DIX support to Linux iSER initiator
  - Added T10-DIF support to LIO iSER target
    - Also added T10-DIF support to Target core and backend emulations
  - Adding T10-DIF support to other open-source iSER target – pending on market requirements
  - Some iSER target vendors plan to support T10-DIF in coming models

Future Plans

What's next…

Multi-Queue Adoption

- The multi-queue block layer support (blk-mq) exists since 3.13
- The multi-queue SCSI layer support (scsi-mq) just recently included in 3.17 (thanks Christoph!)
  - Initial benchmarking using iSER show that scsi-mq significantly improves performance!



- HP-proliant: 16 (8x2) cores
- CPU model: Intel(R) Xeon(R) @ 2.60GHz
- Single FDR link.
- Mellanox ConnectIB

Multi-Queue Adoption

- RDMA devices are naturally multi-queued
  - Multiple HW queues
  - Spreading MSIX interrupts across CPUs allows spreading completion processing load better!

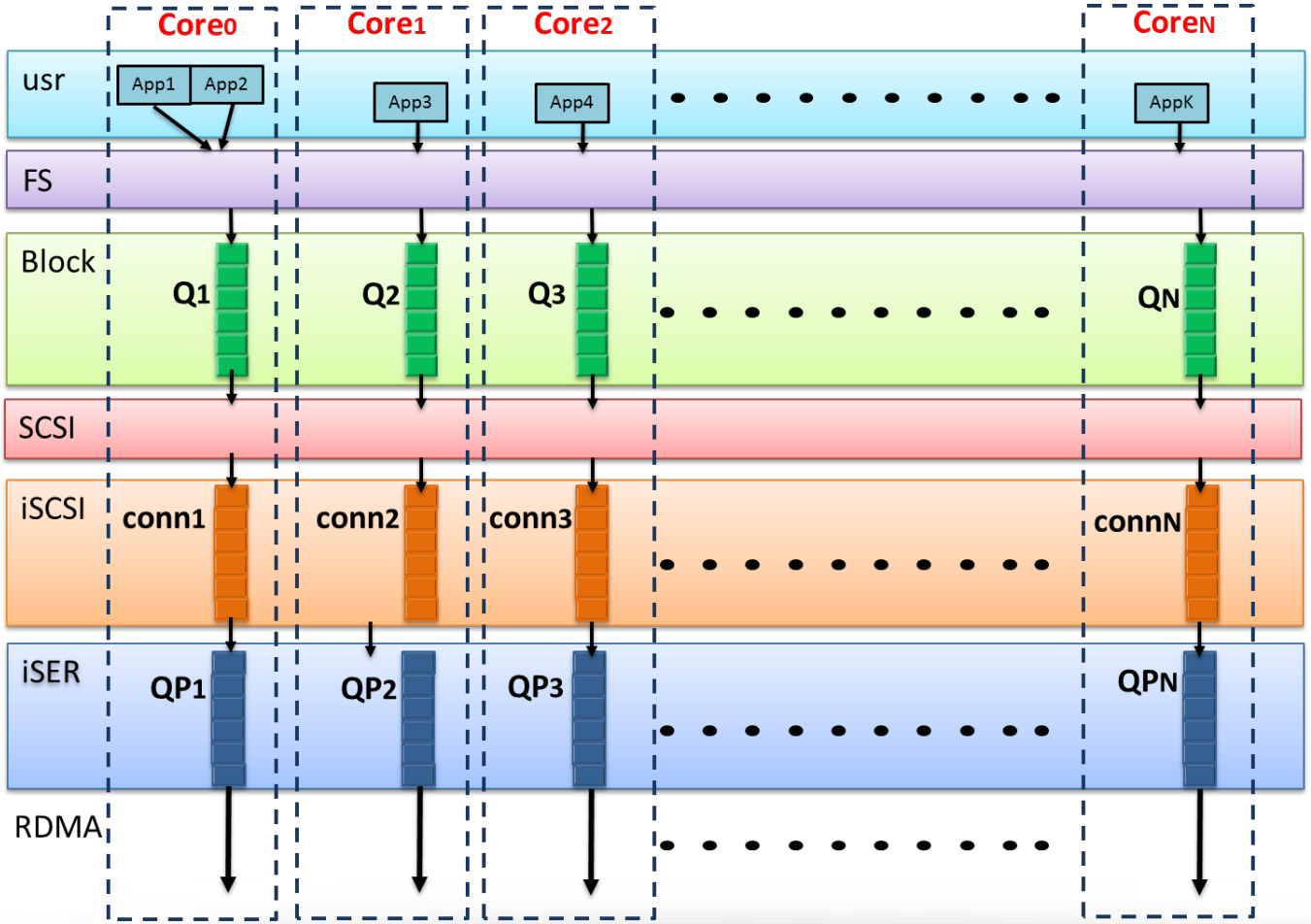
```
$ cat /proc/interrupts | grep mlx | awk {'print $NF'}
mlx4-comp-0@pci:0000:08:00.0
mlx4-comp-1@pci:0000:08:00.0
mlx4-comp-2@pci:0000:08:00.0
mlx4-comp-3@pci:0000:08:00.0
mlx4-comp-4@pci:0000:08:00.0
mlx4-comp-5@pci:0000:08:00.0
mlx4-comp-6@pci:0000:08:00.0
mlx4-comp-7@pci:0000:08:00.0
mlx4-comp-8@pci:0000:08:00.0
```

Multi-Queue Adoption

- iSCSI specifications states some session-wide command ordering constraints
  - “Command numbering is session-wide and is used for ordered command delivery over multiple connections”
  - “On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN”
  - “Responses in transit from the target to the initiator are numbered. The StatSN (Status Sequence Number) is used for this purpose. StatSN is a counter maintained per connection.”
- Adoption: Implement Multiple Connections per Session (MCS).

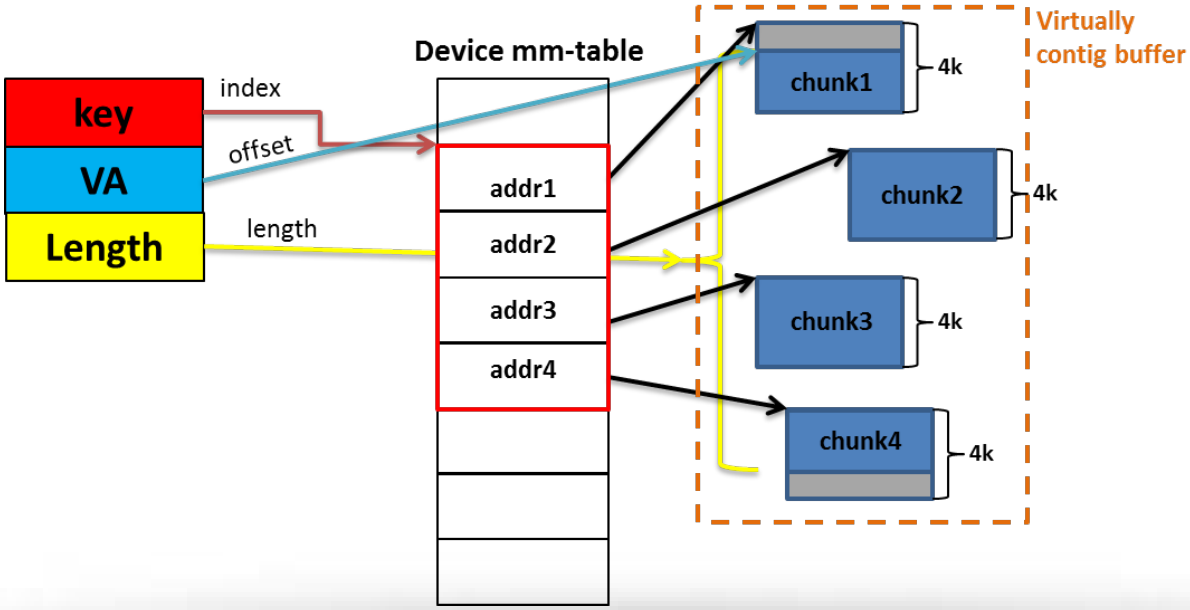
Multi-Queue Adoption

Apps 1..K accessing a single iSCSI block device:



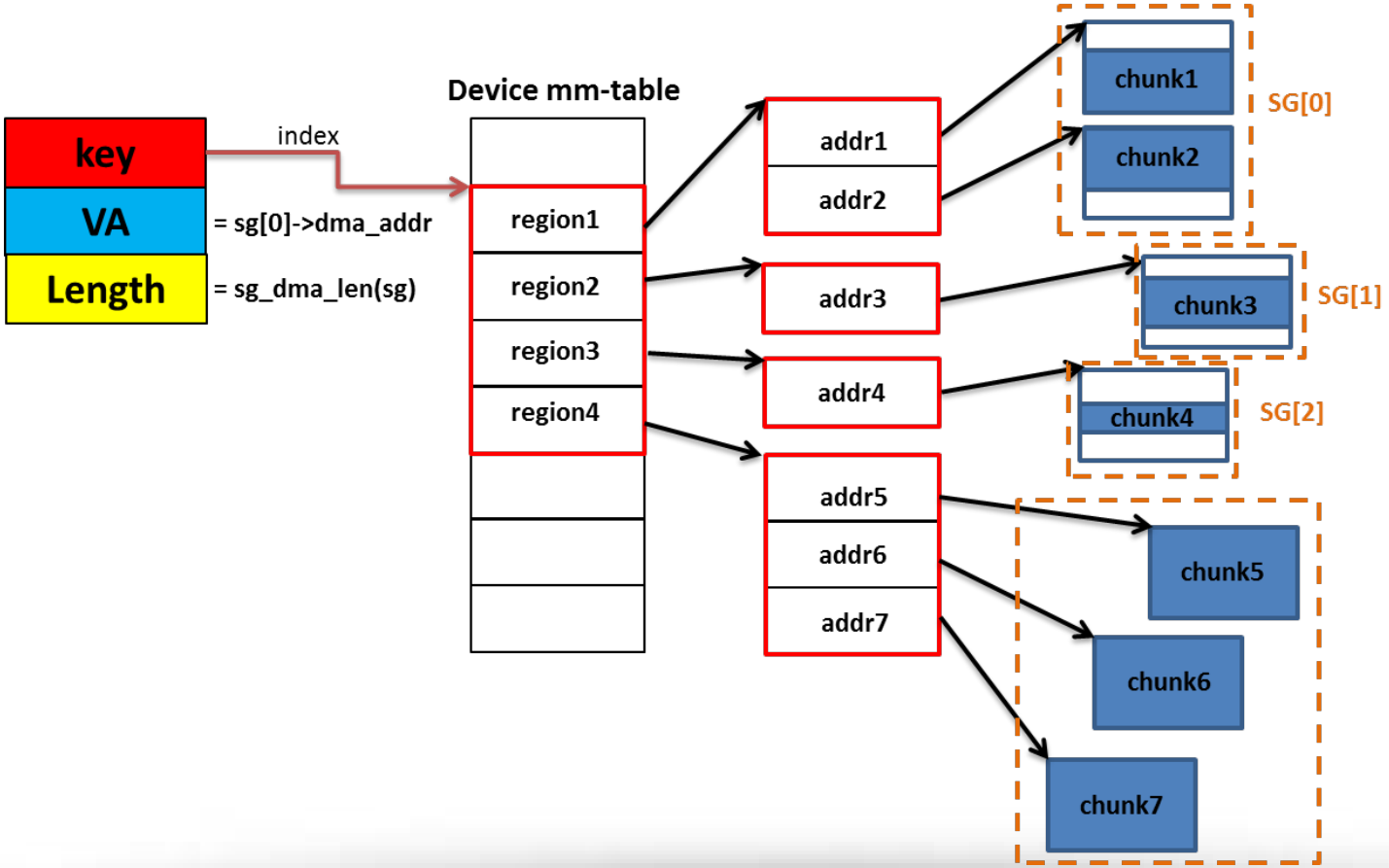
Indirect Fast Memory Registration

- Memory registration procedure can be done very fast for privileged users but has some well-known alignment constraints
- In order to perform a fast registration of a scattered list one must make sure:
  - List has one element which is physically contiguous OR,
  - Scattered elements are in the same size (nicely page aligned)



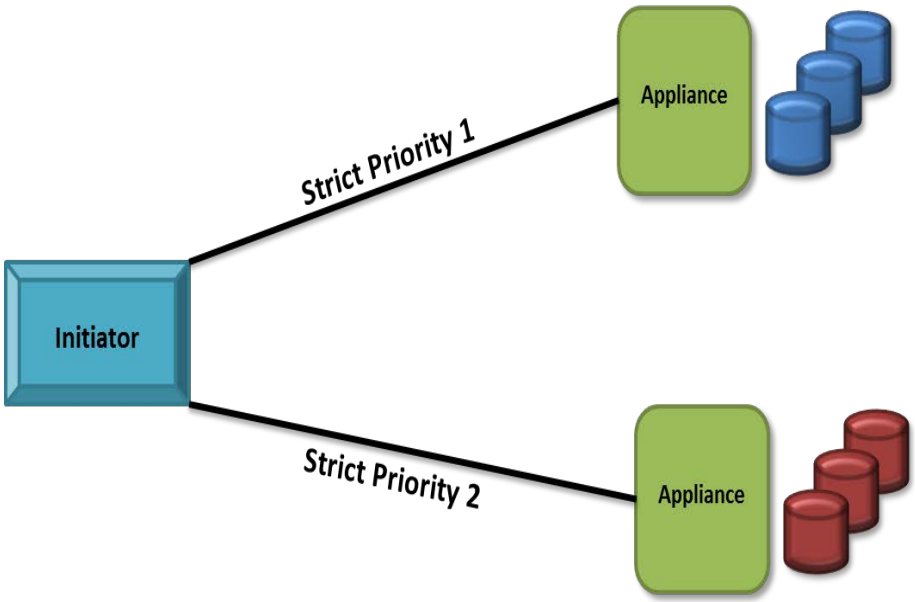
Indirect Fast Memory Registration

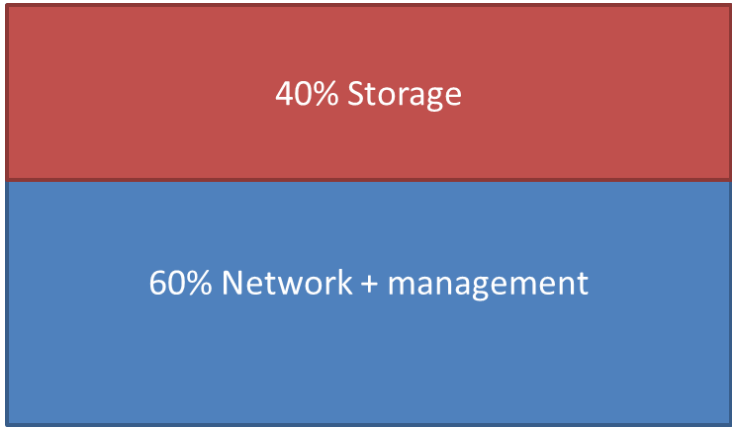
- Next generation HCAs (such as ConnectIB) allow users to register also “unaligned” scatter lists



QoS

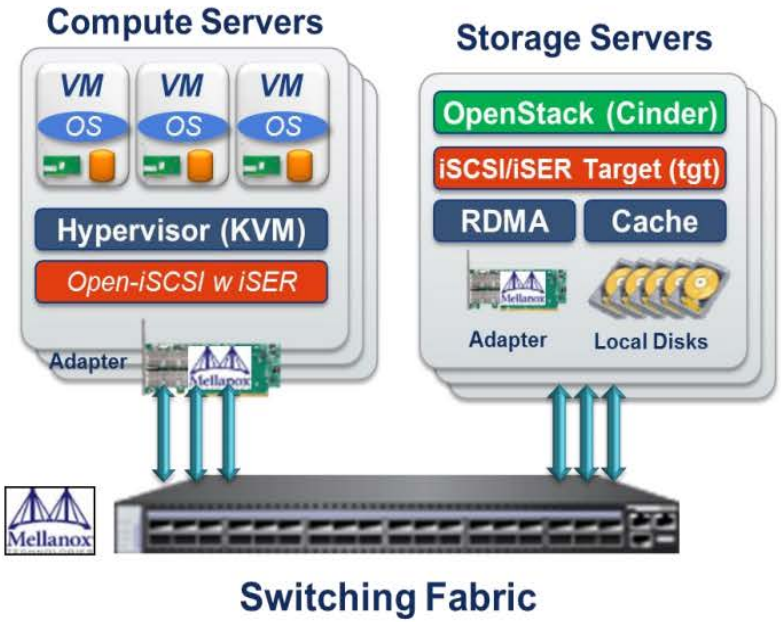
- Motivations:
  - Consolidated fabric – ensure storage traffic
  - Priorities between storage
- Solution: Inherent IP ToS/TC

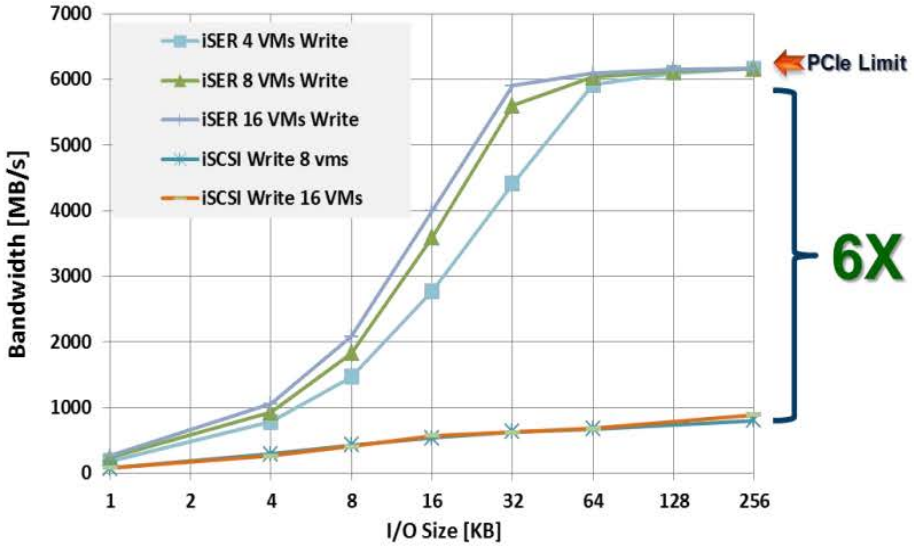




Applications & Deployments

Mellanox OpenStack and SDN Benefits





iSER data-mover accelerates:

- **Storage access**
- **VM migration**
- **Data/VM replication**

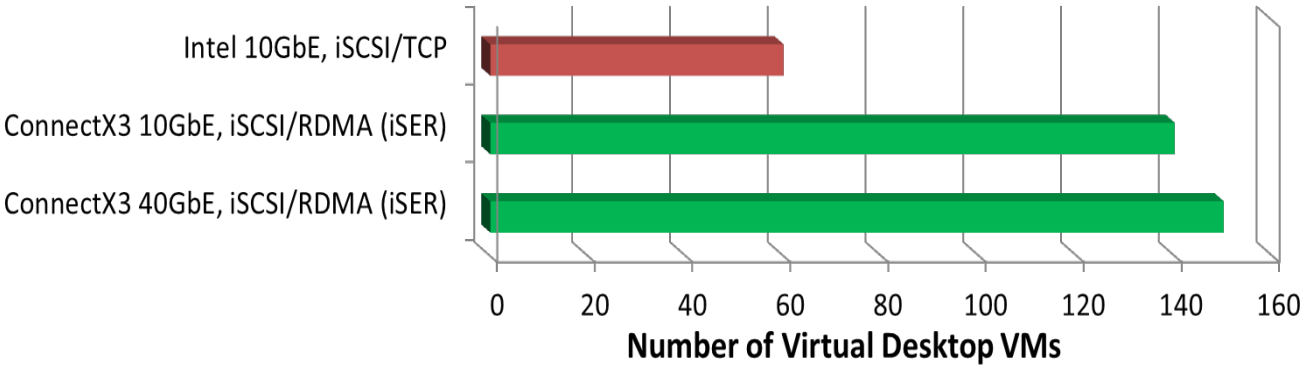
iSER in OpenStack – Cinder Support

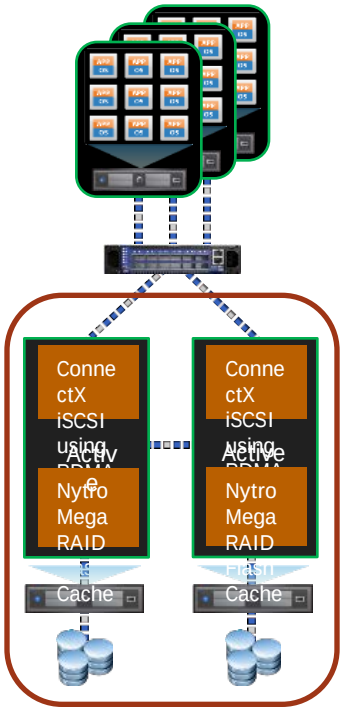
- Built-in components and management (Open-iSCSI, tgt target, Cinder)
- RDMA is already inbox
  - and used by our OpenStack customers !
- Simple: set “allow_rdma = true”

```
$ cat /etc/cinder/cinder.conf
…
allow_rdma = true
iscsi_ip_address = 192.168.52.45
```

Maximize VDI Efficiency over RDMA

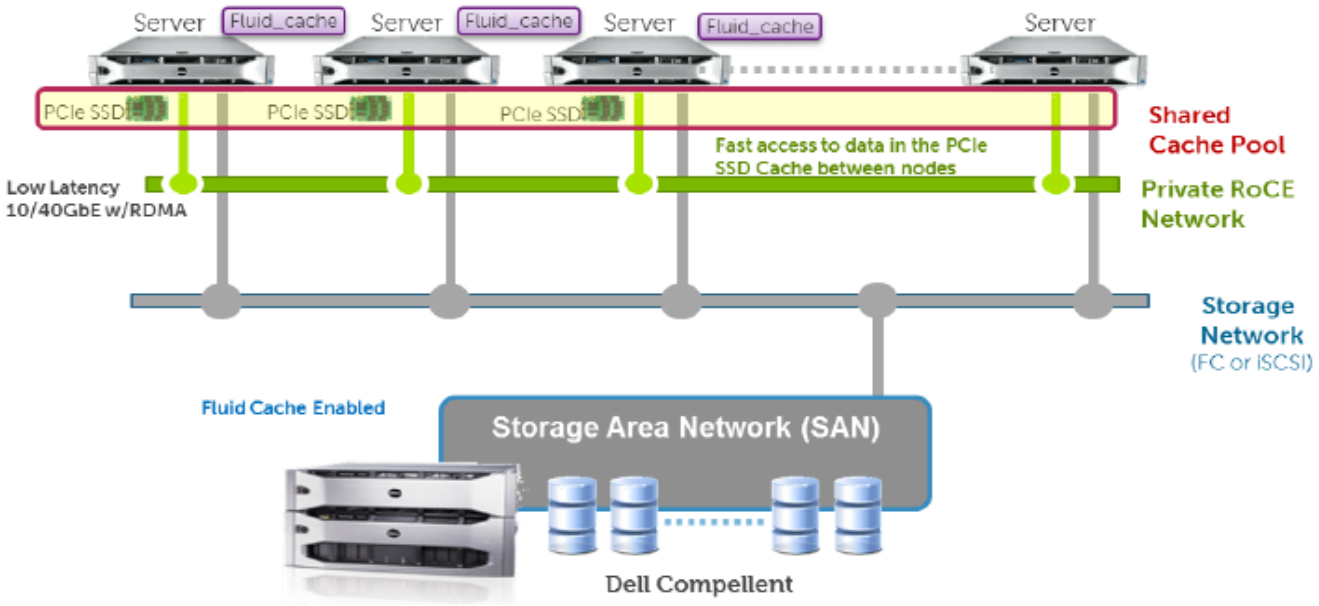
- RDMA eliminates storage bottlenecks in VDI deployments
  - Mellanox ConnectX®-3 with RoCE accelerates the access to cache over RDMA
  - 150 Virtual desktops over RoCE vs. 60 virtual desktops over TCP/IP





Dell Fluid Cache for SAN

- The Fluid cache solution is implemented over ESX 5.5 iSER initiator



Questions?

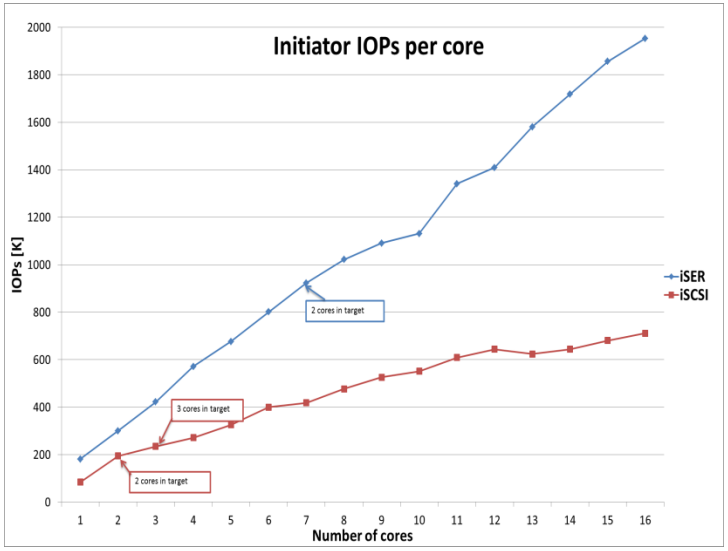
iSCSI Extensions for RDMA

Updates and news

END

Backup

Graphs Testing environment



- **Note:**

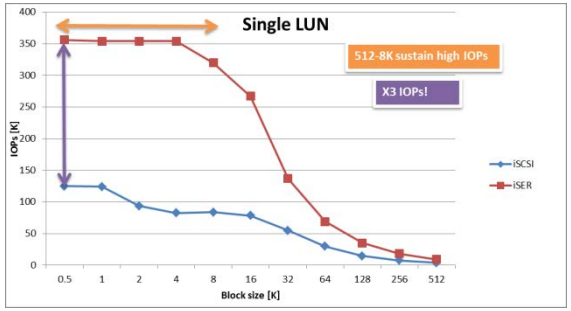
There are two fixes/enhancements added to kernel
In order to achieve the performance results.

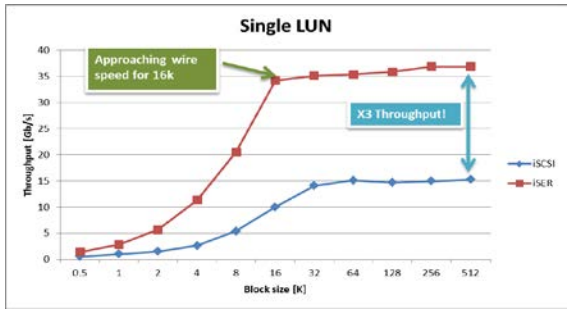
- Shared MSI-X vectors in mlx4_core
- PER-CPU completion contexts in iSER

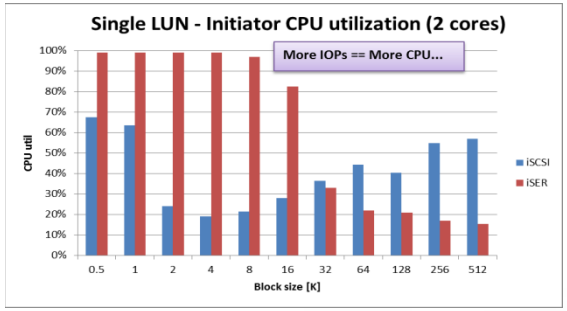
These patches have not maid it mainline yet.

- **HW:**
  - Initiator: 16 cores Intel(R) Xeon(R) @ 2.60GHz
  - Target: 2 cores Intel(R) Xeon(R) @ 2.60GHz
  - ConnectX-3
  - Single 40GE link
- **SW:**
  - RedHat-7.0 + 3.16.0 (Initiator & Target)
  - Target: TGTD
  - fio version: 2.0.13
- **Settings:**
  - MSIX interrupt vectors spread across involved cores.
  - Block Layer settings:
    - scheduler=noop
    - rq_affinity=1
    - add_random=0
    - nomerges=2
  - Default iSCSI settings (ImmediateData=Yes, InitialR2T=No)
    - cmd_per_lun=32
    - can_queue=113
  - Backend: 16 NULL devices (1 LUN per target)
  - IO pattern: randread
  - 2 IO threads per device

Graphs Testing environment

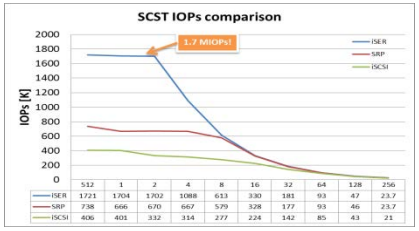


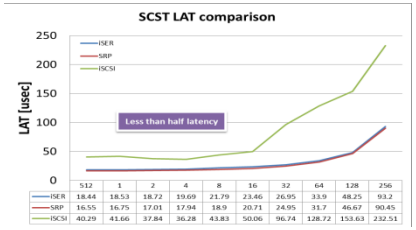


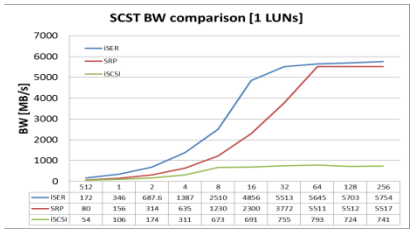


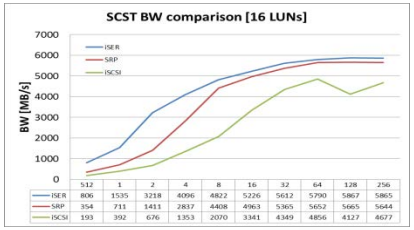
- **HW:**
  - 2 cores Intel(R) Xeon(R) @ 2.60GHz
  - ConnectX-3
  - Single 40GE link
- **SW:**
  - RedHat-7.0 + 3.16.0 (Initiator & Target)
  - Target: TGTD
  - fio version: 2.0.13
- **Settings:**
  - MSIX interrupt vectors spread across involved cores.
  - Block Layer settings:
    - scheduler=noop
    - rq_affinity=1
    - add_random=0
    - nomerges=2
  - Default iSCSI settings (ImmediateData=Yes, InitialR2T=No)
    - cmd_per_lun=32
    - can_queue=113
  - Backend: single NULL device
  - IO pattern: randrw
  - Single thread

Graphs Testing environment



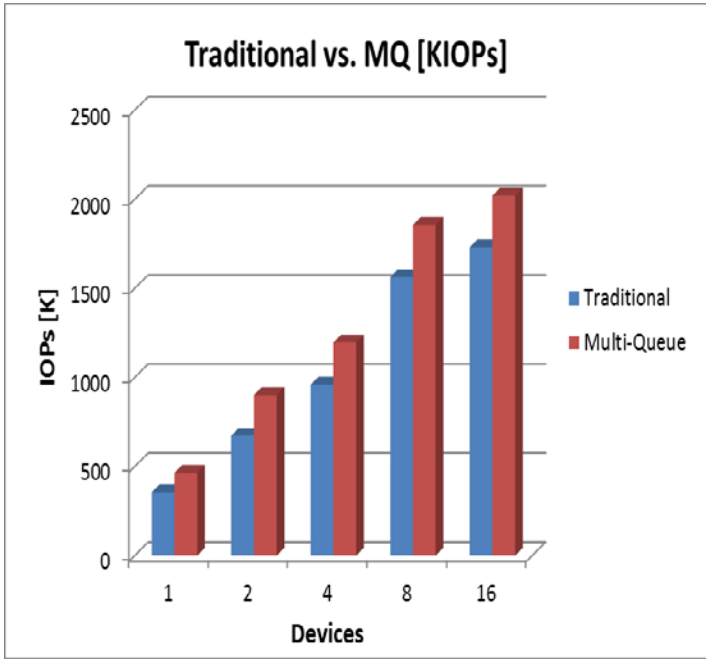






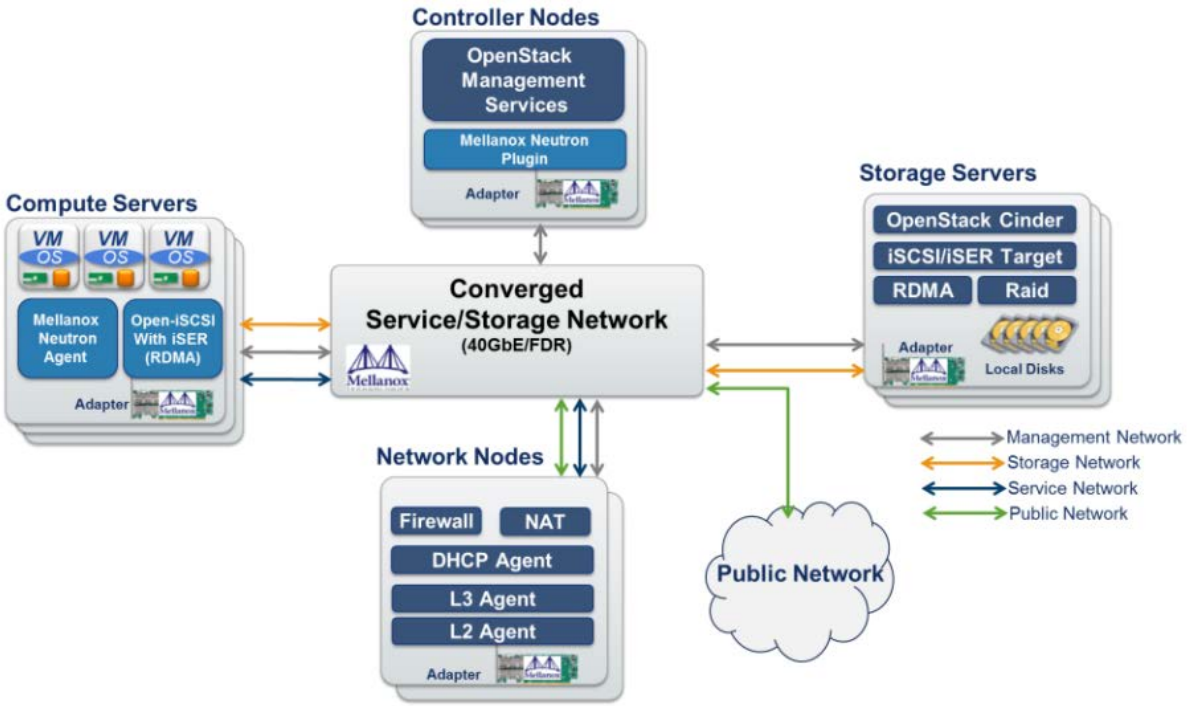
- **HW:**
  - Initiator: 16 cores Intel(R) Xeon(R) @ 2.60GHz
  - Target: 8 cores Intel(R) Xeon(R) @ 2.60GHz
  - ConnectX-3
  - Single IB-FDR link
- **SW:**
  - RedHat-7.0 + 3.16.0 (Initiator & Target)
  - Target: SCST (iser branch)
  - fio version: 2.0.13
- **Settings:**
  - MSIX interrupt vectors spread across involved cores.
  - Block Layer settings:
    - scheduler=noop
    - rq_affinity=1
    - add_random=0
    - nomerges=2
  - Default iSCSI settings (ImmediateData=Yes, InitialR2T=No)
    - cmd_per_lun=32
    - can_queue=113
  - Backend: 16 NULL devices (1 LUN per target)
  - IO pattern: randread
  - 2 IO threads per device

Graphs Testing environment



- **HW:**
  - Initiator: 16 cores Intel(R) Xeon(R) @ 2.60GHz
  - Target: 16 cores Intel(R) Xeon(R) @ 2.60GHz
  - ConnectIB
  - Single IB-FDR link
- **SW:**
  - RedHat-7.0 + 3.16.0 (Initiator & Target)
  - Target: LIO
  - fio version: 2.0.13
- **Settings:**
  - MSIX interrupt vectors spread across involved cores.
  - Block Layer settings:
    - scheduler=noop
    - rq_affinity=1
    - add_random=0
    - nomerges=2
  - Default iSCSI settings (ImmediateData=Yes, InitialR2T=No)
    - cmd_per_lun=32
    - can_queue=113
  - Backend: 16 NULL devices (1 LUN per target)
  - IO pattern: randread
  - 2 IO threads per device

Mellanox OpenStack and SDN Benefits



iSER's RDMA efficient data movement in OpenStack:

- Delivers **6X** better data throughput
- Simultaneously reducing CPU utilization by up to **80%**