NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP) Rev 3.5.0

Table of Contents

1 Overview

NVIDIA® Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)™ technology improves the performance of MPI and Machine Learning collective operation, by offloading collective operations from CPUs and GPUs to the network and eliminating the need to send data multiple times between endpoints.

This innovative approach decreases the amount of data traversing the network as aggregation nodes are reached, and dramatically reduces collective operations time. Implementing collective offloads communication algorithms supporting streaming for Machine Learning in the network also has additional benefits, such as freeing up valuable CPU and GPU resources for computation rather than using them to process communication.

With the 3rd generation of SHARP, multiple aggregation trees can be built over the same topology, enabling the aggregation and reductions benefits (also known as In-Network Computing) to many parallel jobs over the same infrastructure.

Further information on this product can be found in the following NVIDIA SHARP documents:

- Release Notes
- Deployment Guide

1.1 Software Download

Please visit https://developer.nvidia.com/networking/hpc-x

1.2 Document Revision History

For the list of changes made to this document, refer to Revision History.

2 Release Notes

Revision	Date	Description
3.5.0	Nov 7, 2023	Initial release of this document version.

The release note pages provide the following information on NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP).

2.1 General Information

2.1.1 Packages Provided with SHARP

SHARP Software is provided with the following package:

Package	Version
MLNX_OFED	23.07-0.5.1.2
HPC-X	2.17
UFM (Aggregation Manager only)	6.15.0

2.1.2 Prerequisites

NVIDIA SHARP requires either NVIDIA Unified Fabric Manager (UFM®), or a dedicated server running Subnet Manager. In the latter case, the onboard Subnet Manager should be disabled in managed switches.

Name	Firmware/Software Version
NVIDIA Quantum	27.2012.1010
NVIDIA Quantum-2	31.2012.1024
ConnectX-6	20.38.1900
ConnectX-6 DE	22.38.1900
ConnectX-7	28.38.1900
MLNX-OS	3.11.1010
Subnet Manager	OpenSM 5.13

2.1.2.1 Supported Operating Systems and Platforms

For complete list of supported Operating Systems and platforms, please refer to list of operation systems and platforms supported by HPC-X and MLNX_OFED.

2.2 Changes and New Features

2.2.1 Changes and New Features

Feature/Change	Description
Log Verbosity Control per Category	Added support for controlling the `sharp_am log` messages verbosity level per the desired log category.
Bug Fixes	See Bug Fixes section.

2.2.2 Parameter Changes

Parameter	Component	Description
`log_categories_file`	`Sharp_am`	Added support for a new string parameter which enables indicating the log categories file path. The value "(NULL)" indicates that the log categories file does not exist. **Default**: In UFM, the default path is: `/opt/ufm/files/conf/fabric_log_categories.cfg`

2.3 Bug Fixes in this Version

Internal Ref.	Issue
3609384	**Description**: Fixed issues concerning Sharp_AM connection creation with rank zero clients of active jobs during a restart when UCX is enabled.
	Keywords: `sharp_am`, libsharp, restart
	Discovered in Version: 3.4.0
	Fixed in Release: 3.5.0
3541153	**Description**: Fixed an issue where client application is abnormally terminated before the `sharp_coll_finalize` method, `sharp_am` is supposed to automatically detect and clean the job resources. However, with UCX, only one such termination is detected per cycle, leading to incomplete job cleaning. Similarly, when using NCCL and hosts with multiple GPUs/HCAs, each HCA gets its own SHARP job, which results in `sharp_am` taking several cycles to detect all the jobs that require cleaning. As a consequence, hosts operating in the previous application cannot initiate a new SHARP job until `sharp_am` detects and cleans all the necessary jobs.
	Keywords: `sharp_am`, NCCL, UCX

Internal Ref.	Issue
	Discovered in Version: 3.4.0
	Fixed in Release: 3.5.0

2.4 Known Issues

Internal Reference Number	Issues
3478803	**Description:** Getting topology info (`sharp_cmd topology`), fails when executed from the mgmt host.
	Workaround: It is possible to run it from different hosts, or add the following environment variable: `SHARP_ALLOW_SM_PORT=1`
	Keywords: SHARP topology API
	Discovered in Version: 3.5.0
3340353	**Description:** When reconfiguring a standby management host to operate as a compute host, it will not be able to run SHARP jobs unless sharp_am is restarted. In case that a host runs the SM process, it will automatically be detected by the master SM as a standby SM and be reported as a standby management host. Note that restart is not required if ignore_sm_guids is set to FALSE.
	Workaround: N/A
	Keywords: Slave; compute host; ignore_sm_guids
	Discovered in Version: 3.3.0
3371820	**Description:** Congestion Control cannot be configured on the same SLs used by sharp_am.
	Workaround: N/A
	Keywords: Congestion control; SL
	Discovered in Version: 3.3.0
3438393	**Description:** When operating in the following configuration mode, resource limitation is ignored and no limit is set to any application: Dynamic trees allocation is used; Quasi Fat Tree (QFT)-oriented logic is used; and reservation_mode is on.
	Workaround: N/A
	Keywords: Dynamic trees allocation; QFT; resource limitation
	Discovered in Version: 3.3.0
3305335	**Description:** When running mpirun with multiple groups, the following error message might be received: `[error] - AM QPAlloc confirm QP MAD response status 0x1c00` This message is received due to to the fact that multiple unserialized MAD requests are run in parallel.

Internal Reference Number	Issues
	Workaround: Set the SHARP_COLL_SERIALIZE_MADS environment variable to TRUE when running mpirun.
	Keywords: mpirun; SHARP_COLL_SERIALIZE_MADS
	Discovered in Version: 3.2.0
3225401	**Description:** Dynamic trees creation feature does not support a case in which all root switches are down and restarted. If such a scenario takes place, sharp_am should be restarted once the root switches are up and running.
	Workaround: N/A
	Keywords: Aggregation Manager; sharp_am; dynamic trees
	Discovered in Version: 3.1.0
3237831	**Description:** SHARP does not support reassignment of LID values. In case LID reassignment is desired, make sure to stop all SHARP jobs, reassign LIDs via OpenSM, and restart sharp_am once the reassignment is done.
	Workaround: N/A
	Keywords: Aggregation Manager; OpenSM
	Discovered in Version: 3.1.0
3048427	**Description:** In the case that a switch split mode is modified (off/on), sharp_am does not handle the new number of supported ports unless it is restarted.
	Workaround: Restart sharp_am after changing a switch split mode definition.
	Keywords: Aggregation Manager; split mode
	Discovered in Release: 2.7.0
3051699	**Description:** Changing the configuration of SHARP switch ports using device_configuration_file does not take effect on disconnected split ports. If these ports are connected later, they will remain with their default configuration.
	Workaround: If the new configuration is desired for the split ports, make sure to restart the Aggregation Manager after connecting a split port to a host.
	Keywords: Aggregation Manager; split port
	Discovered in Release: 2.7.0
3051924	**Description:** Adding or replacing non-leaf switches is currently not supported by Aggregation Manager for Dragonfly+ topologies.
	Workaround: Restart Aggregation Manager after the Subnet Manager completes fabric reconfiguration followed by the fabric changes.
	Keywords: Fabric extension; Aggregation Manager; AM
	Discovered in Release: 2.7.0
-	**Description**: On multi PKEY environment, UCX in SHARP can use only the default PKEY (PKEY at index 0).
	Workaround: Use sockets for communication over non-default PKEY.
	Keywords: Configuration, SMX, UCX, PKEY

Internal Reference Number	Issues
	Discovered in Release: 2.4.3
1307124	**Description**: Begin Job requests with virtual ports might be rejected until fabric virtualization info file is parsed.
	Workaround: Wait for AM to discover virtual ports before sending Begin Job requests.
	Keywords: Aggregation Manager, Socket Direct, Virtual Ports
	Discovered in Release: 1.5.3
1193629	**Description**: Configuring sharp_am as daemon is not possible when installing from RPM into non-default location.
	Workaround: Configure daemon manually.
	Keywords: Configuration
	Discovered in Release: 1.5.3
1307108	**Description**: Discovering a new Aggregation Node (AN) found on the shortest path between two ANs might invalidate the existing path.
	Workaround: Restart Aggregation Manager after the Subnet Manager completes fabric reconfiguration followed by the fabric changes.
	Keywords: Aggregation Manager, Aggregation Node
	Discovered in Release: 1.5.3
-	**Description:** Aggregation Manager High Availability is currently not supported in HPCX/MLNX OFED packages. Therefore, only a single instance of Aggregation Manager can run in the IB fabric.
	Workaround: Use Aggregation Manager in UFM.
	Keywords: Aggregation Manager
-	**Description:** Aggregation manager should run on the same Host where the Master Subnet Manager (SM) is running.
	Workaround: N/A
	Keywords: Aggregation Manager
-	**Description:** In case of HPCX/MLNX OFED packages, upon Subnet Manager handover/failover, another instance of Aggregation Manager should be started on the Host where the new Master SM is running
	Workaround: Use Aggregation Manager in UFM.
	Keywords: Aggregation Manager
-	**Description:** Aggregation Manager should be started after completion of fabric configuration by the Subnet Manager.
	Workaround: N/A
	Keywords: Aggregation Manager
-	**Description:** Only Fat-Tree, Quasi-Fat-Tree, Hypercube and Dragonfly+ topologies are supported by the Aggregation Manager.
	Workaround: N/A

Internal Reference Number	Issues
	Keywords: Fabric Topology
-	**Description:** Only IB fabrics where all compute nodes are connected to NVIDIA SHARP capable switches are supported by the Aggregation Manager.
	Workaround: Manually configure mapping between the compute port and the Aggregation Node.
	Keywords: Fabric Topology
-	**Description:** Upon changes in configuration file beyond parameters in 3.3, Aggregation Manager should be restarted to deploy new configuration.
	Workaround: N/A
	Keywords: Configuration

3 Introduction

NVIDIA® Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)™ technology improves the performance of MPI and Machine Learning collective operation, by offloading collective operations from CPUs and GPUs to the network and eliminating the need to send data multiple times between endpoints.

This innovative approach decreases the amount of data traversing the network as aggregation nodes are reached, and dramatically reduces collective operations time. Implementing collective offloads communication algorithms supporting streaming for Machine Learning in the network also has additional benefits, such as freeing up valuable CPU and GPU resources for computation rather than using them to process communication.

With the 3rd generation of SHARP, multiple aggregation trees can be built over the same topology, enabling the aggregation and reductions benefits (also known as In-Network Computing) to many parallel jobs over the same infrastructure.

4 Setting up NVIDIA SHARP Environment

NVIDIA SHARP binary distribution is available as part of HPC-X, MLNX_OFED and UFM packages (among SHARP binaries, UFM includes Aggregation Manager (AM) only).

4.1 Setup Requirements

Prior to installing and using NVIDIA SHARP, make sure the following requirements are met.

- Run Aggregation Manager using a "root user" as trusted entities.
- Make sure onboard Subnet Manager is disabled in the managed switches. (Aggregation Manager is a central entity running on a dedicated server with a master Subnet Manager. This dedicated server cannot serve as a compute node.
- Configure TCP/IP before running NVIDIA SHARP and Aggregation Manager communicate over TCP/IP.
- Run NVIDIA Switch-IB 2/NVIDIA Quantum/NVIDIA Quantum-2 switches with the supported firmware versions as specified in the Prerequisites section in the Release Notes (use ibdiagnet utility to check the installed firmware version on the switches).
- Enabled IPoIB interface in compute servers in order to enable using UD multicast for result distribution in SHARP.
- Make sure SHARP Aggregation Manager out-of-the-box subnets are configured with SM using the following routing engines:
    - Tree based topologies: updn, ar_updn, ftree, ar_ftree
    - DragonFly+ topology: dfp
    - Hypercube topologies: dor routing engine with dor_hyper_cube_mode enabled

4.2 Using NVIDIA SHARP from HPC-X

When using HPC-X package, please refer to HPC-X User Manual for installation and configuration procedures.
This deployment guide includes examples on the environment variables `HPCX_SHARP_DIR` and `OMPI_HOME`, and assumes that HPC-X installation is in a shared folder accessible from all compute nodes.
To download the HPC-X packages, go here.

4.3 Using NVIDIA SHARP from MLNX_OFED

When using MLNX_OFED distribution, the HPCX_SHARP_DIR environment variable has to be set to redirect to SHARP installation directory (default location: /opt/mellanox/sharp), and OMPI_HOME environment variable to the MPI installation directory.

To download MLNX_OFED packages, go here.

4.4 Using NVIDIA SHARP Aggregation Manager from UFM

When using Aggregation Manager from UFM, NVIDIA SHARP support has to be enabled in UFM. For further information, refer to the UFM User Manual.

UFM package includes only SHARP Aggregation Manager. Other NVIDIA SHARP components are not available through UFM and should be installed from either HPC-X or MLNX_OFED packages.

4.5 NVIDIA Hardware Capabilities and Limitations

Device	Capabilities and limitations
NVIDIA Quantum	• Supports both SHARP low latency and streaming aggregation operations • Supports up to 126 aggregation trees in the subnet (63 low latency trees, and 63 streaming aggregation trees) **Note:** The number of SHARP streaming aggregation operations is limited to one active tree per switch
NVIDIA Quantum-2	• Supports both SHARP low latency and streaming aggregation operations • Supports up to 1023 aggregation trees in the subnet (511 low latency trees, and 511 streaming aggregation trees) **Note:** Multiple SHARP streaming aggregation operations can be operated in parallel by a single Quantum-2 switch. The limit is one active tree per port
ConnectX-5	Supports SHARP low latency operation only
ConnectX-6 and above	Supports both SHARP low latency and streaming aggregation operations

5 Running NVIDIA SHARP Aggregation Manager (AM) Daemons

> As of NVIDIA SHARP version 2.7.0, sharpd daemon no longer exists. sharpd-related activity is now performed from the user-application process instead.

This section describes how to install Aggregation Manager in the fabric using NVIDIA SHARP AM daemon script.

NVIDIA SHARP Aggregation Manager daemon (`sharp_am`) is executed on a dedicated server along with the Subnet Manager.

Installing Aggregation Manager as a service is required when used from the HPC-X or from MLNX_OFED packages.

5.1 NVIDIA SHARP Daemons Installation Script

In order to install/remove NVIDIA SHARP AM daemons, use *sharp_daemons_setup.sh* script provided with the NVIDIA SHARP package. For example:

```
$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh

Usage: sharp_daemons_setup.sh (-s | -r) [-p SHARP location dir] -d
<sharpd | sharp_am> [-m]
        -s - Setup SHARP daemon
        -r - Remove SHARP daemon
        -p - Path to alternative SHARP location dir
        -d - Daemon name (sharp_am)
        -b - Enable socket based activation of the service
```

5.2 Registering sharp_am as a Service on the Subnet Manager Node

1. Run the following as root:

   ```
   # $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s -d sharp_am
   ```

   Daemon's log location is: /var/log/sharp_am.log

2. Set the "run level".
3. Start sharp_am as root.

   ```
   # service sharp_am start
   ```

5.3 Removing Daemons

To remove sharp_am, run the following on the AM host:

```
# $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -r -d sharp_am
```

5.4 Upgrading NVIDIA SHARP AM Daemons

Upgrading SHARP AM daemons requires their removal and re-registration as instructed in the sections above.

6 Modifying NVIDIA SHARP Aggregation Manager Configuration

SHARP Aggregation Manager (sharp_am) has factory default configuration that can be modified either by command line parameters or through a configuration file.

sharp_am is operated either from UFM or HPC-X.

- In the case of UFM, sharp_am is provided with UFM default config file. For information on how to operate SHARP from UFM, please refer to "NVIDIA SHARP Integration" Appendix in the latest UFM User Manual available here.
- In the case of HPC-X, please follow the instructions below.

6.1 NVIDIA SHARP Integration with HPC-X

SHARP Aggregation Manager (sharp_am) uses a configuration file from the default location */etc/sharp/sharp_am.cfg*.

If no such file exists, sharp_am will use the factory defaults.

sharp_am can also be executed using the parameter -O that provides the location of the config file:

```
$ $HPCX_SHARP_DIR/bin/sharp_am -O <desired config file path>
```

If the file does not exist, it can be created using the following command:

```
$ $HPCX_SHARP_DIR/bin/sharp_am -c /etc/sharp/sharp_am.cfg
```

The above command creates a config file with the factory default settings. Make sure the directory exists before running the command.

In order to modify the configuration settings, edit the file and change the parameter values accordingly. Some parameters require a restart of sharp_am in order to take effect, while others only require only notifying sharp_am that a change in the config file has taken place.

In the config file, every parameter has the following comment:

```
# Parameter supports update during runtime: yes/no
```

If one of the modified parameters does not support update during runtime, then sharp_am restart is required. If not, it is sufficient to signal sharp_am with sighup (kill -1 <pid>).

7 sharp_am Network Interfaces

sharp_am communicates with the following entities:

- IB switches - sharp_am sends MADs to get status and configure the switches for SHARP activities.
  The MADs communication with IB switches takes place over the IB network.
- libsharp - Rank0 of collective operation, sending SHARP job requests to sharp_am and receiving sharp_am instructions.
  The communication with libsharp is performed via a proprietary binary protocol called smx. The transport layer of the smx can be via IB using UCX (InfiniBand transport), or via sockets (Ethernet).
- UFM - when operating inside UFM, various information and configuration commands are passed from UFM to sharp_am.
  The communication with UFM is also performed via the smx proprietary protocol. However, the transport layer of this communication is unix-socket.

7.1 Network Interfaces Configuration

By default, sharp_am uses the opensm IB interface for the MADs and libsharp communication.

The communication with libsharp is done via socket (Ethernet) transport by default.

A unix-socket is kept open by default for communication with UFM.

It is possible to specify certain interfaces and to change the communication protocol, using the following configuration parameters:

Parameter	Component	Description
ib_port_guid	sharp_am	Sets the GUID of the port to which sharp_am binds to, for all MAD communication with the switches. Value of 0 means to use the same port that is used by OpenSM. Default value: 0
smx_enabled_protocols	sharp_am	A bitmask specifying which transport layers should be enabled for smx communication. It is possible to provide multiple options. Bit 1 (value 1) - UCX Bit 2 (Value 2) - Sockets. Bit 3 (value 4) - Unix sockets (needed for UFM). Default value: 6, which means Sockets & Unix sockets.
smx_protocol	sharp_am	Defines the default protocol that will be used when communicating with libsharp. Value 1 - UCX. Value 2 - Sockets. Default value: 2, which means sockets.

Parameter	Component	Description
smx_sock_interface	sharp_am	Relevant only in case that smx socket transport is enabled. Sets the interface to be used by smx for the sockets connections. The interface should be mentioned by its name. Empty value means to use the same interface used by OpenSM, using IP-over-IB in this case. When sharp_am is operating inside UFM, this parameter is automatically set by UFM according to its internal logic and the am_interface parameter in the gv.cfg file. Default value: Empty.
smx_sock_port	sharp_am	Relevant only in case that smx socket transport is enabled. IP port number to be used for the socket listener. Default value: 6126
smx_ucx_interface	sharp_am	Relevant only in case that smx UCX transport is enabled. Sets the interface to be used by smx for the UCX connections. The interface should be mentioned by its name. Empty value means to use the same interface used by OpenSM. Default value: Empty.

7.2 Management Host Network Interfaces High Availability

In case the management host has multiple network interfaces, sharp_am can operate in HA mode, automatically handling network interface failures and switching to an active interface without interrupting any activity.

HA support for the IB transport is handled by sharp_am itself, while HA for Ethernet transport is handled by ip-bonding.

> In the event of network failure while a new job is being established, the operation will fail. However, upcoming job requests will not be affected, and on-going jobs will continue to operate as usual.

7.2.1 HA Configuration

1. ib_port_guid should be set to 0 (as its default), indicating that sharp_am should choose which port to use and which not to use.
2. allow_remote_sm - should be set to False (as its default). HA of the IB ports can operate only when sharp_am resides on the same machines with OpenSM.
3. In case smx ucx is enabled, smx_ucx_interface should be empty (as its default), indicating that sharp_am should choose which interface to use and which not to use.

4. In case that smx socket is enabled, ip-bonding should be configured on the management host and smx_sock_interface should be set to the bond interface.

7.3 UFM Appliance Firewall Settings

UFM Appliance Gen 3.x uses firewall that is configured to block the TCP port used by sharp_am by default, preventing SHARP clients from communicating with sharp_am. However, if you need to use UFM Appliance Gen 3.x with SHARP, you can resolve this by opening the required TCP port by running `ufw allow 6126/tcp.` Make sure that the port you specify in the 'smx_sock_port' config parameter matches the one you allow through the firewall.

8 sharp_am Log and Dump Files

The `sharp_am` logs its active logs to a log file named `sharp_am.log`.

sharp_am also generates various dump files, useful for monitoring and used by `sharp_am` at restart, to retain the previous run state.

Since some of the dump files are used by `sharp_am` at restart, it is important not to modify their content.

8.1 Activity Log Verbosity Level

Configuration parameters allow control over the verbosity level of the activity log file. The logged messages are categorized by relevancy, such as those related to network activity or SHARP trees calculations. Each category can have a distinct verbosity level. The two configuration parameters that control the log verbosity are `log_verbosity` and `log_categories_file`.

The `log_verbosity` config parameter functions as the main log verbosity parameter. The value set in this parameter defines the desired log verbosity for all categories unless specified differently for a particular category.

The `log_categories_file` parameter specifies the full path to a configuration file that defines the desired log verbosity for each category. By default in UFM, the provided file does not set specific levels to any category; all categories are commented out.

If a category is defined in the `fabric_log_catgeories.cfg` file, its definition overrides the main log verbosity. `Sharp_am log` verbosity can be updated without restarting by sending a SIGHUP signal. When updating the `sharp_am` configuration, you can modify the main log verbosity, update the location of the categories file, and adjust the content of the categories file.

8.2 Log Levels

There are five log levels, and their configuration is determined by numerical representation, however, the log messages are displayed with their full names.

The log levels include:

1. Error
2. Warning
3. Info
4. Debug
5. Trace/Verbose

When operating under normal conditions, it is advisable to set the log level to 3-Info.

8.3 Log categories config file

The configuration file included in the SHARP package lists possible categories but does not assign any values to them initially, as they are all commented out. To adjust the log level for a specific category, it is necessary to remove the "#" symbol and set the desired log level.

To implement the modification, either restart `sharp_am` or send a SIGHUP signal.

The provided package file contains the following text:

```
# A line starting with "#" is a comment line
# To set a specific category, remove the "#"
#
# Possible levels are: 1-5
# 1 – Error, 2-Warning, 3-Info, 4-Debug, 5-Verbose

#SHARP_GENERAL = 3
#SHARP_SR = 3
#SHARP_SMX = 3
#SHARP_SIGNAL = 3
#SHARP_MADS = 3
#SHARP_JOBS = 3
#SHARP_RESERVE = 3
#SHARP_FGRAPH = 3
#SHARP_FABRIC = 3
#SHARP_TREES = 3
```

9 Operating NVIDIA SHARP in Dynamic Trees Allocation Mode

9.1 SHARP Trees

A SHARP tree defines a set of switches and their connected links to be used by one or more SHARP jobs.

- A single tree can be used by multiple jobs, as long as they are using different areas of the tree.
- A single job can also utilize multiple trees, in case the job is operating on multiple rails, while each rail can use a different tree.

9.2 Dynamic vs. Static Allocation Mode

In SHARP v3.3 and earlier, sharp_am used to operate in "Static trees" mode by default. In this mode, SHARP trees were created in the sharp_am initialization phase. When a new SHARP job started, it was assigned to one of the existing SHARP trees that was available to operate the job.

As of SHARP v3.4, sharp_am's default operation mode is "Dynamic trees" mode. This mode is recommended as the preferred option to use.

When sharp_am operates in Dynamic trees mode, trees are not created in the initialization phase. Instead, they are created per job, immediately assigned to the job that requires them, and are deleted once the job ends.

The Dynamic trees mode of operation has some benefits over the Static trees mode, as it defines the SHARP configuration on the switches only when necessary, and enables better utilization of the fabric resource. There are various scenarios in which a Static mode of operation may respond with “No resources” to a SHARP job request, while in Dynamic mode, the SHARP job would be fulfilled.

9.3 Dynamic Trees Allocation Algorithms

sharp_am takes multiple factors into consideration when deciding on the trees to create for each job. Initially, the allocated trees must meet the job's requirements. However, sharp_am also aims to allocate trees in a manner that preserves available links and switch resources for future jobs that may be needed.

The distinction between the combinations of trees that can be created in a regular FatTree versus a Quasi Fat Tree are significant. Consequently, sharp_am offers two distinct algorithms to determine how trees should be created for each job. One algorithm is optimized for SuperPOD fabrics, while the other is tailored for Quasi Fat Trees (QFTs).

Note the following:

- Only one algorithm can be used at a given time
- sharp_am should be restarted when switching algorithms
- Under specific circumstances, when employing one algorithm and running multiple jobs simultaneously, sharp_am might potentially declare "No resources" for a particular job

request. However, if the other algorithm were utilized, the resources would be distributed differently, fulfilling all job requests.

Please note that there are no definitive right or wrong algorithms for any given topology, as each algorithm comes with its own advantages and limitations. Additionally, certain features are exclusive to specific algorithms.

It is recommended to consult with NVIDIA experts regarding the suitable algorithm for your system. You can contact us through either of the following methods:

E-mail: Enterprisesupport@nvidia.com

Enterprise Support page: https://www.nvidia.com/en-us/support/enterprise

9.4 Configuring Dynamic Trees Allocation Mode

sharp_am's default operation mode is set to Dynamic trees mode. Follow the instructions below in case you have the mode set to Static trees or a change of algorithm is required.

To operate in Dynamic trees mode, make sure dynamic_tree_allocation parameter is set to TRUE.

By default, the SuperPOD-oriented algorithm is used. To switch to the QFT-oriented algorithm, use the dynamic_tree_algorithm parameter.

If the number of root switches in the fabric is larger than 126 when using the SuperPOD-oriented algorithm, it is desired to modify max_trees_to_build to be equal to the number of root switches.

Note that sharp_am restart is required for the configuration to take effect.

9.5 Limitations

- Dynamic trees allocation mode is currently available for fat-tree and Quasi-Fat-Tree (QFT) topologies only, and is not supported for Dragonfly or hypercube topologies. In case sharp_am is configured to operate in Dynamic mode and the topology does not match, sharp_am will automatically operate in Static mode.
- When operating in Dynamic trees mode, ibdiagnet may print warning messages about the existence of multiple distinct trees with the same tree ID. In Dynamic trees mode, this is a valid situation and these warnings should be ignored.
  Warning example: `-W- <> - In Node <> found root tree (parent qpn <>) which is already exists for treeID: <>`
  Note: You can avoid this warning by adding the following parameters to the ibdiagnet command line: `--sharp_opt ad_hoc`
- Dynamic trees creation does not support a case in which all root switches are down and restarted. If such a scenario takes place, sharp_am should be restarted once the root switches are up and running.

10 SHARP Application Awareness

Different entities, such as tenants, jobs, and others - all considered "applications", can be bound together through SHARP. In other words, SHARP can be application-aware, providing isolation and a set of attributes to each application.

The most common example related to SHARP is an application being a SLURM job, created to perform a certain task.

To be application-aware, the following conditions should be met:

- sharp_am should run with config parameter reservation_mode set to TRUE
- sharp_am should operate from within UFM, as UFM REST-API use is a must to operate in this mode

Once sharp_am operates in reservation_mode, no compute host is allowed to ask for a SHARP job, unless it was specifically requested via UFM REST-API.

The REST-API enables to define a set of hosts that function as a single application, with an option to define also a pkey that they share and a limit of resources that can be used by the app.

With this method, the admin of the fabric can control which compute hosts are allowed to leverage SHARP, and can even limit the number of trees allocated per application. By default, once an application is declared, there is no limit for the number of trees it can allocate. In case a limitation is required, it is advised that the minimum value be the same as the number of rails in the system.

Full details of the REST-API can be found in UFM REST-API document.

11 Operating NVIDIA SHARP with PKeys

SHARP can operate in a system that has either a single static special PKey or a system that dynamically allocates PKeys.

11.1 Defining a Special PKey

Use this method when SHARP is intended to operate exclusively on a single known PKey. To implement this, adjust the ib_qpc_pkey field to the desired PKey value in the SHARP configuration file. Remember to ensure the membership bit is properly set, which entails setting both bit 0x8000 and the corresponding pkey value at all times.

11.2 Supporting Dynamic PKeys

SHARP, when operating from the UFM management host, enables dynamic declaration of PKeys. This feature is facilitated by the reservation_mode config parameter mentioned in SHARP Application Awareness section. Configuring SHARP to operate in reservation_mode via the UFM config file allows UFM to relay PKeys information to SHARP.

- To enable this functionality, make sure to set the following parameter in the UFM gv.cfg file:
  `sharp_allocation_enabled = true`
- Restart UFM to apply the updated settings.

Note that any creation, removal, or modification of PKeys should be performed using the UFM PKeys REST-API, detailed here.

12 Disabling SHARP on Specific Network Devices in OpenSM

Disabling SHARP on a specific network switch device can be performed using OpenSM device configuration file by performing the following:

1. Define a port group with the specified network device in the port groups file. The group is specified by the `pgrp_policy_file` parameter in the OpenSM configuration file.
   For example:

   ```
   port-group
   name: NON_SHARP_SWITCHES
   port-guid: 0x0002c90000000001
   end-port-group
   ```

2. Configure OpenSM to disable SHARP on the devices of the specified port groups in device configuration file specified by the `device_configuration_file` parameter in OpenSM configuration file.

   ```
       port-conf
           port-group-name: NON_SHARP_SWITCHES
           sharp-enabled: 0
       end-port-conf
   ```

3. Reload OpenSM.

13 Testing NVIDIA SHARP Setup

13.1 Aggregation Trees Diagnostics

Run *ibdiagnet* utility with SHARP diagnostics option.

```
$ibdiagnet --sharp
```

Check *fabric summary* table in *ibdiagnet* output for the number of identified aggregation nodes. For example:

```
Fabric Summary

Total Nodes            : 24
IB Switches            : 4
IB Channel Adapters    : 16
IB Aggregation Nodes   : 4
IB Routers             : 0

Total number of links  : 24
Links at 4x50          : 24

Master SM: Port=1 LID=1 GUID=0x248a070300a28c4d devid=4119 Priority:0 Node_Type=CA Node_Description=pnemo HCA-2
Standby SM : No Standby SM
```

Check *summary* table in *ibdiagnet* output for errors in SHARP diagnostics stage. For example:

```
Summary
-I- Stage                    Warnings   Errors     Comment
-I- Discovery                0          0
-I- Lids Check               0          0
-I- Links Check              0          0
-I- Subnet Manager           0          0
-I- Port Counters            0          0
-I- Nodes Information        0          0
-I- Speed / Width checks     0          0
-I- Alias GUIDs              0          0
-I- Virtualization           0          0
-I- Partition Keys           0          0
-I- Temperature Sensing      0          0
-I- SHARP                    0          0
```

Check in SHARP diagnostics output file (/var/tmp/ibdiagnet2/ibdiagnet2.sharp) that SHARP aggregation trees are configured in the subnet.

For example: count number of configured aggregation trees constructed by Aggregation Manager using *grep* command:

```
$cat /var/tmp/ibdiagnet2/ibdiagnet2.sharp | grep -c TreeID
126
```

Note that when operating in dynamic trees mode, ibdiagnet may print warning messages about the existence of multiple distinct trees with the same tree ID. In dynamic trees mode, this is a valid situation and these warnings should be ignored.

Warning example:

```
-W- <> - In Node <> found root tree (parent qpn <>) which is already exists for treeID: <>
```

13.2 NVIDIA SHARP Hello

NVIDIA SHARP distribution provides sharp_hello test utility for testing SHARP's end-to-end functionality on a compute node. It creates a single SHARP job and sends a barrier request to SHARP Aggregation node.

Help

```
$sharp_hello -h
usage:  sharp_hello <-d | --ib_dev> <device> [OPTIONS]
OPTIONS:
        [-d | --ib_dev]      - HCA to use
        [-v | --verbose]     - libsharp coll verbosity level(default:2)
                                   Levels: (0-fatal 1-err 2-warn 3-info 4-debug 5-trace)
        [-V | --version]     - print program version
        [-h | --help]        - show this usage
```

Example #1

```
$ sharp_hello -d mlx5_0:1 -v 3
[thor001:0:15042 - context.c:581] INFO job (ID: 12159720107860141553) resource request quota: ( osts:0
 user_data_per_ost:0 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[thor001:0:15042 - context.c:751] INFO tree_info: type:LLT tree idx:0 treeID:0x0 caps:0x6 quota: ( osts:167
 user_data_per_ost:1024 max_groups:167 max_qps:1 max_group_channels:1)
[thor001:0:15042 - comm.c:393] INFO [group#:0] group id:a tree idx:0 tree_type:LLT rail_idx:0 group size:1 quota:
(osts:2 user_data_per_ost:1024) mgid: (subnet prefix:0xff12a01bfe800000 interface id:0x3f020000000a) mlid:c007
Test Passed.
```

Example #2

```
$ SHARP_COLL_ENABLE_SAT=1 sharp_hello -d mlx5_0:1 -v 3

[swx-dgx01:0:59023 - context.c:581] INFO job (ID: 15134963379905498623) resource request quota: ( osts:0
 user_data_per_ost:0 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[swx-dgx01:0:59023 - context.c:751] INFO tree_info: type:LLT tree idx:0 treeID:0x0 caps:0x6 quota: ( osts:167
 user_data_per_ost:1024 max_groups:167 max_qps:1 max_group_channels:1)
[swx-dgx01:0:59023 - context.c:755] INFO tree_info: type:SAT tree idx:1 treeID:0x3f caps:0x16
[swx-dgx01:0:59023 - comm.c:393] INFO [group#:0] group id:3c tree idx:0 tree_type:LLT rail_idx:0 group size:1
 quota: (osts:2 user_data_per_ost:1024) mgid: (subnet prefix:0xff12a01bfe800000 interface id:0xd6060000003c)
mlid:c004
[swx-dgx01:0:59023 - comm.c:393] INFO [group#:1] group id:3c tree idx:1 tree_type:SAT rail_idx:0 group size:1
 quota: (osts:64 user_data_per_ost:0) mgid: (subnet prefix:0x0 interface id:0x0) mlid:0
Test Passed
```

13.3 NVIDIA SHARP Benchmark

NVIDIA SHARP distribution provides a source code for the benchmark to test native SHARP low-level performance for allreduce and barrier operations.

Source code:

```
$module load hpcx
$HPCX_SHARP_DIR/share/sharp/examples/mpi/coll/
```

Build and run instructions:

```
$module load hpcx
$HPCX_SHARP_DIR/opt/Mellanox/sharp/share/sharp/examples/mpi/coll/README
```

13.3.1 NVIDIA SHARP Benchmark Script

NVIDIA SHARP distribution provides a test script which executes OSU (allreduce, barrier) benchmark running with and without NVIDIA SHARP. To run the NVIDIA SHARP benchmark script, the following packages are required to be installed.

- ssh
- pdsh
- environment-modules.x86_64

You can find this script at $HPCX_SHARP_DIR/sbin/sharp_benchmark.sh after loading the HPC-X module. This script should be launched from a host running SM and Aggregation Manager. It receives a list of compute nodes from SLURM allocation or from “hostlist” environment variable. “hostlist” is a comma-separated list which requires hca environment variables to be supplied. It runs OSU allreduce and barrier benchmarks with and without NVIDIA SHARP.

Help

```
This script includes OSU benchmarks for MPI_Allreduce and MPI_Barrier blocking collective operations.
Both benchmarks run with and without using SHARP technology.

Usage: sharp_benchmark.sh [-t] [-d] [-h] [-f]
        -t - tests list (e.g. sharp:barrier)
        -d - dry run
        -h - display this help and exit
        -f - supress error in prerequsites checking

Configuration:
 Runtime:
  sharp_ppn - number of processes per compute node (default 1)
  sharp_ib_dev - Infiniband device used for communication. Format <device_name>:<port_number>.
                 For example: sharp_ib_dev="mlx5_0:1"
                 This is a mandatory parameter. If it's absent, sharp_benchmark.sh tries to use the first active
device on local machine
  sharp_groups_num - number of groups per communicator. (default is the number of devices in sharp_ib_dev)
  sharp_num_trees - number of trees to request. (default num tress based on the #rails and #channels)
  sharp_job_members_type - type of sharp job members list. (default is SHARP_MEMBER_LIST_PROCESSES_DATA)
  sharp_hostlist - hostnames of compute nodes used in the benchmark. The list may include normal host names,
                   a range of hosts in hostlist format. Under SLURM allocation, SLURM_NODELIST is used as a default
  sharp_test_iters - number of test iterations (default 10000)
  sharp_test_skip_iters - number of test iterations (default 1000)
  sharp_test_max_data - max data size used for testing (default and maximum 4096)
 Environment:
  SHARP_INI_FILE - takes configuration from given file instead of /labhome/danielk/.sharp_benchmark.ini
  SHARP_TMP_DIR - store temporary files here instead of /tmp
  HCOLL_INSTALL - use specified hcoll install instead from hpcx

Examples:
  sharp_ib_dev="mlx5_0:1" sharp_benchmark.sh  # run using "mlx5_0:1" IB port. Rest parameters are loaded from /
labhome/danielk/.sharp_benchmark.ini or default
  SHARP_INI_FILE=~/benchmark.ini  sharp_benchmark.sh # Override default configuration file
  SHARP_INI_FILE=~/benchmark.ini  sharp_hostlist=ajna0[2-3]  sharp_ib_dev="mlx5_0:1" sharp_benchmark.sh # Use
specific host list
  sharp_ppn=1 sharp_hostlist=ajna0[1-8] sharp_ib_dev="mlx5_0:1" sharp_benchmark.sh  -d # Print commands without
actual run

Dependencies:
  This script uses "python-hostlist" package. Visit https://www.nsc.liu.se/~kent/python-hostlist/ for details
```

14 NVIDIA SHARP Collective Library

NVIDIA SHARP distribution provides a collective library implementation with high level API to easily integrate into other communication runtime stacks, such as MPI, NCCL and others.

The SHARP collective library offers collective operations such as Barrier, Allreduce, Reduce, Bcast, Reduce-scatter, and Allgather. It accommodates datatypes including 16/32/64-bit Integer/Floating-point, as well as 16-bit Bfloat and 8-bit Integer.

14.1 NVIDIA SHARP Library Flags

14.1.1 NVIDIA SHARP Configuration Flags

As of NVIDIA SHARP version 2.7.0, sharpd daemon no longer exists, and its activity is now performed from application process.

The previous sharpd configuration is now done from the application command-line instead using the following flags.

Flag	Description
`SHARP_LOG_VERBOSTIRY`	Log verbosity level 1 - Errors 2 - Warnings 3 - Info 4 - Debug 5 - Trace **Default:** 2
`SHARP_LOG_FILE`	Log file **Default:** stdout The log file name accepts the following modifiers in the file name to create a unique file • **%D** date as DDMMYYYY • **%T** thread ID • **%H** host name
`SHARP_SMX_SOCK_INTERFACE`	Network interface to be used by SMX: empty string (default) - Use interface used for AM connection **Default**: (null)
`SHARP_SMX_SOCK_ADDR_FAMILY`	Determines which address family will be used in SMX's sockets. The value needs to be one of the following: { ipv4, ipv6 } IPv4 support is required even when choosing the ipv6 option. **Default**: ipv6
`SHARP_SMX_UCX_INTERFACE`	Network interface to be used by SMX for UCX connections: empty string (default) - Use interface used for AM connection **Default**: (null)

14.1.2 NVIDIA SHARP Resource Tuning for Low Latency Operations

The following SHARP library flags can be used when running NVIDIA SHARP collectives.

Flag	Description
SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST	Maximum payload per OST (outstanding transactions). Value 0 means "allocate default value". **Valid values**: • 0 (default) • 128-1024
SHARP_COLL_JOB_QUOTA_OSTS	Maximum job (per tree) OST quota request. Value 0 means "allocate default quota". **Default**: 0
SHARP_COLL_JOB_QUOTA_MAX_GROUPS	Maximum number of groups (comms) quota request. Value 0 means "allocate default value". **Default**: 0
SHARP_COLL_OSTS_PER_GROUP	Number of OSTs per group. **Default**: 8
SHARP_COLL_JOB_QUOTA_MAX_QPS_PER_PORT	Maximum QPs/port quota request. Value 0 means "allocate default value".

14.1.3 NVIDIA SHARP Streaming Aggregation

The following NVIDIA SHARP library flags can be used to enable Streaming Aggregation Tree (SAT) and tuning.

Flag	Description
SHARP_COLL_ENABLE_SAT	Enables SAT capabilities. **Default**: 0 (Disabled) The Maximum message size SAT protocol support is 1073741792 Bytes (32B less than 1GB).
SHARP_COLL_SAT_THRESHOLD	Message size threshold to use SAT on generic allreduce collective requests. **Default**: 16384
SHARP_COLL_SAT_LOCK_BATCH_SIZE	SAT lock batch size. Set this to “1” if multiple communicators want to use SAT resources. **Valid range**: 1-65535. **Default**: 65535 (Infinity)
SHARP_COLL_LOCK_ON_COMM_INIT	Get SAT Lock resource during communicator init if lock batch size is Infinity. Return failure if failed to lock **Default:** 0 (Disabled), 1(Enabled) with NCCL SHARP plugin
SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD	Lazy group resource allocation. 0 - Disable lazy allocation, allocate group resource at communicator create time #n - Allocate sharp group resource after #n collective calls requested on the group **Default**: 1
SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE	SAT (Streaming Aggregation Tree) exclusive lock mode for job. Possible values: • 0 - no exclusive lock • 1 - try exclusive lock • 2 (**default**)- force exclusive lock

14.1.4 SHARP Miscellaneous Tuning

Flag	Description
`SHARP_COLL_ENABLE_CUDA`	Enables CUDA GPU support. Possible values: • 0 - disable • 1 - enable • 2 **(default)** - try
`SHARP_COLL_PIPELINE_DEPTH`	Size of fragmentation pipeline for larger collective payload. **Default**: 64
`SHARP_COLL_ENABLE_MCAST_TARGET`	Enables MCAST target on NVIDIA SHARP collective operations. Possible values: • 0 (**default**) - disable • 1 - enable
`SHARP_COLL_MCAST_TARGET_GROUP_SIZE_THRESHOLD`	Group size threshold to enable mcast target. **Default**: 2
`SHARP_COLL_POLL_BATCH`	Defines the number of CQ completions to poll on at once. **Valid range:** 1-16 **Default**: 4
`SHARP_COLL_ERROR_CHECK_INTERVAL`	Interval in milliseconds that indicates the time between the error checks. If you set the interval as 0, error check is not performed. **Default**: 180,000
`SHARP_COLL_JOB_NUM_TREES`	Number of SHARP trees to request. 0 means requesting the number of trees based on the number of rails and the number of channels. **Default**: 0
`SHARP_COLL_GROUPS_PER_COMM`	Number of NVIDIA SHARP groups per user communicator. **Default**: 1
`SHARP_COLL_JOB_PRIORITY`	Job priority. **Valid values:** 0-10 **Default**: 0
`SHARP_COLL_ENABLE_PCI_RELAXED_ORDERING`	Enable PCI relaxed order memory access. Possible values: • 0 - disable • 1 - enable • 2 (**default**) - auto

For the complete list of SHARP_COLL tuning options, run the `sharp_coll_dump_config` utility:
`$HPCX_SHARP_DIR/bin/sharp_coll_dump_config`

15 Using NVIDIA SHARP with Open MPI

NVIDIA SHARP library is integrated into HCOLL collective library to offload collective operations in MPI applications.

The following basic flags should be used in environment to enable NVIDIA SHARP protocol in the HCOLL middleware. For the rest of flags, please refer to NVIDIA SHARP Release Notes.

15.1 HCOLL Library Flags

The following HCOLL flags can be used when running NVIDIA SHARP collective with mpirun utility.

Flag	Description
`HCOLL_ENABLE_SHARP`	Sets whether SHARP should be used. Possible values: • 0 (**default**) – do not use NVIDIA SHARP • 1 - probe NVIDIA SHARP availability and use it • 2 - force to use NVIDIA SHARP • 3 - force to use NVIDIA SHARP for all MPI communicators • 4 - force to use NVIDIA SHARP for all MPI communicators and for all supported collectives (barrier, allreduce)
`SHARP_COLL_LOG_LEVEL`	NVIDIA SHARP coll logging level. Messages with a higher or equal level to the selected will be printed. Possible values: • 0 - fatal • 1 - error • 2 (**default**) - warn • 3 - info • 4 - debug • 5 - trace
`HCOLL_SHARP_NP`	Number of nodes (node leaders) threshold in the communicator to create NVIDIA SHARP group and use NVIDIA SHARP collectives. **Default**: 4
`HCOLL_SHARP_UPROGRESS_NUM_POLLS`	Number of unsuccessful polling loops in libsharp coll for blocking collective wait before calling user progress (HCOLL, OMPI). **Default**: 999

`HCOLL_ALLREDUCE_SHARP_MAX`
`(or)`
`HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX`
	Maximum allreduce size run through NVIDIA SHARP. A message size greater than the above the specified value by this parameter will fall back to non-SHARP-based algorithms (multicast based or non-multicast based).
	The threshold is calculated based on the group resources.
	Threshold = #OSTS * Payload_per_ost
	Default: Dynamic

15.1.1 Example of Allreduce with Default Settings with SHARP Enable

```
$ mpirun -np 128 -map-by ppr:1:node  -x UCX_TLS=dc,shm,self  -x HCOLL_ENABLE_SHARP=3 -x SHARP_COLL_ENABLE_SAT=1
$HPCX_OSU_DIR/osu_allreduce

# OSU MPI Allreduce Latency Test v5.6.2
# Size       Avg Latency(us)
```

```
4                    7.44
8                    8.43
16                   7.81
32                   8.55
64                   9.06
128                  8.44
256                  9.41
512                  8.50
1024                 9.03
2048                10.43
4096                42.61
8192                37.93
16384               15.48
32768               16.26
65536               17.62
131072              23.09
262144              33.90
524288              58.98
1048576            101.53
```

16 Using NVIDIA SHARP with NVIDIA NCCL

RDMA and SHARP collectives are enabled with NVIDIA NCCL ('nickel') collective communication library through the NCCL-SHARP plugin.

The NCCL-SHARP plugin is distributed through the following channels:

- Binary distribution with HPC-X. The plugin will be loaded in the environment with HPC-X modules and NCCL will load it automatically. The plugin can be built from the source of other CUDA versions.
- Source distribution: https://github.com/Mellanox/nccl-rdma-sharp-plugins
  User can build the plugin from the source and set LD_LIBRARY_PATH to use it by NCCL.

16.1 Requirements

- NVIDIA ConnectX-6 HDR and above
- NVIDIA Quantum HDR switch and above
- MNLX_OFED
- GPUDirectRDMA
  - Plugin
  - Source code repository

  It is important to verify that the GPUDirect RDMA kernel module is properly loaded on each of the computing systems where you plan to run the job that requires the GPUDirect RDMA.

  To check whether the GPUDirect RDMA module is loaded, run:

  ```
  # service nv_peer_mem status
  ```

  To run this verification on other Linux flavors:

  ```
  # lsmod | grep nv_peer_mem
  ```
- NCCL version 2.7.3 or higher
  Please refer to NVDIA's Developer Guide for more details: https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/index.html

16.2 Control Flags

The following environment variables enable the SHARP aggregation with NCCL when using the NCCL-SHARP plugin.

- NCCL variables:
  - `NCCL_COLLNET_ENABLE=1`
  - `NCCL_ALGO=CollNet` (Required to overcome a bug in NCCL <= 2.7.8)
- SHARP variables:
  - For guaranteed SAT resources on initialization: These options are enabled by default with NCCL SHARP Plugin version >= 2.1.x. Users can enable explicitly using following variables:
    - SHARP_COLL_LOCK_ON_COMM_INIT=1 (

        - SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD=0
    - [Optional] SHARP_COLL_LOG_LEVEL=3
- NCCL SHARP Plugin variables:
    - NCCL_SHARP_DISABLE
        - NCCL SHARP Streaming aggregation is supported on a single NCCL communicator/process group (PG). Applications can selectively enable SHARP on specific Process Group (PG) by setting this variable in the application before creating the PG
    - NCCL_SHARP_GROUP_SIZE_THRESH
        - Application can set this code option to selectively enable SHARP on the PG based on the group size
    - NCCL_IBEXT_DISABLE
        - NCCL plugin will be disabled and NCCL native communication transports will be used instead

16.3 Cluster Topology for Using NVIDIA SHARP SAT with NCCL

On systems with multiple GPUs and multiple HCAs, NCCL creates an aggregation streaming flow (NCCL Ring/Channel) per HCA rail. It is required to build the cluster topology in such a way that leaf level switches connected to same HCA rail from each server.

16.4 NCCL Benchmark Example

The sanity performance of the setup can be verified with NCCL tests. Please refer to NCCL tests here: https://github.com/NVIDIA/nccl-tests

Example:

```
$ mpirun  -np 1024  -map-by ppr:8:node -x UCX_TLS=dc,shm,self   -x LD_LIBRARY_PATH=/sw/nccl/build/lib::/sw/nccl-
rdma-sharp-plugins/install/lib:$LD_LIBRARY_PATH -x  NCCL_COLLNET_ENABLE=1  all_reduce_perf -b 4 -e 2G -f 2 -g 1 -w
50 -n 50

           4             1   float     sum    44.53    0.00    0.00  3e-05    44.21    0.00    0.00  3e-05
           8             2   float     sum    45.42    0.00    0.00  3e-05    45.85    0.00    0.00  3e-05
          16             4   float     sum    46.34    0.00    0.00  3e-05    45.84    0.00    0.00  2e-05
          32             8   float     sum    46.20    0.00    0.00  2e-05    46.56    0.00    0.00  2e-05
          64            16   float     sum    46.00    0.00    0.00  2e-05    48.33    0.00    0.00  2e-05
         128            32   float     sum    48.77    0.00    0.01  2e-05    47.23    0.00    0.01  2e-05
         256            64   float     sum    47.88    0.01    0.01  2e-05    47.85    0.01    0.01  2e-05
         512           128   float     sum    51.44    0.01    0.02  3e-05    48.66    0.01    0.02  3e-05
        1024           256   float     sum    51.27    0.02    0.04  4e-05    51.78    0.02    0.04  4e-05
        2048           512   float     sum    57.93    0.04    0.07  4e-05    56.45    0.04    0.07  4e-05
        4096          1024   float     sum    57.32    0.07    0.14  4e-05    93.51    0.04    0.09  4e-05
        8192          2048   float     sum    106.4    0.08    0.15  4e-05    59.70    0.14    0.27  4e-05
       16384          4096   float     sum    103.0    0.16    0.32  4e-05    58.23    0.28    0.56  4e-05
       32768          8192   float     sum    74.85    0.44    0.87  4e-05    137.8    0.24    0.48  4e-05
       65536         16384   float     sum    96.71    0.68    1.35  4e-05    92.89    0.71    1.41  4e-05
      131072         32768   float     sum    115.6    1.13    2.27  4e-05    120.7    1.09    2.17  4e-05
      262144         65536   float     sum    197.7    1.33    2.65  4e-05    167.6    1.56    3.13  4e-05
      524288        131072   float     sum    222.7    2.35    4.70  4e-05    239.2    2.19    4.38  4e-05
     1048576        262144   float     sum    280.9    3.73    7.46  4e-05    197.7    5.30   10.60  4e-05
     2097152        524288   float     sum    218.0    9.62   19.22  4e-05    213.9    9.81   19.59  4e-05
     4194304       1048576   float     sum    257.6   16.28   32.53  4e-05    254.7   16.47   32.90  4e-05
     8388608       2097152   float     sum    354.3   23.68   47.31  4e-05    523.5   16.02   32.02  4e-05
    16777216       4194304   float     sum    505.9   33.16   66.26  4e-05    484.1   34.66   69.24  4e-05
    33554432       8388608   float     sum    639.2   52.50  104.89  4e-05    678.6   49.45   98.80  4e-05
    67108864      16777216   float     sum   1358.2   49.41   98.72  4e-05   1048.6   64.00  127.87  4e-05
   134217728      33554432   float     sum   1737.2   77.26  154.37  4e-05   1777.6   75.51  150.86  4e-05
   268435456      67108864   float     sum   4359.5   61.58  123.03  4e-05   4262.3   62.98  125.83  4e-05
   536870912     134217728   float     sum   5619.7   95.53  190.88  4e-05   5699.0   94.20  188.22  4e-05
  1073741824     268435456   float     sum    12169   88.23  176.30  4e-05    11508   93.30  186.42  4e-05
  2147483648     536870912   float     sum    22618   94.94  189.70  4e-05    21814   98.44  196.70  4e-05
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 41.2497
```

```
#
```

17 SHARP Cleanup

This feature enables cleaning up all SHARP-related definitions when it is no longer desired to operate with it. The cleanup helps in leveraging the full potential of the switch capabilities without allocating resources for SHARP.

Furthermore, when an error takes place due to stuck jobs, uncleaned memory, or other scenarios, a cleanup should help in solving the error without the need for a switch reboot.

To perform a cleanup, follow the steps below.

1. Stop sharp_am or make sure sharp_am is not running.
2. Verify that there are no active SHARP jobs running. In case there are, be aware that cleaning SHARP resources will terminate these jobs, so either wait for them to finish or stop them gracefully.
3. Run sharp_am with the following parameters: `sharp_am --log_verbosity 3 -- clean_and_exit TRUE`
   sharp_am will clean all ANs mentioned in the smdb file and exit.
4. To verify success, look into the sharp_am log file for a message in the following format: `"Sent clean command to <> ANs, successes: <>, fails: <>"`

18 Deployment Guide Revision History

Revision	Date	Description
3.5.0	November 7, 2023	• Added sharp_am Log and Dump Files • Updated NVIDIA SHARP Collective Library

3.4.0	August 07, 2023	• Added **UFM Appliance Firewall Settings** under sharp_am Network Interfaces
		• Updated Dynamic vs. Static Allocation Mode
		• Updated Dynamic Trees Allocation Algorithms
		• Added Operating NVIDIA SHARP with PKeys
		• Under NVIDIA SHARP Collective Library:
		  • Updated `SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST` flag description
		  • Updated the default value of `SHARP_COLL_OSTS_PER_GROUP` flag
		  • Updated `SHARP_COLL_SAT_LOCK_BATCH_SIZE` flag description
		  • Added the default value for `SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD` flag
		  • Updated the default value for `SHARP_COLL_ENABLE_MCAST_TARGET` flag
		  • Updated `SHARP_COLL_POLL_BATCH` description
		  • Added the default value for `SHARP_COLL_ERROR_CHECK_INTERVAL`
		  • Updated `SHARP_COLL_JOB_PRIORITY` flag description
		  • Updated `SHARP_COLL_ENABLE_PCI_RELAXED_ORDERING` flag description

3.3.0	May 1, 2023	• Added Configuring Dynamic Trees Allocation Mode • Added SHARP Application Awareness • Updated Operating NVIDIA SHARP in Dynamic Trees Allocation Mode • Updated smx_sock_interface entry under **Network Interfaces Configuration** table in sharp_am Network Interfaces
3.1.1	November 30, 2022	Added SHARP Cleanup.
3.1.0	October 31, 2022	• Added Operating NVIDIA SHARP in Dynamic Trees Allocation Mode • Updated Modifying NVIDIA SHARP Aggregation Manager Configuration • Updated Aggregation Trees Diagnostics under Testing NVIDIA SHARP Setup

Revision	Date	Description
2.7.0	May 3, 2022	• Updated Setup Requirements under Setting up NVIDIA SHARP Environment • Updated NVIDIA Hardware Capabilities and Limitations under Setting up NVIDIA SHARP Environment • Updated Running NVIDIA SHARP Aggregation Manager (AM) Daemons • Updated Number of Simultaneous Streaming Aggregation Flows under Disabling SHARP on Specific Network Devices in OpenSM • Added NVIDIA SHARP Configuration Flags under NVIDIA SHARP Collective Library • Added NCCL_IBEXT_DISABLE under Control Flags in Using NVIDIA SHARP with NVIDIA NCCL
2.6.1	December 5, 2021	Updated the following sections: • Updated NVIDIA Hardware Capabilities and Limitations • Updated the SHARP_COLL_ENABLE_SAT flag in NVIDIA SHARP Streaming Aggregation • Updated SHARP variables and added NCCL SHARP Plugin variables in Control Flags

19 Release Notes Revision History

19.1 Release Notes Change History

19.1.1 Changes and New Features History

Feature/Change	Description
Rev 3.4.0	
Parameter Changes	dynamic_tree_allocation sharp_am A boolean parameter, indicates whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Update**: Default value is now True
Rev 3.3.0	
Syslog Capabilities	Added support for a new syslog capability to libsharp. Syslog verbosity level can now be controlled using the SHARP_SYSLOG_VERBOSITY environment variable.
Dynamic Trees Allocation Algorithms	Added support for selecting one of two algorithms that determine how trees should be created for each SHARP job. One algorithm is optimized for SuperPOD fabrics, while the other is optimized for Quasi Fat Trees (QFTs). For further information, please see Dynamic Trees Allocation Algorithms section.
REST API Jobs Query	Added support for retrieving the status of the current active SHARP jobs along with the structure of the trees assigned to them. Note that this information is retrieved via REST-API and requires the use of UFM.
Unhealthy Ports	Added support in OpenSM to inform SHARP of dangling or unhealthy links in order to avoid their use in SHARP jobs.
Bug Fixes	See Bug Fixes section.
Rev 3.2.0	
High Availability in sharp_am Network Interfaces	sharp_am leverages multiple network interfaces of the management host to provide high availability in case of a network interface failure. For further information, please see sharp_am Network Interfaces.
Reliable Multicast	Added support for SHARP to leverage reliable multicast option with NVIDIA Quantum-2.
SM Data	Removed support for reading sm data by a client application. The API functions sharp_request_sm_data, sharp_get_sm_data_buf_len, and sharp_get_sm_data have been removed and can no longer be used. In addition, the configuration parameter ftree_ca_order_file is ignored in sharp_am.
Bug Fixes	See Bug Fixes section.
Rev 3.1.1 LTS	
SHARP Cleanup	Added the ability to clean up all SHARP-related definitions either to spare resources or to contribute to the recovery from an error.
General	Updated MLNX_OFED and firmware versions in General Information section.

Bug Fixes	See Bug Fixes section.
Rev 3.1.0	
Aggregation Manager (AM)	Added support for dynamic creation of trees instead of static allocation when SHARP is initialized.
Rev 3.0.1	
Bug Fixes	See Bug Fixes section.
Rev 3.0.0	
General	Added support for executing multiple jobs that aggregate data through the same set of switches, while each job utilizes a different set of links.
	SHARP logic is now application-aware with UFM capabilities. SHARP jobs can be assigned an App-ID, which can be used as a reference to the customer application performing these jobs. For further information, please refer to UFM SLURM Integration Appendix in UFM UM.
	Added the option to limit the SHARP resources that applications are allowed to consume. For further information, please refer to UFM SLURM Integration Appendix in UFM UM.
AM	Modified the default resources provided to LLT & SAT jobs. This enables operation of a larger amount of SAT jobs in parallel to few LLT jobs (please see the first three entries in the table below).
libsharp	SHARP jobs are now executed in exclusive lock mode by default (please see SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE in the table below).
Rev 2.7.0	
Switches	Added support for NVIDIA Quantum-2 switches with NDR speed
Adapter Cards	Added support for NVIDIA ConnectX-7 adapter card with 400 Gb/s speed
SHARPD	sharpd daemon process has been removed. sharpd-related activity is now performed from the user application process
AM	Upon restart of AM, it no longer needs to wait for all concurrent jobs to finish before being able to accept new jobs
	Added a mechanism that periodically checks for errors in Aggregation Trees and attempts to fix them
General	Added support for new data types BFLOAT16, INT8 and UNIT8 for performing reduction operations
Rev 2.6.1	
General	Added support for running libsharp_coll from SHARP 2.6.1 with SHARPD from SHARP 2.4.0 – 2.6.1
General	Added information about updatable configuration parameters in the configuration file and help menu
Network	Added support for keep-alive on connections to SHARPD
Network	Added support for asynchronous connections
Network	Disabled UCX listener as default in SHARP Aggregation Manager
AM	Added support for the non-default subnet prefix

AM	Added support for DF+ topologies with more than two-level islands
SHARPD	Added support for caching AM address
Rev 2.5.0	
Resource Management	Added support for exclusive lock requests for streaming aggregation jobs.
Network	Enabled connection keep-alive between SHARPD and Aggregation Manager.
Rev 2.4.3	
General	Added support for identifying Aggregation Nodes based on SMDB.
General	Improved minhop tables calculation.
General	Added a new API for querying events.
Rev 2.1.4	
sharp_am/sharpd/ libsharp_coll: Streaming Aggregation	Added support for Streaming Aggregation over ConnectX-6 adapter card and Quantum switch.
libsharp_coll: GPU Accelerator	Added support for NVIDIA GPU buffers.
sharp_am: OOB	Added support for identifying the topology type from the OpenSM SMDB file.
sharp_am: Reboot	Fixed an issue where recovery failed after reboot of all switches in the cluster.
Rev 2.0.0	
sharp_am/sharpd/ libsharp_coll	Added support for the following NVIDIA Quantum switch capabilities: • Performing data operations on new data types (unsigned short, short, and short floating point data types) • 1K OST payload
sharp_am/sharpd: Resource Management	Added support for enabling and disabling reproducibility on the job level.
sharp_am/sharpd: Subnet Management	Added support for controlling the SA key for SA operations.
libsharp_coll: GPUDirect	Added support for CUDA GPUDirect and GPUDirect RDMA.
Rev 1.8.1	
Aggregation Manager (sharp_am): Resiliency	Added support for waiting for jobs to end prior to performing fabric reinitialization on AM startup.
Mellanox SHARP Daemon (sharpd): Out-of-Box Improvements	Socket-based is now activated by default when installed from RPM/ MLNX_OFED.

19.1.2 Parameters Change History

Parameter	Component	Description
Rev 3.3.0		

dynamic_tree_algorithm	sharp_am	**New parameter**: Sets which algorithm should be used by the dynamic tree mechanism. This parameter is ignored when dynamic_tree_allocation is false. Possible values: 0 - SuperPOD oriented algorithm 1 - Quasi Fat Tree oriented algorithm **Default**: 0 – SuperPOD oriented algorithm
app_resources_default_limit	sharp_am	Sets the default max number of trees allowed to be used in parallel by a single app. Modified the possible range of values where the value of -1 means no resource limit, and 0 means no resources by default. **Default**: -1 – No resource limit
max_quota	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated. It is ignored and should not be used.
default_quota	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated. It is ignored and should not be used.
SHARP_SYSLOG_VERBOSITY	libsharp	**New parameter**: Sets the libsharp syslog verbosity level. Possible values: 0 – Disable syslog 1 – Errors log level 2 – Warnings log level 3 – Info log level **Default**: 1 – Errors log level
SHARP_GROUP_JOIN_MAD_TIMEOUT	libsharp	Sets the timeout till a retry for GroupJoin MAD, in milliseconds. Modified the default value. **Default**: 3000 milliseconds
SHARP_GROUP_JOIN_MAD_RETRIES	libsharp	Sets the number of retries for GroupJoin MAD. Modified the default value. **Default**: 5 retries
SHARP_QP_CONFIRM_MAD_TIMEOUT	libsharp	Sets the timeout till a retry for QP Allocation confirmation MAD, in milliseconds. Modified the default value. **Default**: 2000 milliseconds
Rev 3.2.0		
ignore_host_guids_file	sharp_am	**New parameter**: File with a list of Host GUIDs to be ignored for SHARP trees. **Default**: Null.
ignore_sm_guids	sharp_am	**New parameter**: A boolean parameter, telling whether SM GUIDs need to be ignored in SHARP trees parsed from SMDB file. **Default**: True.
ftree_ca_order_file	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated, it is ignored and should not be used.
enable_sat	sharp_am	**Deprecated parameter**: This parameter controlled whether SHARP should allow SAT jobs. The parameter is now marked as deprecatel. it is ignored and should not be used. SAT is always supported.
SHARP_COLL_SERIALIZE_MADS	libsharp	**New parameter**: Serialize sharp MADs in tree connect and group join operations, it is recommended to set this flag to true when running mpirun with multiple groups. **Default**: False.

SHARP_COLL_JOB_REQUEST_RMC	libsharp	**New parameter**: If set to True, require that any allocated SHARP trees will support the Reliable Multicast feature. **Default**: False.
SHARP_COLL_FORCE_BCAST_AS_ALLREDUCE	libsharp	**New parameter**: Force Bcast(rmc) as Allreduce operation **Default**: False.
Rev 3.1.1 LTS		
clean_and_exit	sharp_am	**New parameter:** A boolean parameter. When set to TRUE, sharp_am does not operate normally, but instead cleans SHARP resources from all switches and exits. **Default**: False - Operate normally.
Rev 3.1.0		
dynamic_tree_allocation	sharp_am	**New parameter:** A boolean parameter, tells whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Default**: False
max_trees_to_build	sharp_am	**Update**: In case dynamic_tree_allocation is set to True, this parameter will have no effect on the number of trees allocated; sharp_am would determine that value based on the amount of possible trees the switches can have. However, in the dynamic trees mode, this parameter affects the number of skeleton trees that sharp_am will use. It is recommended that the minimal value be the same as the number of root switches in the fabric. In case dynamic_tree_allocation is set to False, this parameter can be used to fulfil its purpose. **Default**:
SHARP_COLL_IB_TIMEOUT	libsharp	**New parameter:** Transport timeout on SHARP QP **Default**: 18
SHARP_COLL_IB_RETRY_COUNT	libsharp	**New parameter:** Transport retries on SHARP QP **Default**: 7
SHARP_COLL_IB_RNR_TIMER	libsharp	**New parameter:** RNR timeout on SHARP QP **Default**: 12
SHARP_COLL_IB_RNR_RETRY	libsharp	**New parameter:** RNR retries on SHARP QP **Default**: 7
SHARP_COLL_IB_SL	libsharp	**New parameter:** SL **Default**: 0
SHARP_COLL_ENABLE_MCAST_TARGET	libsharp	**Update**: Modified the default value from True to False. **Default**: False
Rev 3.0.0		
per_prio_default_quota	sharp_am	**Update**: This parameter controls only the default percentage provided to LLT jobs. Its default value is modified from 3 to 20
per_prio_default_sat_quota	sharp_am	**New parameter**: Default percentage of quota (OSTs, Buffers and Groups) per aggregation node per tree, to be requested for a single SAT job by its priority. If no explicit quota request is submitted, this parameter will set the quota percentage to be used. **Format**: prio_0_quota, [prio_1_quota, ..., prio_9_quota] Note that if only one value is set, it will be applied to all priorities. **Default**: 3

sat_jobs_default_absolute_osts	sharp_am	**New parameter**: Default number of OSTs to be allocated for SAT jobs per aggregation node per tree. Zero value means that no absolute value should be used, and the default percentage value is used instead. Note that the number of OSTs also affects the number of groups. **Default**: 0
app_resources_default_limit	sharp_am	**New parameter**: A numerical parameter, applicable only when reservation_mode is set to true. Sets the default max number of trees allowed to be used in parallel by a single app. This default value can be overridden per app upon reservation request. A value of 0 means no allowed resources, which means an app cannot execute any sharp job. **Default**: 1
force_app_id_match	sharp_am	**New parameter**: A boolean parameter, applicable only when reservation_mode is set to true. When set to true, an application ID must be provided upon job request, and it must match the application ID provided upon reservation request. Otherwise, the job will be denied. **Default**: False
SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE	libsharp	**Update**: Changed default value from 0 (no exclusive lock) to 2 (force exclusive lock)
Rev 2.7.0		
recovery_retry_interval	sharp_am	**New parameter:** A timeout in seconds for trees recovery retries. A value of 0 means do not try to recover trees. **Default**: 300
enable_seamless_restart	sharp_am	**New parameter:** A boolean flag. If enabled, AM tries to recover state from last AM run and continue the operation of the current jobs. **Default**: True
seamless_restart_trees_file	sharp_am	**New parameter:** Set the SHARP trees file used in Seamless restart. Need to mention only the file name, full path is constructed using 'dump_dir'. **Default**: sharp_am_trees_structure.dump
seamless_restart_max_retries	sharp_am	**New parameter:** Set the number of consecutive retries of seamless restart. If seamless restart fails more times in a row, it will be disabled in the next run. **Default**: 3
max_tree_radix	sharp_am	**Update**: Change default to 252
Ib_sat_max_mtu	sharp_am	**Update**: Change default to 5, to support MAD value that represents 4K MTU.
per_prio_default_quota	sharp_am	**Update**: Changed default to 3 instead of 20, enabling more SAT jobs to take place in parallel on each switch.
Rev 2.6.1		
dump_dir	sharp_am	**Update**: Changed default to /var/log
smx_enabled_protocols	sharp_am	**Update**: Changed default from 7 to 6 (disable UCX by default)
ib_mad_timeout	sharp_am	**Update**: Change deault from 200 to 500
dump_dir	sharp_am	**Update**: Change default to /var/log
sr_mad_timeout	sharpd	**New parameter:** Control timeout for ServiceRecord queries **Default**: 10000 millieconds

sr_mad_retries	sharpd	**New parameter:** Control number of retries for ServiceRecord queries **Default**: 3 retires
Rev 2.5.0		
smx_keepalive_interval	sharp_am/ sharpd	**New parameter:** Keep alive interval in seconds 0 to disable keep alive. **Default:** 60 seconds
smx_incoming_conn_keepalive_interval	sharp_am	**New parameter:** Keep alive interval for incoming connections **0 to disable** **Default: 300 seconds**
enable_exclusive_lock	sharp_am	**New parameter:** Enable/Disable exclusive lock feature. **Default:** True
enable_topology_api	sharp_am	**New parameter:** Enable/Disable Toplogy API feature **Default:** True
max_trees_to_build	sharp_am	**New parameter**: Control number of trees for AM to build **Default:** 126
Rev 2.4.3		
ib_max_mads_on_wire	sharp_am	**Modified behavior**: Changed default from 100 to 4096
ib_qpc_local_ack_timeout	sharp_am	**Modified behavior**: Changed default from 0x1F to 0x12
ib_sat_qpc_local_ack_timeout	sharp_am	**Modified behavior**: Changed default from 0x1F to 0x12
ib_qpc_timeout_retry_limit	sharp_am	**Modified behavior**: Changed default from 7 to 6
ib_sat_qpc_timeout_retry_limit	sharp_am	**Modified behavior**: Changed default from 7 to 6
Rev 2.0.0		
control_path_version	sharp_am	**New parameter** **Default**
max_compute_ports_per_agg_node	sharp_am	**Modified behavior**: When set to 0, AN radix is set to maximal radix value. **Default**: 0
default_reproducibility	sharp_am	**New parameter**: Control default reproducibility mode for jobs. **Default**: TURE
ib_sa_key	sharp_am	**New parameter**: Control SA key for SA operations. **Default**: 0x1
coll_job_quota_max_payload_per_ost	sharp_job_quota	**Modified behavior**: Change default value to 1024.
SHARP_COLL_MAX_PAYLOAD_SIZE	Libsharp_coll	**Removed**
SHARP_COLL_NUM_SHARP_COLL_REQ	Libsharp_coll	**Removed**

SHARP_COLL_ENABLE_REPRODUCIBLE_MODE	Libsharp_coll	**New parameter**: Control job reproducibility mode: 0 - Use default. 1 - No reproducibility. 2 - Reproducibility.
SHARP_COLL_ENABLE_CUDA	Libsharp_coll	**New parameter:** Enables CUDA GPU direct.
SHARP_COLL_ENABLE_GPU_DIRECT_RDMA	Libsharp_coll	**New parameter:** Enables GPU direct RDMA.
Rev 1.8.1		
pending_mode_timeout	sharp_am	**New parameter**: Defines AM waiting time for jobs to complete prior to fabric re-initialization upon startup.
job_info_polling_interval	sharp_am	**New parameter**: Defines job status polling interval when waiting for jobs to complete upon startup.

19.2 Bug Fixes History

The following table provides a list of bugs fixed in this SHARP version.

Internal Ref.	Issue
3400293	**Description:** Fixed an issue in libsharp where it failed to respond to messages from the SM while searching for Service Records, causing the SM to print timeout messages.
	Keywords: sharp_am; openSM
	Discovered in Version: 3.1.0
	Fixed in Release: 3.4.0
3479721	**Description:** Fixed the issue where sharp_am did not handle hypercube topologies well, causing it to incorrectly treat different switches as duplicates.
	Keywords: sharp_am; hypercube
	Discovered in Version: 3.3.0
	Fixed in Release: 3.4.0
3496440	**Description:** Fixed the issue in sharp_am where excessive log messages were printed for each disconnected or restarted compute host. Now, the information is printed in a consolidated manner in the form of summaries of disconnected hosts or a list of those hosts in a single log message. However, for more comprehensive details, the complete list of hosts is still available and printed at the DEBUG level.

Internal Ref.	Issue
	Keywords: sharp_am
	Discovered in Version: 3.3.0
	Fixed in Release: 3.4.0
3336788	**Description:** Fixed the issue in Firmware where MAD error responses might have been received in libsharp.
	Keywords: sharp_am; libsharp
	Discovered in Version: 3.2.0
	Fixed in Release: 3.3.0 (Quantum-2 Firmware 31.2010.6064)
3343503	**Description:** Fixed the issue where sharp_am installed from MLNX_OFED used an invalid range of job IDs, resulting in occasional errors when trying to establish new SHARP jobs.
	Keywords: MLNX_OFED; sharp_am
	Discovered in Version: 3.2.0
	Fixed in Release: 3.3.0
3368381	**Description:** Fixed the issue of when no sufficient amount of retries was made to resend failed libsharp GroupJoin MADs, SHARP jobs failed before they even started.
	Keywords: libsharp; MADs
	Discovered in Release: 3.0.0
	Fixed in Release: 3.3.0
3393902	**Description:** Fixed the issue where re-created virtual ports were not recognized by sharp_am, thus the correct tree was not built for them. This resulted in SAT jobs getting ibv_poll_cq failure in libsharp.
	Keywords: Virtual port; sharp_am; libsharp; SAT; ibv_poll_cq
	Discovered in Version: 3.2.0
	Fixed in Release: 3.3.0
3404474	**Description:** Fixed an issue where failure of application allocation of all hosts done via /app/sharp/resources REST-API returned a successful job instead of error.
	Keywords: REST API; allocation
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0

Internal Ref.	Issue
3406186	**Description:** Fixed an issue where SHARP AM failed handling reports from OpenSM if some switch ports were down or isolated.
	Keywords: Aggregation Manager; Aggregation Node; OpenSM
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0
3236363	**Description:** Fixed the way physical link failures between switches are handled. In the event of a link failure, a SHARP job utilizing the link has to be stopped; however, this will bear no effect on the other present or future jobs.
	Keywords: Aggregation Manager; sharp_am; Link Failure
	Discovered in Release: 3.1.0
	Fixed in Release: 3.2.0
3230585	**Description:** Fixed the issue of when operating in Dynamic trees mode, ibdiagnet may have printed warning messages about the existence of multiple distinct trees with the same tree ID.
	Keywords: Dynamic tree; ibdiagnet
	Discovered in Version: 3.1.0
	Fixed in Release: 3.2.0
3226743	**Description:** Fixed the issue of when a management host was not connected to a leaf switch, sharp_am might have printed a number of warning messages about trees that could not reach all aggregation nodes. As of SHARP v3.2.0, the active management host is automatically identified and is not treated as a potential compute host. However, please note that this does not include standby management hosts for which a warning message would still appear. These management hosts can be mentioned in a list of GUIDs to ignore via the parameter ignore_host_guids_file.
	Keywords: Aggregation Manager; sharp_am; leaf; GUID
	Discovered in Release: 3.0.1
	Fixed in Release: 3.2.0
3274564	**Description:** Fixed an issue where sharp_benchmark bash script failed to operate on all bash versions.

Internal Ref.	Issue
	Keywords: sharp_benchmark
	Discovered in Release: 3.1.1
	Fixed in Release: 3.2.0
3262936	**Description:** Fixed the issue where a crash took place during sharp_am reboot while physical links were hanging between switches in the fabric.
	Keywords: sharp_am; physical links; crash
	Discovered in Release: 3.1.0
	Fixed in Release: 3.1.1 LTS
3192770	**Description:** Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.
	Keywords: SR-IOV
	Discovered in Release: 3.0.0
	Fixed in Release: 3.1.0
3163697	**Description:** Fixed the issue of when the client application used more than 1024 file descriptors (range limit defined by FD_SETSIZE), libsharp was prevented from using any more file descriptors. Using poll() instead of select() enables using the full range of allowed file descriptors by Linux.
	Keywords: File descriptor; libsharp; HCOLL; HPC-X
	Discovered in Release: 3.0.0
	Fixed in Release: 3.1.0
3192770	**Description:** Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.
	Keywords: SR-IOV
	Discovered in Release: 3.0.0
	Fixed in Release: 3.0.1
3163697	**Description:** Fixed the issue of when the client application used more than 1024 file descriptors (range limit defined by FD_SETSIZE), libsharp was prevented from using any more file descriptors. Using poll() instead of select() enables using the full range of allowed file descriptors by Linux.
	Keywords: File descriptor; libsharp; HCOLL
	Discovered in Release: 3.0.0

Internal Ref.	Issue
	Fixed in Release: 3.0.1
2995739	**Description:** Sharp_am daemon is no longer removed when performing rpm upgrade and is overridden instead.
	Keywords: Aggregation Manager; rpm
	Discovered in Release: 2.6.1
	Fixed in Release: 2.7.0
2972970	**Description:** Fixed the issue where completion of SHARP installation using sharp_daemons_setup.sh script depended on python availability.
	Keywords: Aggregation Manager
	Discovered in Release: 2.6.1
	Fixed in Release: 2.7.0
2749073	**Description:** SHARP AM reports the rediscovery of aggregation nodes on every topology change.
	Keywords: Aggregation Manager
	Workaround: N/A
	Discovered in Release: 2.5.0
2736102	**Description:** SHARP AM and SHARPD overrides backlog files after restart when log rotation is enabled.
	Keywords: Aggregation Manager, SHARPD, log file
	Workaround: N/A
	Discovered in Release: 2.5.0
2700530	**Description:** Terminating a job process during job initialization before sending a job request to Aggregation Manager, might result in job resource leakage in the SHARP Aggregation Manager.
	Workaround: N/A
	Keywords: SHARPD, Aggregation Manager
	Discovered in Release: 2.5.0
2726821	**Description:** Terminating SHARPD while the job process is still running will result in job resource leakage in SHARP Aggregation Manager.
	Workaround: Terminate SHARPD after terminating the job processes.
	Keywords: SHARPD, Aggregation Manager

Internal Ref.	Issue
2795902	**Description:** SHARPD might allocate handlers on GPU when running with UCX.
	Keywords: SHARPD, SMX, UCX
	Workaround: N/A
	Discovered in Release: 2.5.0
	Workaround: Disable UCX
2770210	**Description:** Syslog verbosity depends on log file verbosity.
	Keywords: SHARPD, Aggregation Manager
	Discovered in Release: 2.5.0
	Workaround: None
2825519	**Description:** Aggregation Manager continue to run after SM failover.
	Keywords: Aggregation Manager
	Discovered in Release: 2.5.0
	Workaround: Stop AM daemon manually
2754175	**Description:** SHARP Aggregation Manger might allocate bad links for jobs after receiving timeouts from Aggregation Nodes.
	Workaround: Restart corresponding switch or restart SHARP Aggregation Manager.
	Keywords: Aggregation Manager
	Discovered in Release: 2.5.0
2796317	**Description:** SHARP jobs may hang when running in reservations mode (i.e. SHARP allocation is enabled), and reservation is created with limited PKEY, and configuring reservation PKEY on tree is enabled.
	Workaround: The PKEY used for creating the reservation should be "full" (the most significant bit should be on e.g. 0x805c instead of 0x5a).
	Keywords: Aggregation Manager, Reservations, PKEY, UFM
	Discovered in Release: 2.5.0

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of NVIDIA Corporation and/or