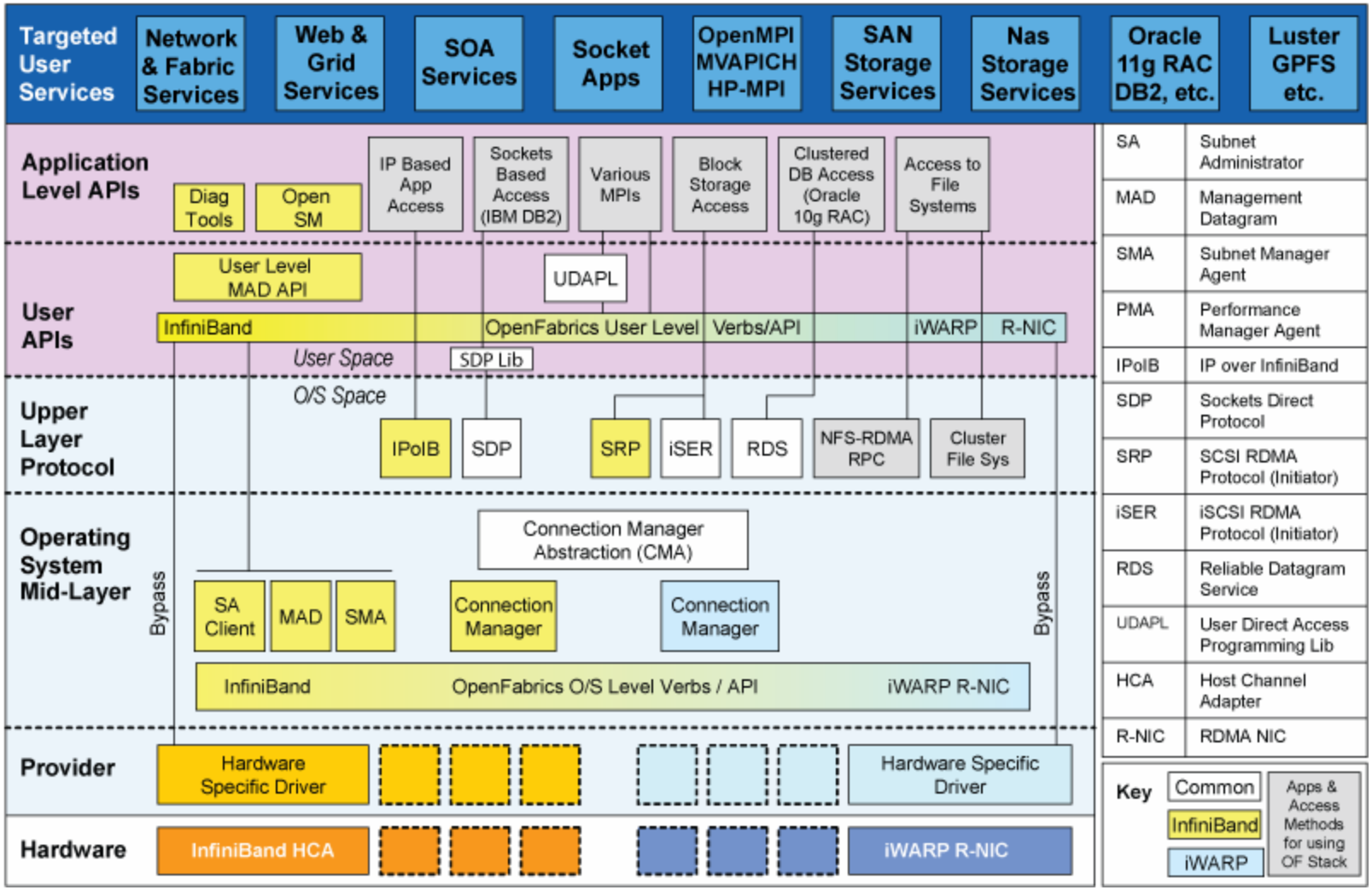
InfiniBand Linux SW Stack

MLNX_OFED

OpenFabrics Enterprise Distribution (OFED)

- Open Fabrics Enterprise Distribution (OFED) is a complete SW stack for RDMA capable devices.
- Contains low level drivers, core, Upper Layer Protocols (ULPs), Tools and documents
- Available on OpenFabrics.org or as a Mellanox supported package at:
  - http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=26&menu_section=34
- Mellanox OFED is a single Virtual Protocol Internconnect (VPI) software stack based on the OFED stack
  - Operates across all Mellanox network adapters
  - Supports:
    - 10, 20 and 40Gb/s InfiniBand (SDR, DDR and QDR IB)
    - 10Gb/s Ethernet (10GigE)
    - Fibre Channel over Ethernet (FCoE)
    - 2.5 or 5.0 GT/s PCI Express 2.0

The SW stack


Targeted User Services
Network & Fabric Services
Web & Grid Services
SOA Services
Socket Apps
OpenMPI MVAPICH HP-MPI
SAN Storage Services
Nas Storage Services
Oracle 11g RAC DB2, etc.
Luster GPFS etc.
Application Level APIs
Diag Tools
Open SM
IP Based App Access
Sockets Based Access (IBM DB2)
Various MPIs
Block Storage Access
Clustered DB Access (Oracle 10g RAC)
Access to File Systems
User APIs
User Level MAD API
UDAPL
InfiniBand
OpenFabrics User Level Verbs/API
iWARP
R-NIC
User Space
SDP Lib
O/S Space
Upper Layer Protocol
IPoIB
SDP
SRP
iSER
RDS
NFS-RDMA RPC
Cluster File Sys
Operating System Mid-Layer
Bypass
Connection Manager Abstraction (CMA)
SA Client
MAD
SMA
Connection Manager
Connection Manager
Bypass
InfiniBand
OpenFabrics O/S Level Verbs / API
iWARP R-NIC
Provider
Hardware Specific Driver
Hardware Specific Driver
Hardware
InfiniBand HCA
iWARP R-NIC
SA Subnet Administrator
MAD Management Datagram
SMA Subnet Manager Agent
PMA Performance Manager Agent
IPoIB IP over InfiniBand
SDP Sockets Direct Protocol
SRP SCSI RDMA Protocol (Initiator)
iSER iSCSI RDMA Protocol (Initiator)
RDS Reliable Datagram Service
UDAPL User Direct Access Programming Lib
HCA Host Channel Adapter
R-NIC RDMA NIC
Key
Common
InfiniBand
iWARP
Apps & Access Methods for using OF Stack

Mellanox OFED

- Mellanox OFED is delivered as an ISO image.
- The ISO image contains both source code and binary RPMs for selected Linux distributions.
- It also contains installation scripts called mlnxofedinstall. The install script performs the necessary steps to accomplish the following:
  - Discovers the currently installed kernel
  - Uninstalls any IB stacks that are part of the standard operating system distribution or other commercial IB stacks
  - Installs the Mellanox OFED binary RPMs if they are available for the current kernel
  - Identifies the currently installed IB HCA and perform the required firmware updates

MLNX_OFED Installation

- Pre-built RPM install.
  - 1. mount -o rw,loop /work/MLNX_OFED_LINUX-1.4-sles10_sp1_sp2.iso /mnt
  - 2. cd /mnt
  - 3. ./mlnxofedinstall
- Building RPMs for un-supported kernels.
  - 1. mount -o rw,loop MLNX_OFED_LINUX-1.4-rhel5.3.iso /mnt
  - 2. cd /mnt/src
  - 3. cp OFED-1.4.tgz /root (this is the original OFED distribution tarball)
  - 4. tar zxvf OFED-1.4.tgz
  - 5. cd OFED-1.4
  - 6. copy ofed.conf to OFED-1.4 directory
  - 7. ./install.pl -c ofed.conf

Configuration

Loading and Unloading the IB stack

- /etc/infiniband/openib.conf controls boot time configuration

```
# Start HCA driver upon boot
ONBOOT=yes

# Load IPoIB
IPOIB_LOAD=yes
```

- Manually start and stop the stack once the node has booted
  - /etc/init.d/openibd start|stop|restart|status

OpenSM Subnet Manager

OpenSM - Features

- OpenSM (osm) is an Infiniband compliant subnet manger.
- Included in Linux Open Fabrics Enterprise Distribution.
- Ability to run several instance of osm on the cluster in a Master/Slave(s) configuration for redundancy.
- Partitions (p-key) support
- QoS support
- Enhanced routing algorithms:
  - Min-hop
  - Up-down
  - Fat-tree
  - LASH
  - DOR

Running OpenSm

- Command line
  - Default (no parameters)
    - Scans and initializes the IB fabric and will occasionally sweep for changes
  - opensm –h for usage flags
    - E.g. to start with up-down routing: opensm –-routing_engine updn
  - Run is logged to two files:
    - /var/log/messages – opensm messages, registers only general major events
    - /var/log/opensm.log - details of reported errors.
    - /var/log/opensm-subnet.lst – Topology as configured by OSM
- Start on Boot
  - As a daemon:
    - /etc/init.d/opensmd start|stop|restart|status
    - Just like any other service, “chkconfig opensmd on” for example
- SM detection
  - /etc/init.d/opensd status
    - Shows opensm runtime status on a machine
  - sminfo
    - Shows master and standby subnets running on the cluster

OpenSM Command Line parameters

- A few important command line parameters:

-c, --cache-options. Write out a list of all tunable OpenSM parameters, including their current values from the command line as well as defaults for others, into the file /var/cache/opensm. This file can then be modified to change OSM parameters, such as HOQ (Head of Queue timer).

-g, --guid This option specifies the local port GUID value with which OpenSM should bind. OpenSM may be bound to 1 port at a time. This option is used if the SM needs to bind to Port 2 of an HCA.

-R, --routing_engine This option chooses routing engine instead of Min Hop algorithm (default). Supported engines: updn, file, ftree, lash

-x, --honor_guid2lid. This option forces OpenSM to honor the guid2lid file, when it comes out of Standby state, if such file exists under /var/cache/opensm

-V This option sets the maximum verbosity level and forces log flushing.

Routing Algorithms

- Min Hop algorithm (DEFAULT)
  - Based on the minimum hops to each node where the path length is optimized.
- UPDN unicast routing algorithm
  - Based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure Fat Tree, and a deadlock may occur due to a loop in the subnet.
    - Root GUID list file can be specified using the –a option
- Fat Tree unicast routing algorithm
  - This algorithm optimizes routing for a congestion-free “shift” communication pattern. It should be chosen if a subnet is a symmetrical Fat Tree of various types, not just a K-ary-N-Tree: non-constant K, not fully staffed, and for any CBB ratio. Similar to UPDN, Fat Tree routing is constrained to ranking rules.
    - Root GUID list file can be specified using the –a option
- Addition algorithms
  - LASH - Uses InfiniBand virtual layers (SL) to provide deadlock-free shortest-path routing.
  - DOR. This provides deadlock free routes for hypercube and mesh clusters
  - Table Based. A file method which can load routes from a table.

OFED Tools

IBDIAG and other OFA tools

Single Node	SRC/DST Pair	Network
ibv_devinfo	Ibdiagpath	Ibdiagnet
ibstat	ibtracert	ibnetdiscover
Ibportstate	ibv_rc_pingpong	ibhosts
ibroute	ibv_srq_pingpong	Ibswitches
smpquery	ibv_ud_pingpong	saquery
perfquery	ib_send_bw	sminfo
	ib_write_bw	smpdump

Node Based Tools

Determine if driver is loaded

- /etc/init.d/openibd status
  - HCA driver is loaded
  - Configured devices
    - Ib0
    - Ib1
    - OFED modules are loaded
      - ib_ipoib
      - ib_mthca
      - ib_core
      - ib_srp

Determine modules that are loaded

- lsmod
  - ib_core
  - ib_mthca
  - ib_mad
  - ib_sa
  - ib_cm
  - ib_uverbs
  - ib_srp
  - ib_ipoib
- modinfo ‘module name’
  - List all parameters accepted by the module
  - Module parameter can be added to /etc/modprobe.conf

HCA Device information

- ibstat
  - displays basic information obtained from the local IB driver.
  - Normal output includes Firmware version, GUIDS, LID, SMLID, port state, link width active, and port physical state.
  - Has options to list CAs and/or Ports.
- ibv_devinfo
  - Reports similar information to ibstat
  - Also includes PSID and an extended verbose mode (-v).
- /sys/class/infiniband
  - File system which reports driver and other ULP information.
    - e.g. [root@ibd001 /]# cat /sys/class/infiniband/mlx4_0/board_id
      MT_04A0110002

HCA Firmware management

- Determine HCA firmware version
  - /usr/bin/ibv_devinfo
  - /usr/bin/mstflint –d mlx4_0 v
  - /usr/bin/mstflint –d 07:00.0 q
- Burn new HCA firmware
  - usr/bin/mstflint [switches] <command > [parameters…]
  - /usr/bin/mstflint –d mlx4_0 –i fw.bin b

Switch Firmware management

- Determine IS4 firmware version
  - /usr/bin/flint –d lid-6 q
- Burn new IS4 firmware
  - /usr/bin/flint –d lid-6 –i fw.img b

Note: Mellanox FW Tools (MFT) package that contains flint tool can be found at: http://www.mellanox.com/content/pages.php?pg=firmware_HCA_FW_update

Node management utilities

- perfquery
  - Obtains and/or clears the basic performance and error counters from the specified node
  - Can be used to check port counters of any port in the cluster using ‘perfquery <lid> <port number>’
- ibportstate
  - Query, change state (i.e. disable), or speed of Port
    - ibportstate 38 1 query
- ibroute
  - Dumps routes within a switch
- smpquery
  - Dump SMP query parameters, including:
    - nodeinfo, nodedesc, switchinfo, pkeys, sl2vl, vlarb, guids

Performance tests

- Run performance tests
  - /usr/bin/ib_write_bw
  - /usr/bin/ib_write_lat
  - /usr/bin/ib_read_bw
  - /usr/bin/ib_read_lat
  - /usr/bin/ib_send_bw
  - /usr/bin/ib_send_lat
- Usage
  - Server: <test name> <options>
  - Client: <test name> <options> <server IP address>

Note: Same options must be passed to both server and client. Use –h for all options.

Collecting debug information

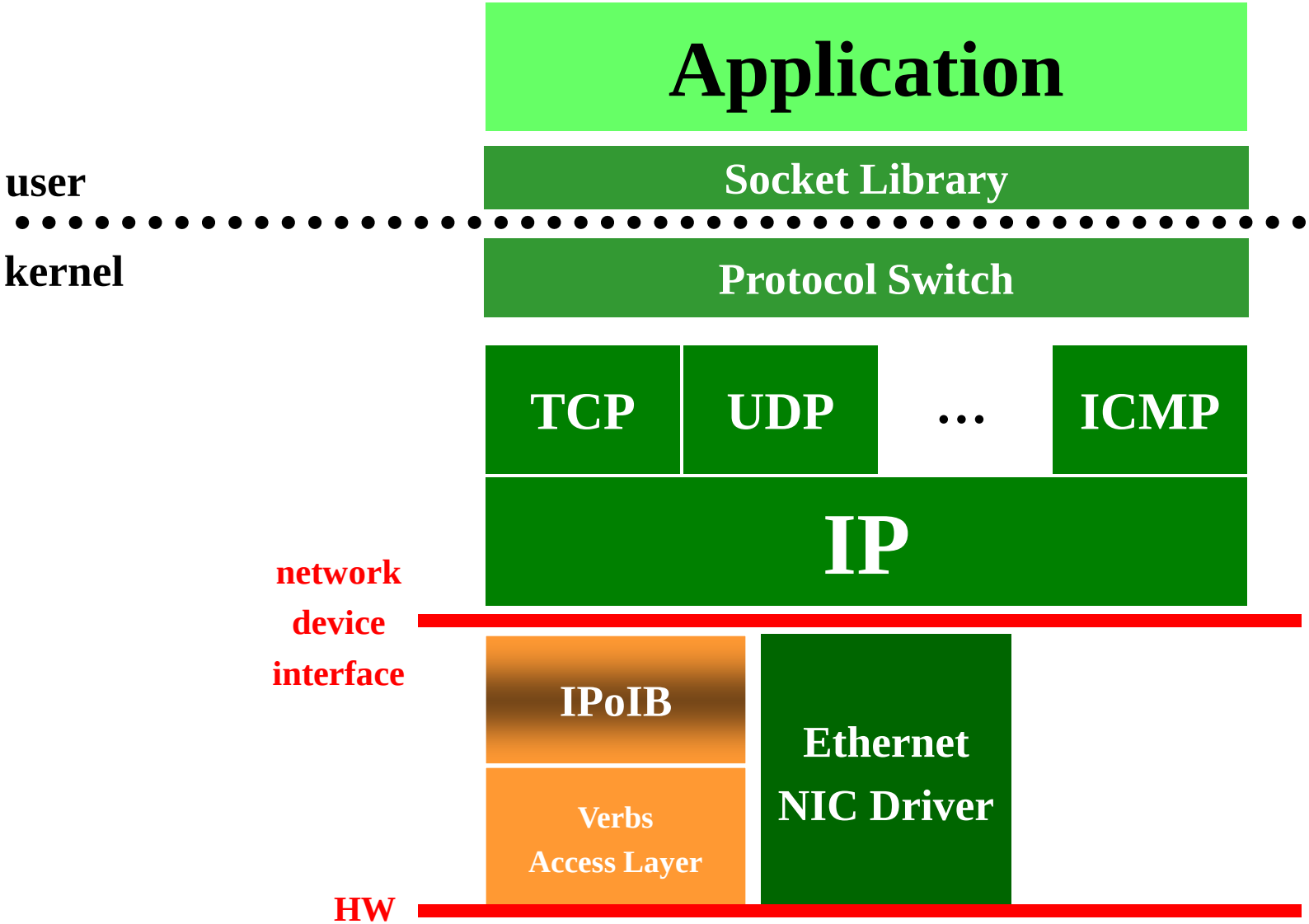
- Collect debug information if driver load fails
  - mstregdump
    - Internal register dump is produced on standard output
    - Store it in file for analysis in Mellanox
    - Examples
      - mstregdump 13:00.0 > dumpfile_1.txt
      - mstregdump mthca > dumpfile_2.txt
  - mstvpd mthca0
  - /var/log/messages
    - tail –n 500 /var/log/messages > messages_1.txt
    - dmesg > dmesg_1.txt

IPoIB

IPoIB in a Nut Shell

- Encapsulation of IP packets over IB
- Uses IB as “layer two” for IP
  - Supports both UD service (up to 2KB MTU) and RC service (connected mode, up to 64KB MTU).
- IPv4, IPv6, ARP and DHCP support
- Multicast support
- VLANs support
- Benefits:
  - Transparency to the legacy applications
  - Allows leveraging of existing management infrastructure
- Specification state: IETF Draft

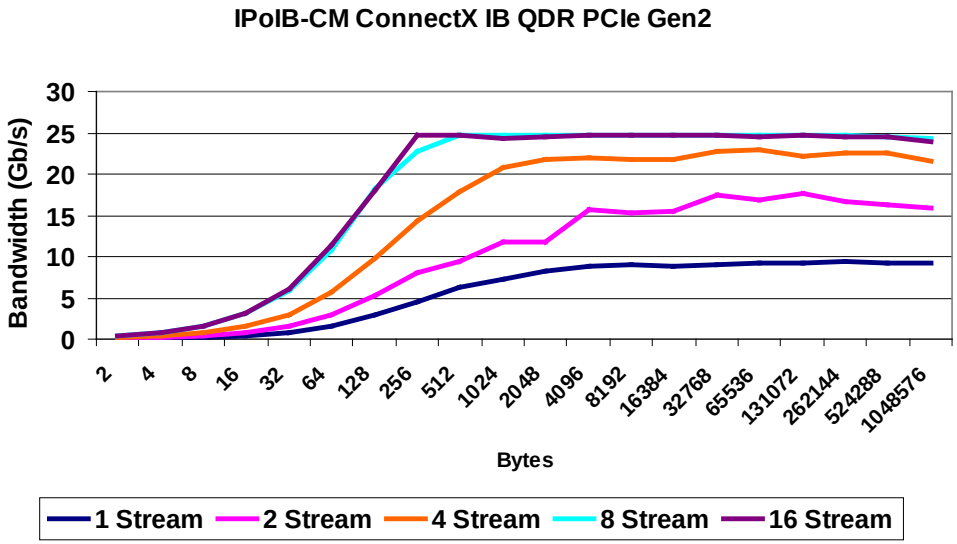
IPoIB in Generic Protocol Stack

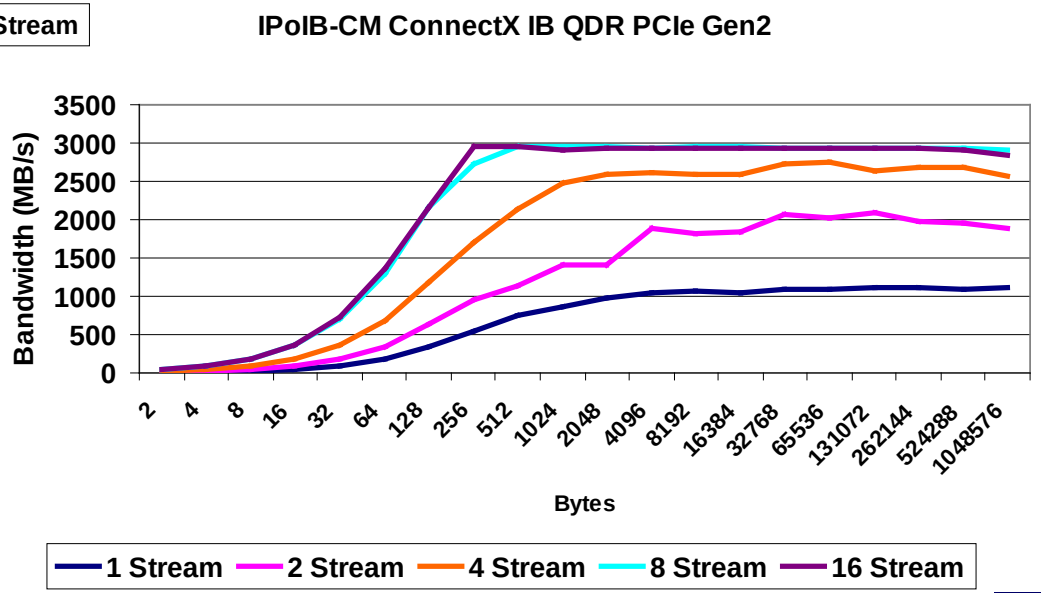


IPoIB Building Blocks

- Two modes: UD or CM (/sys/class/net/ib*/mode)
  - UD uses UD QP
    - Unreliable
    - Each destination described using AV
    - IPoIB MTU constrained by IB MTU
  - CM uses RC QP
    - Allows for large MTU
    - Better performance
- Destination is described by:
  - GID of destination port
  - Destination QP
  - GID + QP used as MAC address
- Uses multicast tree for address resolution
- Uses SA to get path record for node

IPoIB-CM ConnectX Performance - IB QDR PCIe Gen2





IPoIB Mode Settings

- IPoIB runs in two modes
  - Datagram mode using UD transport type
  - Connected mode using RC transport type
- Default mode is Connected Mode
  - This can be changed by editing /etc/infiniband/openib.conf and setting ‘SET_IPOIB_CM=no’.
  - After changing the mode, you need to restart the driver by running:
    - **/etc/init.d/openibd restart**
  - To check the current mode used for out-going connections, enter:
    - **cat /sys/class/net/ib<n>/mode**

IPoIB Configuration

- Requires assigning an IP address and a subnet mask to each HCA port (like any other network adapter)
- The first port on the first HCA in the host is called interface ib0, the second port is called ib1, and so on.
- Configuration can be based on DHCP or on a static configuration
  - Modify /etc/sysconfig/network/ifcfg-ib0

```
DEVICE=ib0
BOOTPROTO=static
IPADDR=10.10.0.1
NETMASK=255.255.255.0
NETWORK=10.10.0.0
BROADCAST=10.10.0.255
ONBOOT=yes
```

  - ifconfig ib0 10.10.0.1 up

MPI

MPI

- A message passing interface
- Used for point to point communication
  - MPI_I/SEND, MPI_I/RECV
- Used for collective operations:
  - MPI_AlltoAll, MPI_Reduce, MPI_barrier
- Other primitives
  - MPI_Wait, MPI_Walltime
- MPI Ranks are IDs assigned to each process
- MPI Communication Groups are subdivisions a job node used for collectives
- Two MPI stacks are included in this release of OFED:
  - MVAPICH 1.1.0
  - Open MPI 1.2.8
- This presentation will concentrate on MVAPICH-1.1.0

MPI Example

```
01:   MPI_Init(&argc,&argv);
02:   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
03:   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
04:
05:   MPI_Barrier(MPI_COMM_WORLD);
06:
07:   if(myid==0)
08:     printf("Passed first barrier\n");
09:
10:   srand(myid*1234);
11:   x = rand();
12:
13:   printf("I'm rank %d and my x is 0x%08x\n",myid, x);
14:
15:   MPI_Barrier(MPI_COMM_WORLD);
16:
17:   MPI_Bcast(&x,1,MPI_INT,0,MPI_COMM_WORLD);
18:
19:   if(myid == 1)
20:      printf("My id is rank 1 and I got 0x%08x from rank 0\n", x);
21:
22:   if(myid == 2)
23:       printf("My id is rank 2 and I got 0x%08x from rank 1\n", x);
24;
25:   MPI_Finalize();
```

Compiling

- mpicc is used to compiling mpi applications
- mpicc is equivalent to gcc
- mpicc includes all the gcc flags needed for compilation
  - Head files paths
  - Libraries paths
- To see real compilation flag run: mpicc –v
- MPI application can be shared or dynamic

Launching MPI jobs using mpirun_rsh

- Prerequisites for Running MPI:
  - The mpirun_rsh launcher program requires automatic login (i.e., password-less) onto the remote machines.
  - Must also have an /etc/hosts file to specify the IP addresses of all machines that MPI jobs will run on.
  - Make sure there is no loopback node specified (i.e. 127.0.0.1) in the /etc/hosts file or jobs may not launch properly.
  - Details on this procedure can be found in Mellanox OFED User's manual
- Basic format:
  - mpirun_rsh –np *procs node1 node2 node3 BINARY*
- Other flags:

  -show: show only

  -paramfile: environment variables

  -hostfile: list of host

  -ENV=VAL (i.e. VIADEV_RENDEZVOUS_THRESHOLD=8000)

Launching MPI jobs using mpirun_rsh (cont…)

- mpirun_rsh -show -np 3 mtilab32 mtilab33 mtilab33 ./dcest:

- command: /usr/bin/ssh mtilab32 cd /home/rabin/tmp; /usr/bin/env MPIRUN_MPD=0 MPIRUN_HOST=mtilab32.mti.mtl.com MPIRUN_PORT=33111 MPIRUN_PROCESSES='mtilab32:mtilab33:mtilab33:' MPIRUN_RANK=0 MPIRUN_NPROCS=3 MPIRUN_ID=26974 DISPLAY=localhost:12.0 ./dcest
- command: /usr/bin/ssh mtilab33 cd /home/rabin/tmp; /usr/bin/env MPIRUN_MPD=0 MPIRUN_HOST=mtilab32.mti.mtl.com MPIRUN_PORT=33111 MPIRUN_PROCESSES='mtilab32:mtilab33:mtilab33:' MPIRUN_RANK=1 MPIRUN_NPROCS=3 MPIRUN_ID=26974 DISPLAY=localhost:12.0 ./dcest
- command: /usr/bin/ssh mtilab33 cd /home/rabin/tmp; /usr/bin/env MPIRUN_MPD=0 MPIRUN_HOST=mtilab32.mti.mtl.com MPIRUN_PORT=33111 MPIRUN_PROCESSES='mtilab32:mtilab33:mtilab33:' MPIRUN_RANK=2 MPIRUN_NPROCS=3 MPIRUN_ID=26974 DISPLAY=localhost:12.0 ./dcest

Eager Mode

- Simple send/receive buffers
- Used for vbuf transfers
- Used once vbufs are exhausted
- WQE will point to vbuf buffers
  - Different vbuf pool than fast path
- Eager mode is transparent to user

Eager vs. Rendezvous modes

- Two modes to transfer buffers between two ranks
- Eager is using send/receive
- Rendezvous is using RDMA (zero copy)
- Switch from Eager to Rendezvous is made once certain threshold is reached
  - Control through VIADEV_RENDEZVOUS_THRESHOLD

Cheat Sheet

- All binaries are under MPIHOME/bin
  - Default /usr/mpi/gcc/mvapich-1.1.0/bin/
- mpirun_rsh –np *num_proc node1 node2 … BINARY PARAMS*

  -debug: open gdb (need display set)

  -show: show what mpi does

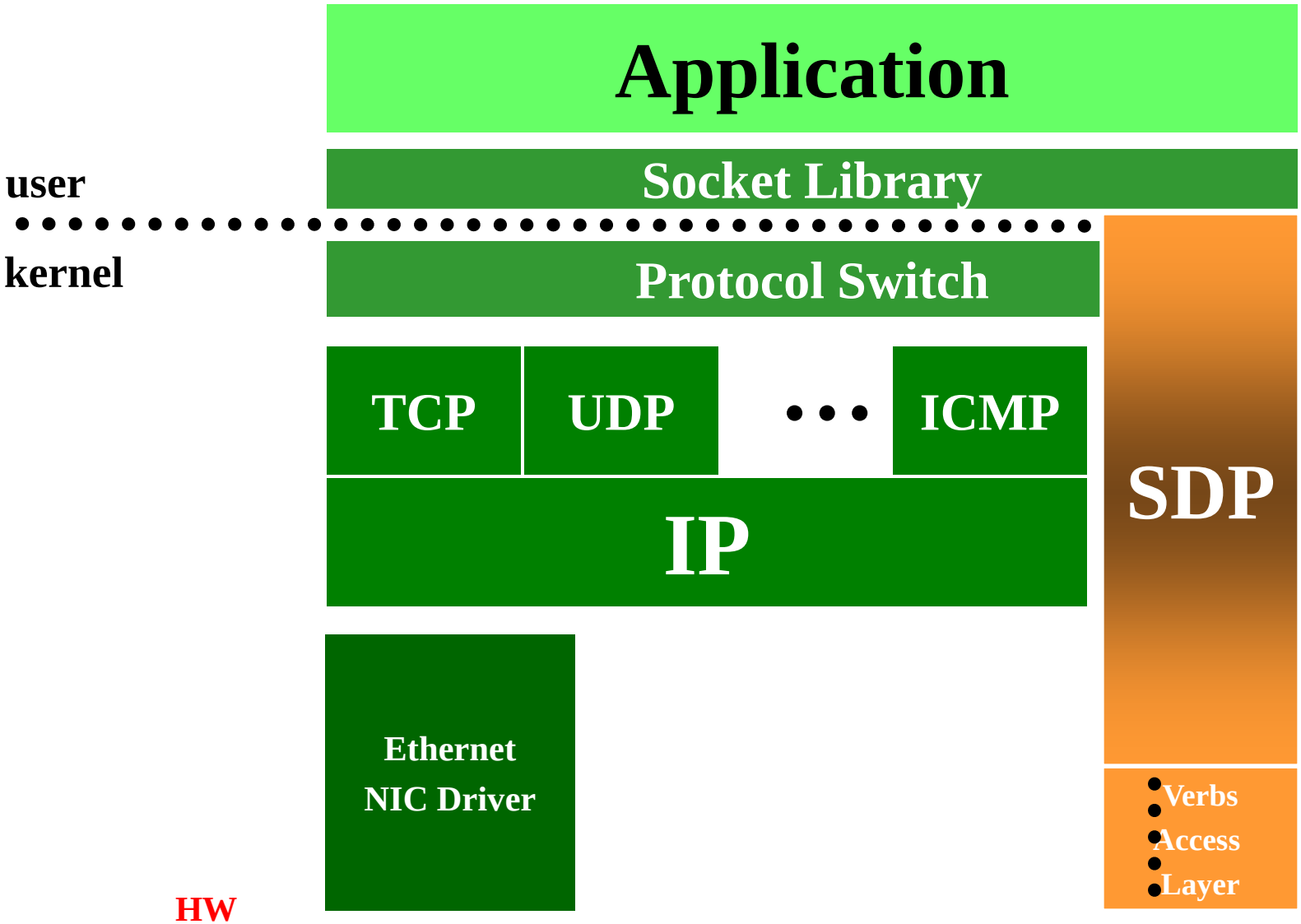
  -hostfile: node list
- mpicc –v: shows commands
- Environment Variables:
  - VIADEV_DEVICE=device name (def=InfiniHost0)
  - VIADEV_DEFAULT_MTU=mtu size (def=1024)
  - VIADEV_DEFAULT_SERVICE_LEVEL=sl to use in QP
  - VIADEV_DEFAULT_TIME_OUT=QP timeout
  - VIADEV_DEFAULT_RETRY_COUNT=RC retry count
  - VIADEV_NUM_RDMA_BUFFER=fast path array size (def=32 0=disabled)

SDP – Sockets Direct Protocol

SDP in a Nut Shell

- An InfiniBand byte-stream transport protocol that provides TCP stream semantics.
- Capable of utilizing InfiniBand's advanced protocol offload capabilities, SDP can provide lower latency, higher bandwidth, and lower CPU utilization than IPoIB running some sockets-based applications.
- Composed of a kernel module that implements the SDP as a new address-family/protocol-family, and a library that is used for replacing the TCP address family with SDP according to a policy.

SDP in Generic Protocol Stack (User)



SDP libdsp.so Library

- Dynamically linked library used for replacing the TCP address family with SDP according to a policy.
- ‘Hijacks’ socket calls and replaces the address family
- Library acts as a user-land socket switch

SDP in OFED Overview

- Linux TCP Socket implementation
  - Uses standard API
  - Socket type: STREAM
  - New socket family: AF_INET_SDP (set to 26)
- Implemented as a kernel module ib_sdp
- Implements BCopy and BZCopy operation (Zcopy in upcoming release)

Configuring SDP

- Loading kernel module
  - Automatic (on boot):
    - Edit /etc/infiniband/openib.conf:

```
SDP_LOAD=yes
```

    - Restart openibd
  - Manual

```
modprobe ib_sdp <_use_zcopy=[0|1] _src_zthresh=[value]>
```

- Change/create kernel application
  - Should use AF_INET_SDP STREAM sockets
  - Include sdp_inet.h

Usage – User Level configuration

- Using dynamically loaded libsdp library
  - Must set the following environment variables:

```
export LD_PRELOAD=/usr/[lib|lib64]/libsdp.so
export LIBSDP_CONFIG_FILE=/etc/libsdp.conf
```

  - Or… Inside the command line

```
env LD_PRELOAD='stack_prefix'/[lib|lib64]/libsdp.so
  LIBSDP_CONFIG_FILE='stack_prefix'/etc/libsdp.conf <program>
```

- Simplest usage
  - All sockets from AF_INET family of type STREAM will be converted to SDP

```
export SIMPLE_LIBSDP=1
```

- For more finite control use libsdp.conf

Usage – User Level configuration (cont.)

- Configure /etc/libdsp.conf
  - Substitute particular socket connections by SDP
  - Match vs match_both directives
  - Matching according to program name

    ```
    [match|match_both] program <regular expr.>
    ```

  - Matching according to IP address
    - on source

      ```
      [match|match_both] listen <tcp_port>
      ```

      Where tcp_port is

      ```
      <ip_addr>[/<prefix_length>][:<start_port>[-<end_port>]]
      ```

    - on destination

      ```
      match destination <tcp_port>
      ```

Usage – User Level configuration (cont.)

- Running ssh, scp over SDP
  - In libsdp.conf:

```
match_both listen *:22
```

  - On the server side

```
/etc/init.d/sshd stop
env LD_PRELOAD=/usr/lib64/libsdp.so
  LIBSDP_CONFIG_FILE=/u/etc/libsdp.conf /etc/init.d/sshd start
```

  - On the client side

```
LD_PRELOAD=/usr//lib64/libsdp.so
LIBSDP_CONFIG_FILE=/etc/libsdp.conf scp <file> <user>@<IPoIB
addr>:<dir>
```

Debug and monitoring

- Make sure ib_sdp module is loaded using:
  - **lsmod | grep sdp**

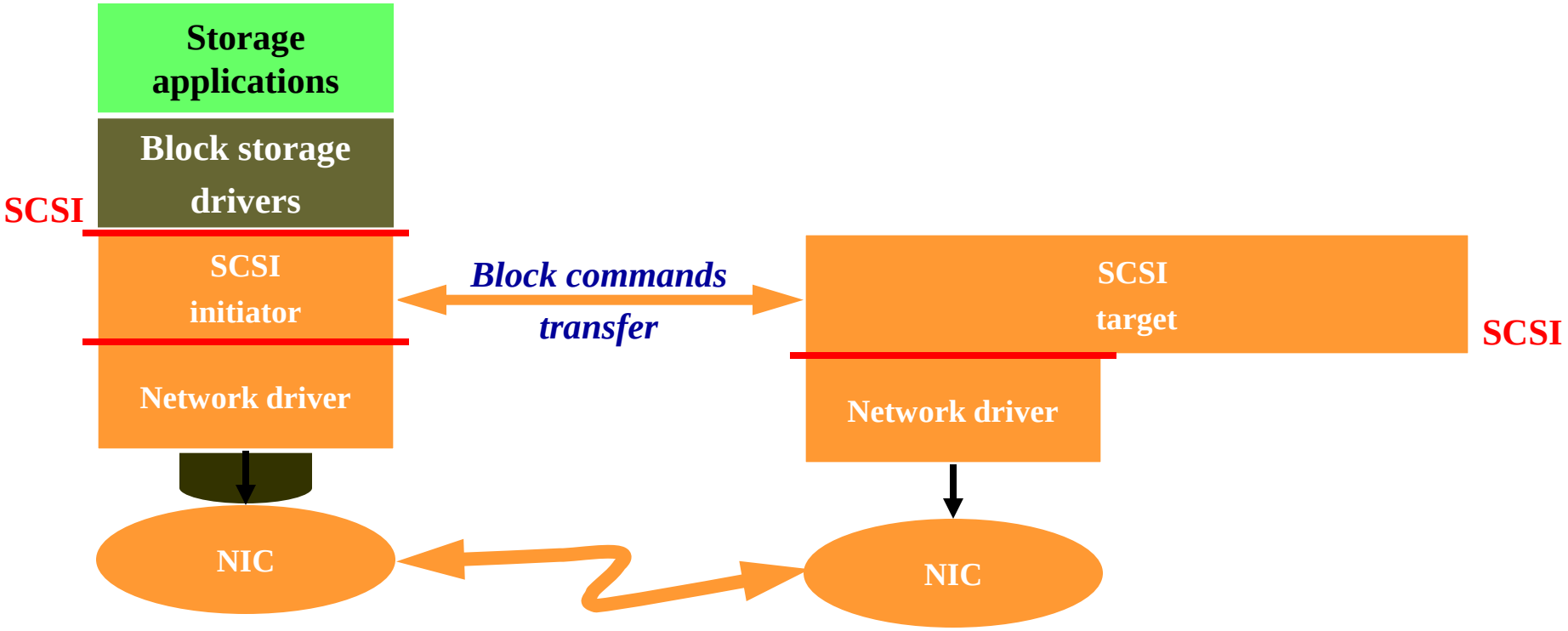
- To determine if a particular application is actually going over SDP use:
  - **sdpnetstat -S**

SRP – SCSI RDMA Protocol

SRP in a Nut Shell

- Maintain local disk access semantics
  - Plugs to the bottom of SCSI mid-layer
  - Delivers same functionality as Fiber Channel
  - Provides all hooks for storage network management
    - Requires in-network agents and SW
- Benefits – protocol offload
  - Enable RDMA optimized transfers
  - Protocol offload (SAR, retransmission, ack, etc)
- SRP defines the wire protocol

SCSI – from local to network storage



Loading SRP Initiator and Target Discovery

- Manual Load: modprobe ib_srp
  - Module parameter srp_sg_tablesize – max number of scatter/gather entries per I/O – default is 12
- Automatic Load: modify /etc/infiniband/openib.conf with SRP_LOAD=yes
- Discovering targets
  - ibsrpdm –c –d /dev/infiniband/umadXX
    - umad0: port 1 of first HCA in the system (mthca0 or mlx4_0)
    - umad1: port 2 of first HCA in the system
    - umad2: port 1 of second HCA in the system
    - …

Example-> ibsrpdm -c -d /dev/infiniband/umad3

id_ext=0002c9020023130c,ioc_guid=0002c9020023130c,dgid=fe800000000000002c9020023130d,pkey=ffff,service_id=0002c9020023130c

id_ext=0002c9020023193c,ioc_guid=0002c9020023193c,dgid=fe800000000000002c9020023193d,pkey=ffff,service_id=0002c9020023193c

id_ext=0002c9020023187c,ioc_guid=0002c9020023187c,dgid=fe800000000000002c9020023187d,pkey=ffff,service_id=0002c9020023187c

id_ext=0002c9020021d6f8,ioc_guid=0002c9020021d6f8,dgid=fe800000000000002c9020021d6f9,pkey=ffff,service_id=0002c9020021d6f8

SRP Tools - ibsrpdm

- Used to:
  - Detect targets on the fabric reachable by the Initiator
  - Output target attributes in a format suitable for use in the above “echo” command.
    - To detect all targets run: ibsrpdm
    - To generate output suitable for echo command run: ibsrpdm -c
      - » Sample output:

        id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
        dgid=fe8000000000000002c90200402bd5,pkey=ffff,
        service_id=200400a0b81146a1
    - Next you can copy paste this output into the “echo” command to establish the connection

SRP Tools - SRP Daemon

- srp_daemon is based on ibsrpdm and extends its functionalities.
  - Establish connection to target without manual issuing the *echo <target login info>* command
  - Continue running in the background, detecting new targets and establishing connections to targets (in daemon mode)
  - Enable High Availability operation (working together with Device-Mapper Multipath)
  - Have a configuration file (including/excluding targets to connect to)

SRP Tools - SRP Daemon

- srp_daemon commands equivalent to ibsrpdm
  - srp_daemon –a –o (same as *ibsrpdm*)
  - srp_daemon –c –a –o (same as *ibsrpdm –c*)
- srp_daemon extensions
  - To discover target from HCA name and port number: srp_daemon –c –a –o –i <mthca0> -p <port#>
  - To discover target and establish connections to them, just add the *-e* option and remove the *-a* option to the above commands
  - Configuration file /etc/srp_daemon.conf. Use –f option to provide a different configuration file. You can set values for optional parameters(ie. max_cmd_per_lun, max_sect…)

SRP Tools - SRP Daemon

- Run srp_daemon in *daemon* mode
  - run_srp_daemon –e –c –n –i <hca_name> -p <port#> → execute srp daemon as a daemon on specific port of a HCA. Please make sure to run only one instance of run_srp_daemon per port
  - srp_daemon.sh → execute run_srp_daemon on all ports of all HCAs in the system. You can look at srp_daemon log file in /var/log/srp_daemon.log
- Run srp_daemon automatically
  - Edit /etc/infiniband/openib.conf and turn on SRPHA_ENABLE=yes

Verifying SRP installation correctness

- “lsscsi” or “fdisk –l” will show the current scsi disk(s) in the system ie. /dev/sda
- Manual loading the SRP module and login to targets
- “lsscsi” or “fdisk –l” will show the new scsi disk(s) in the system ie. /dev/sdb, /dev/sdc,…
- Running some raw “dd”, xdd,… to new block devices ie. *dd if=/dev/sdb of=/dev/null bs=64k count=2000*
- Creating/mounting file-system
  - fdisk /dev/sdb (to create partitions)
  - mkfs –t ext3 /dev/sdb1
  - mount /dev/sdb1 /test_srp

SRP High Availability

- Using Device-Mapper (DM) multipath and srp_daemon
- There are several connections between an initiator host and target through different ports/HCAs of both host and target
- DM multipath is responsible for identifying paths to the same target and fail-over between paths
- When a path (say from port1) to a target fails, the ib_srp module starts an error recovery process.

SRP High Availability

- To turn on and run DM multipath automatically
  - For RHEL4, RHEL5:
    - Edit /etc/multipath.conf to comment out the devnode_blacklist (rhel4) or the blacklist (rhel5)
    - chkconfig multipathd on
  - For SLES10
    - chkconfig boot.multipathd on
    - Chkconfig multipathd on
- To manually run DM
  - modprobe dm-multipath
  - multipath –v 3 –l → list all luns with paths
  - multipath –m
- Access the srp luns/disks on /dev/mapper

Lab Exercise

Exercise #1 – Basic checks

1. Check that all nodes have MLX_OFED install
    1. Install latest MLX_OFED if missing
2. Which nodes do not have the driver up and running?
3. Are all cards in the cluster the same card type?
4. All port 1 links should be Active. Is this the case?
5. Verify that all HCAs are running latest firmware.

Exercise #2 – Update firmware

1. Upgrade firmware on all down rev nodes to latest FW.

Exercise #3 – Driver checks

1. Are all machines running OFED-1.4?

Exercise #4 – Subnet manager checks

1. Determine which nodes are running Master and any Standby Subnet managers.
2. Turn off Master SM.
3. Verify that a Standby SM has come on line.
4. Configure your designated node to load OSM automatically on boot-up.

Exercise #5 – Performance Tests

1. Run ib_send_bw between two nodes. What uni-directional bandwidth is achieved? What bi-directional bandwidth
2. Run ib_write_lat between two nodes. What latency is achieved?

Exercise #6 – MPI Tests

1. Run Pallas benchmark between two nodes, two processes per node.

Thank You

www.mellanox.com