In-Network Computing SHARP Technology for MPI Offloads

Devendar Bureddy

HPCAC-Stanford,  Feb 8, 2017
Accelerating HPC Applications

- Significantly reduce MPI collective runtime
- Increase CPU availability and efficiency
- Enable communication and computation overlap

Enabling Artificial Intelligence Solutions to Perform Critical and Timely Decision Making

- Accelerating distributed machine learning
SwitchIB-2 : Smart Switch

SHARP Enables Switch-IB 2 to Manage and Execute MPI Operations in the Network

Switch-IB 2 Enables the Switch Network to Operate as a Co-Processor

- The world fastest switch with <90 nanosecond latency
- 36-ports, 100Gb/s per port, 7.2Tb/s throughput, 7.02 Billion messages/sec
- Adaptive routing, congestion control
- Multiple topologies
### SHARP Performance Data – OSU Allreduce 1PPN, 128 nodes

<table>
<thead>
<tr>
<th>Message Size [B]</th>
<th>SHArP based</th>
<th>Host Based</th>
<th>SHArP improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2.76</td>
<td>5.82</td>
<td>2.11</td>
</tr>
<tr>
<td>16</td>
<td>2.76</td>
<td>5.91</td>
<td>2.14</td>
</tr>
<tr>
<td>32</td>
<td>2.86</td>
<td>6.04</td>
<td>2.11</td>
</tr>
<tr>
<td>64</td>
<td>3.01</td>
<td>6.76</td>
<td>2.25</td>
</tr>
<tr>
<td>128</td>
<td>3.24</td>
<td>7.37</td>
<td>2.27</td>
</tr>
<tr>
<td>256</td>
<td>3.50</td>
<td>8.99</td>
<td>2.57</td>
</tr>
<tr>
<td>512</td>
<td>4.06</td>
<td>11.11</td>
<td>2.74</td>
</tr>
<tr>
<td>1024</td>
<td>5.49</td>
<td>18.04</td>
<td>3.29</td>
</tr>
<tr>
<td>2048</td>
<td>8.44</td>
<td>33.61</td>
<td>3.98</td>
</tr>
<tr>
<td>4096</td>
<td>14.48</td>
<td>46.93</td>
<td>3.24</td>
</tr>
</tbody>
</table>
MiniFE is a Finite Element mini-application
- Implements kernels that represent implicit finite-element applications
Allreduce – Four Process Recursive Doubling

Step 1

Step 2
SHARP: Scalable Hierarchical Aggregation Protocol

- Reliable Scalable General Purpose Primitive
  - In-network Tree based aggregation mechanism
  - Large number of groups
  - Multiple simultaneous outstanding operations

- Applicable to Multiple Use-cases
  - HPC Applications using MPI / SHMEM
  - Distributed Deep Learning applications

- Scalable High Performance Collective Offload
  - Barrier, Reduce, All-Reduce, Broadcast
  - Sum, Min, Max, Min-loc, max-loc, OR, XOR, AND
  - Integer and Floating-Point, 32 / 64 bit
SHARP Tree

- **SHARP Operations are Executed by a SHARP Tree**
  - Multiple SHArP Trees are Supported
  - Each SHArP Tree can handle Multiple Outstanding SHArP Operations
  - Within a SHArP Tree, each Operation is Uniquely Identified by a SHArP-Tuple
    - GroupID
    - SequenceNumber

- **SHARP Tree is a Logical Construct**
  - Nodes in the SHArP Tree are IB Endnodes
  - Logical tree defined on top of the physical underlying fabric
  - SHArP Tree Links are implemented on top of the IB transport (Reliable Connection)
  - Expected to follow the physical topology for performance but not required
SHARP Tree (cont’d)

- **SHARP Tree is a Logical Construct**
  - Motivation to loosely follow the underlying Physical Topology for Performance
    - Efficient use of Physical Link Bandwidth
    - Shortest Overall Path Length from Leafs to Root

- **SHARP Group is a Tree Subset**
  - A sub-tree spanning a subset of the tree end-nodes

- **SHARP Tree Nodes are ‘Processes’**
  - Multiple SHARP Tree “Nodes” can reside on the same Physical HCA
SHARP SW Overview

### Release

<table>
<thead>
<tr>
<th>version</th>
<th>MOFED version</th>
<th>SwitchIB-2 FW</th>
<th>HPCX n</th>
<th>UFM version</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>MLNX OFED 3.3-x.x.x</td>
<td>15.1100.0072</td>
<td>1.6</td>
<td>-</td>
</tr>
<tr>
<td>v1.1</td>
<td>MLNX OFED 3.4-0.1.2.0</td>
<td>15.1200.0102</td>
<td>1.7</td>
<td>-</td>
</tr>
<tr>
<td>v1.2</td>
<td>MLNX OFED 4.0-x.x.x</td>
<td>15.1200.0102</td>
<td>1.8</td>
<td>5.8</td>
</tr>
</tbody>
</table>

### Prerequisites

- SHARP software modules are delivered as part of HPC-X
- Switch-IB 2 firmware - 15.1200.0102 or later
- MLXN OS - 3.6.1002 or later
- MLNX OFED 4.0
- MLNX OpenSM 4.7.0 or later (available with MLNX OFED 3.3-x.x.x or UFM 5.6)
Mellanox HPC-X™ Scalable HPC Software Toolkit

- Complete MPI, PGAS OpenSHMEM and UPC package
- Maximize application performance
- For commercial and open source applications
- Best out of the box experience
Mellanox HPC-X - Package Contents

- Allow fast and simple deployment of HPC libraries
  - Both Stable & Latest Beta are bundled
  - All libraries are pre-compiled
  - Includes scripts/module files to ease deployment

- Package Includes
  - OpenMPI / OpenSHMEM
  - BUPC (Berkeley UPC)
  - UCX
  - MXM
  - FCA-2.5
  - FCA-3.x (HCOLL)
  - SHARP
  - KNEM
    - Allows fast intra-node MPI communication for large messages
  - Profiling Tools
    - Libibprof
    - IPM
  - Standard Benchmarks
    - OSU
    - IMB
**HPCX/SHARP SW architecture**

- **HCOLL**
  - optimized collective library
- **Libsharp_coll.so**
  - Implementation of high level sharp API for enabling sharp collectives for MPI
  - uses low level libsharp.so API
- **Libsharp.so**
  - Implementation of low level sharp API
- **High level API**
  - Easy to use
  - Easy to integrate with multiple MPIs (OpenMPI, MPICH, MVAPICH*)

![Diagram of the HPCX/SHARP SW architecture]

**Diagram:**
- **MPI (OpenMPI)**
- **HCOLL(libhcoll)**
- **SHARP(libsharp/libsharp_coll)**
- **InfiniBand Network**
SHARP SW Architecture

Computes node

MPI Process

SHARPD daemon

Aggregation Node

Aggregation Node

Aggregation Node

Subnet Manager

Aggregation Manager (AM)

SHARPD daemon
SHARP SW Components

- **SHARP SW components:**
  - **Libs**
    - libsharp.so (low level api)
    - libsharp_coll.so (high level api)
  - **Daemons (sysadmin)**
    - sharpd, sharp_am
  - **Scripts**
    - sharp_benchmark.sh
    - sharp_daemons_setup.sh
  - **Utilities**
    - sharp_coll_dump_config
    - sharp_hello
    - sharp_mpi_test
  - **public API**
    - sharp_coll.h
SHARP: SHARP Daemons

- **sharp_am**: Aggregation Manager daemon
  - same node as Subnet Manager
  - Resource manager

- **sharpd**: SHARP daemon
  - compute nodes
  - Light wait process
  - Almost 0% cpu usage
  - Only control path
SHARP: Configuring Subnet Manager

- Edit the opensm.conf file.
- Set the parameter “sharp_enabled” to “2”.
- Run OpenSM with the configuration file.
  - % opensm -F <opensm configuration file> -B

- Verify that the Aggregation Nodes were activated by the OpenSM, run "ibnetdiscover".
  
  For example:
  vendid=0x0
  devid=0xcf09
  sysimgguid=0x7cfe900300a5a2a0
  caguid=0x7cfe900300a5a2a8
  Ca 1 "H-7cfe900300a5a2a8" # "Mellanox Technologies Aggregation Node"
  [1](7cfe900300a5a2a8) "S-7cfe900300a5a2a0"[37] # lid 256 lmc 0 "MF0;sharp2:MSB7800/U1" lid 512 4xFDR
SHARP: Configuring Aggregation Manager

- Create a configuration directory for the future SHArP configuration file.
  - `% mkdir $HPCX_SHARP_DIR/conf`

- Create the fabric.lst file.
  - Copy the subnet LST file created by the Subnet Manager to the AM’s configuration files directory
  - Rename it to fabric.lst:
    - `% cp /var/log/opensm-subnet.lst $HPCX_SHARP_DIR/conf/fabric.lst`

- Create root GUIDs file.
  - Copy the root_guids.conf file if used for configuration of Subnet Manager to $HPCX_SHARP_DIR/conf/root_guid.cfg (or)
  - Identify the root switches of the fabric and create a file with the node GUIDs of the root switches of the fabric.
  - For example: if there are two root switches files contains
    0x0002c90000000001
    0x0002c90000000008

- Create sharp_am.conf file
  `% cat > $HPCX_SHARP_DIR/conf/sharp_am.conf << EOF
  fabric_lst_file $HPCX_SHARP_DIR/conf/fabric.lst
  root_guids_file $HPCX_SHARP_DIR/conf/root_guid.cfg
  ib_port_guid <PortGUID of the relevant HCA port or 0x0> EOF`
SHARP: Running SHARP Daemons

### Setup the daemons
- `$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh`

### Usage
- Usage: `sharp_daemons_setup.sh <s> <r> [-p SHArP location dir] <d daemon> <m>`
  - `-s` - Setup SHArP daemon
  - `-r` - Remove SHArP daemon
  - `-p` - Path to alternative SHArP location dir
  - `-d` - Daemon name (sharpd or sharp_am)
  - `-m` - Add monit capability for daemon control

- `$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharp_am`
- `$service sharp_am start`
SHARP: Running SHARP Daemons

- **sharp_am**
  - `%%$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharp_am`
  - `%%service sharp_am start`
  - Log: `/var/log/sharp_am.log`

- **Sharpd**
  - conf file: `$HPCX_SHARP_DIR/conf/sharpd.conf`
    - `ib_dev <relevant_hca:port>`
    - `sharpd_log_level 2`
  - `%pdsh -w <hostlist> $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharpd`
  - `%pdsh -w jupiter[001-032] service sharpd start`
  - Log: `/var/log/sharpd.log`
SHARP resource (quota) handling

- **Job Quota**
  - Max groups
    - disable sharp on communicator if it fails to create sharp group
  - Max osts
    - 1 for blocking collectives
    - > 1 for non-blocking & fragmentation
  - Data size per OST
    - Max sharp supported payload (256 B)
  - Max QPs
    - Need at least #PPN QPs/node

- **Hard problem to efficiently allocate resources**
  - Application profiling might help
  - Per group
    - Max OSTs
    - Data size per OST
MPI Collective offloads using SHARP

- Enabled through FCA-3.x (HCOLL)
- Flags
  - HCOLL_ENABLE_SHARP (default: 0)
    0 - Don't use SHArP
    1 - probe SHArP availability and use it
    2 - Force to use SHArP
    3 - Force to use SHArP for all MPI communicators
    4 - Force to use SHArP for all MPI communicators and for all supported collectives
  - HCOLL_SHARP_NP (default: 2)
    Number of nodes (node leaders) threshold in communicator to create SHArP group and use SHArP collectives
  - SHARP_COLL_LOG_LEVEL
    0 – fatal , 1 – error, 2 – warn, 3 – info, 4 – debug, 5 – trace
  - HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX
    Maximum allreduce size run through SHArP
MPI Collectives offloads using SHARP

- Resources (quota)
  - `SHARP_COLL_JOB_QUOTA_MAX_GROUPS`
    - #communicators
  - `SHARP_COLL_JOB_QUOTA_OSTS`
    - Parallelism on communicator
  - `SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST`
    - Payload/OST

- For complete list of SHARP COLL tuning options
  - `$HPCX_SHARP_DIR/bin/sharp_coll_dump_config -f`
MPI Collectives offloads using SHARP

$ mpirun --display-map --bind-to core --map-by node -np 32 -mca pml yalla -mca btl_openib_warn_default_gid_prefix 0 -mca rmaps_dist_device mlx5_0:1 -mca rmaps_base_mapping_policy dist:span -x MXM_RDMA_PORTS=mlx5_0:1 -x HCOLL_MAIN_IB=mlx5_0:1 -x HCOLL_ENABLE_SHARP=1 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=4096 -x SHARP_COLL_LOG_LEVEL=3 -x HCOLL_ENABLE_MCAST_ALL=0 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=128 /home/user02/hpcx-v1.7.405-gcc-MLNX_OFED_LINUX-3.4-1.0.0.0-redhat6.5-x86_64/ompi-v1.10/tests/osu-micro-benchmarks-5.2/osu_allreduce -i 10000 -x 1000 -m 256

# OSU MPI Allreduce Latency Test v5.2
# Size  Avg Latency(us)
4       2.01
8       2.58
16      2.04
32      2.59
64      2.10
128     2.79
256     2.69
SHARP API
SHARP API

- Datatypes
  - 32 bit – INT/UINT/FLOAT
  - 64 bit – LONG/Unsinged LONG/DOUBLE

- Reduce Ops
  - MIN
  - MAX
  - SUM
  - LAND
  - LOR
  - BOR
  - LXOR
  - BXOR
  - MAXLOC
  - MINLOC
  - PROD (Not supported)
SHARP API

- **Control Path**
  - sharp Init / finalize
  - sharp comm(group) create/destroy

- **Data path**
  - Collectives operations
    - Allreduce
    - Barrier
    - Bast (Not supported yet)
    - Reduce (Not supported yet)
  - Blocking & Non-Blocking
SHARP API

- Control path
- Data path
- Request pool
- mpool
- libsharp_coll.so
- utils(timer/env/data structures..)
- Low level SHARP API
- libsharp.so
SHARP API – Control path

- Sharp job initialize
  - API
    ```c
    int sharp_coll_init(struct sharp_coll_init_spec *sharp_coll_spec,
                        struct sharp_coll_context **sharp_coll_context)
    ```

- Code flow in HPC-X

- Input spec

```
struct sharp_coll_init_spec {
    int job_id;          /* MPI Job ID */
    char *hostlist;      /* optional in job scheduler case */
    int world_rank;
    int world_size;
    int (*progress_func)(void); /* External progress func */
    struct sharp_coll_config config; /* SHARP COLL Configuration. See 'sharp_coll_default_config' */
    struct sharp_coll_out_of_band_colls oob_colls;
};
```
High level API – Control path..

- Sharp job finalize
  - api
    ```c
    void sharp_coll_finalize(struct sharp_coll_context *context);
    ```

- Code flow in HPC-X

```plaintext
MPI_Finalize() → hcoll_finalize() → sharp_coll_finalize()
```
High level API – Control path...

- Sharp comm(group) create
  - Sharp comm is created for hcoll ptp subgroup
  - api
    ```c
    int sharp_coll_comm_init(struct sharp_coll_context *context,
                            struct sharp_coll_comm_init_spec *spec,
                            struct sharp_coll_comm **sharp_coll_comm);
    ```

  - code flow in HPCX

  - Input spec
    ```c
    struct sharp_coll_comm_init_spec {
      int rank;       /* MPI rank */
      int size;       /* Communicator size */
      int is_comm_world; /* is MPI_COMM_WORLD */
      void *oob_ctx;  /* External context for OOB functions */
    };
    ```
High level API – Control path...

- **Sharp comm(group) create**
  - Sharp comm is created for hcoll ptp subgroup
  - api
    ```c
    void sharp_coll_comm_destroy(struct sharp_coll_comm *comm);
    ```
  - code flow in HPC-X

```
struct sharp_coll_comm_init_spec {
    int rank;   /* MPI rank */
    int size;   /* Communicator size */
    int is_comm_world; /* is MPI_COMM_WORLD */
    void *oob_ctx; /* External context for OOB functions */
};
```
High level API – data path

- **Barrier**
  - `int sharp_coll_do_barrier(struct sharp_coll_comm *comm);`

- **Allreduce**
  - `int sharp_coll_do_allreduce(struct sharp_coll_comm *comm, struct sharp_coll_reduce_spec *spec);`

- Reduce spec:
  ```
  struct sharp_coll_reduce_spec {
    int root; /* root rank number (ignored for allreduce) */
    struct sharp_coll_data_desc sbuf_desc; /* data to submit */
    struct sharp_coll_data_desc rbuf_desc; /* buffer to receive the result */
    enum sharp_datatype dtype; /* data type */
    int length; /* reduce operation size */
    enum sharp_reduce_op op; /* operator */
  };
  ```

- Input Buffer types
  - Contig Buffer
  - Stream (not supported yet)
  - IOV (not supported yet)
High level API – data path..

- **Non-blocking API**
  - Returns request handle
  - use can test/wait for the completion
  - API
    - `int sharp_coll_do_barrier_nb(struct sharp_coll_comm *comm, struct sharp_coll_request **req);`
    - `int sharp_coll_do_allreduce_nb(., ., struct sharp_coll_request **req);`
    - `int sharp_coll_req_test(struct sharp_coll_request *req);`
    - `int sharp_coll_req_wait(struct sharp_coll_request *req);`
    - `int sharp_coll_req_free(struct sharp_coll_request *req);`
Summary

- Scalable In-network computing
- Easy to use
- Easy to integrate API