Hewlett Packard
Enterprise

HPE Cray System Management Diagnostics Guide (1.2) (S-8038)

Part Number: S-8038
Published: July 2022

HPE Cray System Management Diagnostics Guide

Contents

Copyright and Version

HPE Cray System Management Diagnostics Guide

4 Overview

CSM Diagnostics (CSM Diags) provides a set of diagnostic tools to perform various node level and cluster-wide tests on various cluster components (on compute nodes) such as CPU, MEMORY, DISK, GPU, INTERCONNECT, and etc.

CSM Diags provides both functional and performance test suites and also includes both MPI and non MPI test suites. CSM Diags contains a reporting mechanism that provides clear and key insights about a cluster's health.

5 Install CSM Diags

5.1 Prerequisites

The following products or components must be installed and verified:

- Cray System Management (CSM)

- Cray Programming Environment (CPE)
- Workload Manager is installed and verified and also it includes the following Helm Charts:
  - cray-smd
  - Slurm or PBS
  - Cray System Management (CSM)
  - Cray Programming Environment (CPE)
- Shared filesystem between the worker and compute nodes

To run CSM Diags from any node, ensure that Cray CLI is initialized using the following command:

```
ncn-w#cray init
hostname: https://api-gw-service-nmn.local/
user: vers
password: ********
```

Verify the installation of these components using the following script:

```
ncn-w# ./pre-install-checks.py
```

Note: Using the command `ncn-w# tar -xzvf csm-diags-*.tar` untar the CSM Diag tarball to locate the `pre-install-checks.py` script file.

This script is used to verify if the system is meeting all prerequisites to run `csm-diags`.

`pre-install-checks` script verifies if the `shared fs` is in a location. Verifies if PE and WLM (slurm/pbs) are available in a node.

To understand how to run this script, use the `-h` option along with the script.

```
ncn-w# ./pre-install-checks.py -h
```

To verfiy the `lustre` or `shared fs` on a path, use the following script:

```
ncn-w# ./pre-install-checks.py -l path_to_lustre -n xname_to_check_pe -w xname_to_check_wlm
```

```
ncn-m001:~ # kubectl get pods -A | grep "cray-smd"
services            cray-smd-7bd97f78f5-4dczq                                 2/2     Running            0          12d
services            cray-smd-7bd97f78f5-9jl58                                 2/2     Running            1          59d
services            cray-smd-7bd97f78f5-xgq9g                                 2/2     Running            1          59d
services            cray-smd-init-2rb7s                                       0/2     Completed          0          55d
services            cray-smd-postgres-0                                       3/3     Running            0          59d
services            cray-smd-postgres-1                                       3/3     Running            0          17d
services            cray-smd-postgres-2                                       3/3     Running            0          12d
services            cray-smd-wait-for-postgres-15-bd7wj                       0/3     Completed          0          55d
ncn-m001:~ # kubectl get pods -A | grep "slurm"
services            slurm-config-7-x8gl8                                      0/2     Completed          0          28d
user                slurmctld-5ddd9b57d7-kx4b9                                4/4     Running            5          18d
user                slurmdb-6f78686cb9-s5rzd                                  1/1     Running            0          19d
user                slurmdbd-5987db57df-th9kt                                 3/3     Running            0          12d
ncn-m001:~ #
```

Figure 1: Prerequistes

```
ncn-m001:~ # kubectl get pods -A | grep "pbs"
user                pbs-79898b86c6-cvsm9                                      1/1     Running            0          42d
user                pbs-comm-6c44d4db84-2zppz                                 1/1     Running            0          48d
ncn-m001:~ #
```

Figure 2: PBSpods

5.2 Installing CSM Diags

The `install.sh` script is used to perform the installation of CSM Diags. The script detects the Workload Manager present on the cluster and install CSM Diags pods to use it. Perform the following steps to install CSM Diags:

1. Start a typescript to capture the commands and output from this installation.

   ```
   ncn-m001# script -af product-csm-diags.$(date +%Y-%m-%d).txt
   ncn-m001# export PS1='\u@\H \D{%Y-%m-%d} \t \w # '
   ```

2. Move the CSM Diags tarball to a worker node and login to the same worker node.
3. Untar the CSM Diags tarball using the following command:

   ```
   ncn-w# tar -xzvf csm-diags-*.tar
   ```

4. To install CSM Diags, run the script `install.sh` and make sure that the shared filesystem is present in the `/lus` path.

   ```
   ncn-w# ./install.sh
   ```

   Note: If the shared filesystem is not present in the `/lus` path, the administrator must run the following command to install CSM Diags on a custom mount point:

   ```
   ncn-w# ./install.sh -f /path/to/shared/fs
   ```

5. Finish the typescript file started at the beginning of this installation.

   ```
   ncn-m001# exit
   ```

   Note: If the script fails to identify the Workload Manager (WLM) installed on the system, the installation is aborted. The administrator can proceed with the installation by providing WLM to use as an option to `install.sh` script.

   For using SLURM:

   ```
   ncn-m001# ./install.sh -w SLURM
   ```

   For using PBS:

   ```
   ncn-m001# ./install.sh -w PBS
   ```

5.3 Verifying CSM Diags Installation

To verify the CSM Diags installation, use the following `post-install-check.py` script:

```
ncn-w# ./post-install-check.py
```

The script verifies the state of the HSM, WLM (slurm/pbs), PE, the health of all badger related Kubernetes pods, Lustre or shared filesystem, installation of the CSM Diags RPM are in the location `/opt/cray/csm-diags`.

To verify WLM or PE, you can specify `xname` in the script:

```
ncn-w# ./post-install-check.py -n PENODE
ncn-w# ./post-install-check.py -w WLM_on_node
```

5.4 Upgrade

Note: The following steps must be followed in a system where CSM Diags is installed, and you are upgrading the CSM Diags solution. To verify if the CSM Diags is installed, see the Verifying CSM DIAGS Installation.

The `upgrade.sh` script is used to perform the upgrade of CSM Diags.

1. Start a typescript to capture the commands and output from this upgrade.

   ```
   ncn-m001# script -af product-csm-diags.$(date +%Y-%m-%d).txt
   ncn-m001# export PS1='\u@\H \D{%Y-%m-%d} \t \w # '
   ```

2. Take the CSM Diags tarball to a worker node and untar using the following command:

   ```
   ncn-w# tar -xzvf csm-diags-*.tar
   ```

3. After untar, locate and run the upgrade script to upgrade CSM Diags product if shared filesystem is present in the `/lus` path. This script automatically acquires the previous configuration of the shared filesystem.

   ```
   ncn-w# ./upgrade.sh
   ```

4. To verify if upgrade is successful, see Verifying CSM DIAGS Installation.
5. Finish the typescript file started at the beginning of this upgrade.

   ```
   ncn-m001# exit
   ```

5.5 CSM Diags version

Run the following command to obtain the version information related to different diagnostics such as Badger, CSM Diags CLI, and various binaries used in CSM Diags:

```
#./sdiag_run.py -r
```

6 Prerequisite - perform sanity checks prior to running CSM Diags

Run the following sanity checks prior to running CSM Diags from the worker node from where the `install.sh` is run.

- (For slurm WLM) Ensure to set `AllowedRAMSpace=100` in `/etc/slurm/cgroup.conf` (on compute nodes) before running sanity checks and CSM Diags.

6.1 Consistency checks

Post-installation of CSM Diags, it is recommended to perform the following sanity checks using the worker node where automation framework RPM is installed:

Note: For **compute nodes**, it is recommended to perform the following sanity checks from where the `pbsframework_install.sh script` or `slurm_framework_install.sh` is run.

1. On the worker node:

   Copy `/opt/cray/csm-diags/examples/sdiag-list.json.consistency-checks` to `/opt/cray/csm-diags/sdiag-list.json`

2. Create a `nodes` file (which is available at `/opt/cray/csm-diags`) populated with all the `xnames` of the compute nodes of the cluster.

   Non compute nodes

   Use the following command to retrieve `xnames`:

   ```
   cray hsm state components list --role compute --state Ready --format json | jq -c  '.Components[]
   | { ID: .ID, NID: .NID, Class: .Class, State: .State}'
   ```

   Compute nodes (PBS or SLURM)

   Use the following command to retrieve `nid` on PBS:

   ```
   pbsnodes -aS
   ```

   Use the following command to retrieve `nid` on SLURM:

   ```
   sinfo
   ```

3. Run the consistency checks using the following command:

   Non compute nodes

   ```
   ncn-w# ./sdiag_run.py #(available at /opt/cray/csm-diags in the worker node)
   ```

   Compute nodes (PBS or SLURM)

   ```
   nid# ./sdiag_run.py #(available at /opt/cray/csm-diags in the compute node where install took
   place)
   ```

4. Ensure that the consistency checks run and progress to a ***PASSED*** state.

This certifies that the CSM Diags is successfully installed on this cluster and also that the consistency checks are working. The other diagnostics can be run on this system now.

If any of the tests are in a ***FAILED*** state, either the CSM Diags installation might have to be reverified, the particular consistency check has reported an inconsistency, or an error on the system.

6.2 Non MPI Diagnostics

Post installation of CSM Diags, it is recommended to perform the following sanity checks using the worker node where automation framework RPM is installed:

1. Copy `/opt/cray/csm-diags/examples/sdiag-list.json.non-mpi-diags` to `/opt/cray/csm-diags/sdiag-list.json` and remove all other tests except `olcmt` (that is found on the node where CSM Diags product is installed). The contents of the `sdiag-list.json` file must be:

   ```
   {
   "test_names": [
   "olcmt"
   ]
   }
   ```

2. Create a `nodes` file (available at `/opt/cray/csm-diags`) populated with all the `xnames` or `nids` of compute nodes of the cluster.

   Non compute nodes

   Use the following command to retrieve `xnames`:

   ```
   cray hsm state components list --role compute --state Ready --format json | jq -c  '.Components[]
   | { ID: .ID, NID: .NID, Class: .Class, State: .State}'
   ```

   Compute nodes (PBS or SLURM)

   Use the following command to retrieve `nid` on PBS:

   ```
   pbsnodes -aS
   ```

   Use the following command to retrieve `nid` on SLURM:

   ```
   sinfo
   ```

3. Run the diagnostic using the following command:

   Non compute nodes

   ```
   ncn-w# ./sdiag_run.py #(available at /opt/cray/csm-diags in the worker node where install took
   place).
   ```

   Compute nodes (PBS or SLURM)

   ```
   nid# ./sdiag_run.py #(available at /opt/cray/csm-diags in the compute node where install took
   place).
   ```

4. Ensure that the test runs and results in a “PASSED” state.

This certifies that the CSM Diags are successfully installed on this cluster and also that the non MPI diagnostics are working as expected.

If the test is in a “FAILED” state, it means CSM Diags installation might have to be reverified or the non MPI Diagnostic, `olcmt` is not running correctly.

```
ncn-m001:~ # cray badger sessions describe "cb6c5446-41e5-44ad-bdac-8243b40aa067"
id = "cb6c5446-41e5-44ad-bdac-8243b40aa067"
finishTimestamp = "2021-08-25T12:52:49.043431Z"
startProcessTimestamp = "2021-08-25T12:03:05Z"
doNotStartProcessAfterTimestamp = ""
dropDeadTimestamp = ""
loopSuiteUntilTimestamp = ""
suiteID = "02b71857-4ba2-450e-bd7a-ff955a4b5743"
suiteName = "olcmt-suite"
workingDirectory = "olcmt"
sessionDirectory = "/lus/cls01053/csm-diags/olcmt/olcmt-suite/2021-08-25T12:03:09Z"
cleanup = false
cleaned = false
analysisStatus = "PASSED"
```

6.3 MPI Diagnostics

To verify if MPI is functioning:

1. Copy `/opt/cray/csm-diags/examples/sdiag-list.json.mpi-diags` to `/opt/cray/csm-diags/sdiag-list.json` and remove all other tests except `rank` (that is found on the node where CSM Diags product is installed). The contents of the `sdiag-list.json` file must be:

   ```
   {
   "test_names": [
   "rank"
   ]
   }
   ```

2. Edit the `nodes` file (available at `/opt/cray/csm-diags`) with more than three `xnames` or `nids` of compute nodes of the cluster.

   Non compute nodes

   Use the following command to retrieve `xnames`:

   ```
   cray hsm state components list --role compute --state Ready --format json | jq -c  '.Components[]
   | { ID: .ID, NID: .NID, Class: .Class, State: .State}'
   ```

   Compute nodes (PBS or SLURM)

   Use the following command to retrieve `nid` on PBS:

   ```
   pbsnodes -aS
   ```

   Use the following command to retrieve `nid` on SLURM:

   ```
   sinfo
   ```

3. Run the diagnostic using the following command:

   Non compute nodes

   ```
   ncn-w# ./sdiag_run.py #(available at /opt/cray/csm-diags in the worker node where install took
   place).
   ```

   Compute nodes (PBS or SLURM)

   ```
   nid# ./sdiag_run.py #(available at /opt/cray/csm-diags in the compute node where install took
   place).
   ```

4. Ensure that the test runs and results in a “PASSED” state.
5. This certifies that the CSM diags are successfully installed on the cluster and the MPI Diagnostics are working as expected.
6. If the test enters the failed state, verify if the failure of this test is due to the MPI based errors or HSN based errors by performing the following steps:

   1. Edit the `sdiag-list.json` file in `/opt/cray/csm-diags` (found on the node where CSM Diags product is installed) directory to just have “rank”.
   2. Edit the nodes file (available at `/opt/cray/csm-diags`) with only one xname of compute nodes of the cluster.

      Command to fetch xnames:

      ```
      cray hsm state components list --role compute --state Ready --format json | jq -c  '.Components[]
      | { ID: .ID, NID: .NID, Class: .Class, State: .State}' )
      ```

   3. Run the diagnostic using

      ```
      ncn-w# ./sdiag_run.py
      #(present in location /opt/cray/csm-diags in the worker node where install took place).
      ```

   4. Ensure that the test runs and results in a “PASSED” state.
   5. This certifies that the cluster has CSM Diags successfully installed and MPI is healthy. The HSN might be unhealthy.

7 How to run CSM Diags on non-compute nodes

7.0.1 Automation framework

An automation script (`sdiag_run.py`, which uses CSM Diags framework) is provided to execute multiple diagnostics (MPI, NON_MPI, GPU, and Slingshot) simultaneously on multiple compute nodes.

Note: Before executing the following steps, ensure sanity checks are complete. See the section, Prerequiste - Perform Sanity checks prior to running CSM Diags.

The automation script can be found at `/opt/cray/csm-diags/sdiag_run.py` in the node where the installation was performed:

1. The administrator must modify the `sdiag-list.json` configuration file with the list of diagnostics that must be executed.

- **`nodes`** Node names on which the diagnostics must execute. By default, populated with a dummy value. The administrator has to enter the appropriate node names.
- **`nodes_gpu`** GPU node names on which the GPU diagnostics must be executed for Nvidia or AMD. By default, the node names are populated with a dummy value. The administrator must enter appropriate node names. For example, `nvidia gpu` for running Nvidia GPU diagnostics `gpuburn` and `xkchecks`. `amd gpu` for running AMD GPU diagnostics `amdgpu` checks.
- **`sdiag-list.json`** List of diagnostics that needs to be executed. The administrator can add or remove multiple diagnostics from the JSON file. By default, this list is shipped as empty. This list can be populated using the examples available in `/opt/cray/csm-diags`.
- **`sdiag-arguments.json`** Argument values for each diagnostic test. The administrator can change the arguments for a particular diagnostics. By default, diagnostics is run with the default arguments.
- **`sdiag-gumball.json`** This file contains information like `gumball-name`, `gumball-location`, `application-name`, `application_description`, `suite_name`, `suite_description`, and `session_directory`. The administrator can modify these values.

2. Add or remove the tests from the test list in `sdiag-list.json`.

   Sample `sdiag-list.json`:

```
{
     "test_names": [
     "oldisk",
     "rank"
 ],

 "osu_benchmark": {
     "one-sided":
     [
                "osu_get_bw",
                "osu_put_bw",
                "osu_get_latency"

     ],
     "pt2pt":
             [
                "osu_bw",
                "osu_latency"
             ]
 }
}
```

3. Modify the input parameters if needed in the `sdiag-arguments.json`.

   Sample `sdiag-arguments.json`:

```
   {
      "test_name":"cwhpcc",
      "arg":"",
```

```
    "invocation_string":"--number-of-tasks-per-node=@auto",
    "binary":"",
    "nodes":"nodes"
}
```

4. Launch the diagnostics using the following command:

   ```
   ncn-w# ./sdiag_run.py
   ```

   Output of the executed command:

   ```
   out_22:40:03.txt output file has been created in /var/log/cray/shasta-diag

   ============================== Logs ================================
   Execution completed
   linpack: cray badger sessions describe 'a882b69a-acfb-43c4-9a77-8a9594c8aae2'
   LOGDIR : "/lus/linpack/linpack-suite/2021-09-02T22:40:09Z"

   stream: cray badger sessions describe '80e6fb20-1b58-4c6a-9452-252442182585'
   LOGDIR : "/lus/stream/stream-suite/2021-09-02T22:40:14Z"

   olconf: cray badger sessions describe 'e64394ef-0966-4f12-8f5d-3c7a221df37f'
   LOGDIR : "/lus/olconf/olconf-suite/2021-09-02T22:40:18Z"
   ```

5. Run the following command to view the run status:

   ```
   ncn# cray badger sessions describe "session-id"
   ```

6. Execute the following command to view the `olconf` test status for the preceding run:

   ```
   ncn# cray badger sessions describe 'e64394ef-0966-4f12-8f5d-3c7a221df37f' #can be run in master or wor
   ```

   Sample Status Output:

   ```
   ncn-w001:/opt/cray/shasta-diags # cray badger sessions describe 'e64394ef-0966-4f12-8f5d-3c7a221df37f'
   id = "e64394ef-0966-4f12-8f5d-3c7a221df37f"
   finishTimestamp = "2021-09-02T22:42:12.875378Z"
   startProcessTimestamp = "2021-09-02T22:40:15Z"
   doNotStartProcessAfterTimestamp = ""
   dropDeadTimestamp = ""
   loopSuiteUntilTimestamp = ""
   suiteID = "710bdb37-84bc-4b90-a36a-ea612e8eda38"
   suiteName = "olconf-suite"
   workingDirectory = "olconf"
   sessionDirectory = "/lus/olconf/olconf-suite/2021-09-02T22:40:18Z"
   cleanup = false
   cleaned = false
   analysisStatus = "PASSED"
   underUtilizedNodes = []
   notFound = []
   emptyNodes = []
   ```

   Time stamped folders with log files are created for each of the diags that are executed.

   For the preceding run, `olconf` logs can be found at `/lus/olconf/olconf-suite/2021-09-02T22:40:18Z`.

   Analyze Output:

   Sample output for olconf:

```
ncn-w001:/lus/olconf/olconf-suite/2021-09-02T22:40:18Z/0/olconf-application/step/25 # cat 25.analyze.manifest.out
Olconf:


Description:


Olconf is an application that is intended as a system wide confidence test which will run on a Single System Image.
This test will test the sanity of cpu, memory and intercluster connectivity


--------------------------------------------------------------------------
Test Summary:


Olconf Completed Successfully on All Nodes


--------------------------------------------------------------------------
Detailed Report:


Hostname:  nid000001

Olconf PASSED

-------------------------------------------------
Hostname:  nid000002

Olconf PASSED

-------------------------------------------------
Hostname:  nid000003

Olconf PASSED

-------------------------------------------------
Hostname:  nid000004

Olconf PASSED

-------------------------------------------------
```

8 How to run CSM Diags on compute nodes using PBS or SLURM

Note: The following procedure is an alternate approach for users to launch CSM Diags directly from compute nodes using PBS or SLURM. This approach removes the dependency of the `lustre fs` or `shared fs` on the worker nodes. This procedure includes an `rpm` that can be installed on multiple compute nodes for launching CSM Diags using the `PBS wlm` or `SLURM wlm`and uses the `lustre fs` or `shared fs` present on the compute nodes for logging and launching jobs.

Prerequisites

1. PBS or SLURM is installed and verified.
2. Shared file system across compute nodes.
3. Cray Programming Environment (CPE) on all compute nodes.

Installing CSM Diags on compute nodes for PBS framework

The following PBS framework script is used to install CSM Diags on compute nodes:

To install `rpm` on specific compute nodes, run the following script along with the `xnames` of the compute nodes:

```
ncn-w# ./pbsframework_install.sh -n <xname1,xname2,..>
```

Note: If the compute node names are not provided, the script selects a compute node and installs it on that node.

Installing CSM Diags on compute nodes for SLURM framework

The following SLURM framework script is used to install CSM Diags on compute nodes:

To install `rpm` on specific compute nodes, run the following script along with the `xnames` of the compute nodes:

```
ncn-w# ./slurm_framework_install.sh -n <xname1,xname2,..>
```

Note: If the compute node names are not provided, the script selects a compute node and installs it on that node.

Running CSM Diags on compute nodes (PBS or SLURM)

Before executing the following steps, ensure sanity checks are complete. See the section, Prerequiste - Perform Sanity checks prior to running CSM Diags.

The CSM Diags CLI is available at `/opt/cray/csm-diags/sdiag_run.py` in the compute node where the installation was performed:

1. The administrator must modify the `sdiag-list.json` configuration file with the list of diagnostics that must be executed.

- **`nodes`** Node names on which the diagnostics must execute. By default, populated with a dummy value. The administrator has to enter the appropriate node names.
- **`nodes_gpu`** GPU node names on which the GPU diagnostics must be executed for Nvidia or AMD. By default, the node names are populated with a dummy value. The administrator must enter appropriate node names. For example, `nvidia gpu` for running Nvidia GPU diagnostics `gpuburn` and `xkchecks`. `amd gpu` for running AMD GPU diagnostics `amdgpu` checks.
- **`sdiag-list.json`** List of diagnostics that needs to be executed. The administrator can add or remove multiple diagnostics from the JSON file. By default, this list is shipped as empty. This list can be populated using the examples available in `/opt/cray/csm-diags`.
- **`sdiag-arguments.json`** Argument values for each diagnostic test. The administrator can change the arguments for a particular diagnostics. By default, diagnostics is run with the default arguments.
- **`sdiag-gumball.json`** This file contains information like `gumball-name`, `gumball-location`, `application-name`, `application_description`, `suite_name`, `suite_description`, and `session_directory`. The administrator can modify these values.

2. Add or remove the tests from the test list in `sdiag-list.json`.

   Sample `sdiag-list.json`:

```
  {
     "test_names": [
     "oldisk",
     "rank"
 ],

 "osu_benchmark": {
     "one-sided":
     [
                "osu_get_bw",
                "osu_put_bw",
                "osu_get_latency"

     ],
     "pt2pt":
             [
                "osu_bw",
                "osu_latency"
             ]
    }
  }
```

3. Modify the input parameters if needed in the `sdiag-arguments.json`.

   Sample `sdiag-arguments.json`:

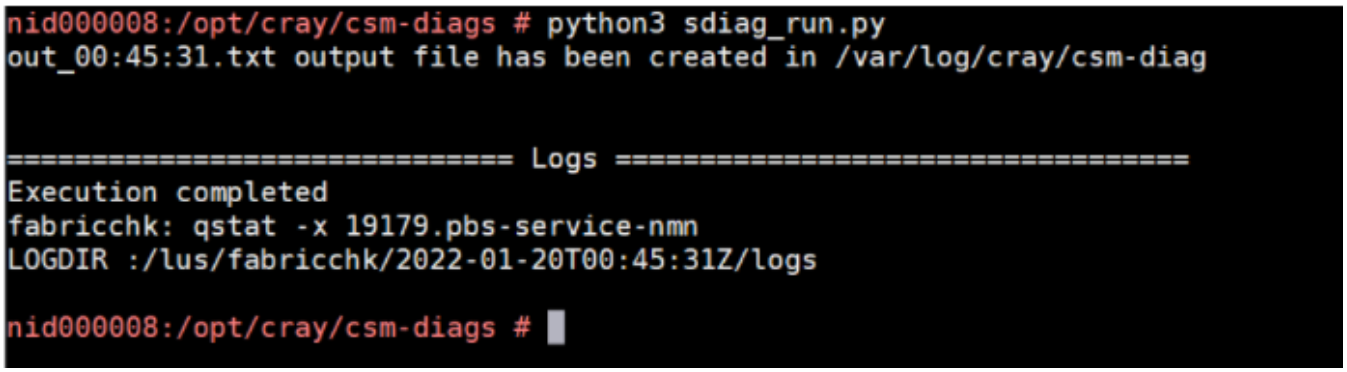
```
   {
      "test_name":"cwhpcc",
      "arg":"",
      "invocation_string":"--number-of-tasks-per-node=@auto",
      "binary":"",
      "nodes":"nodes"
```

```
}
```

4. Launch the diagnostics using the following command:

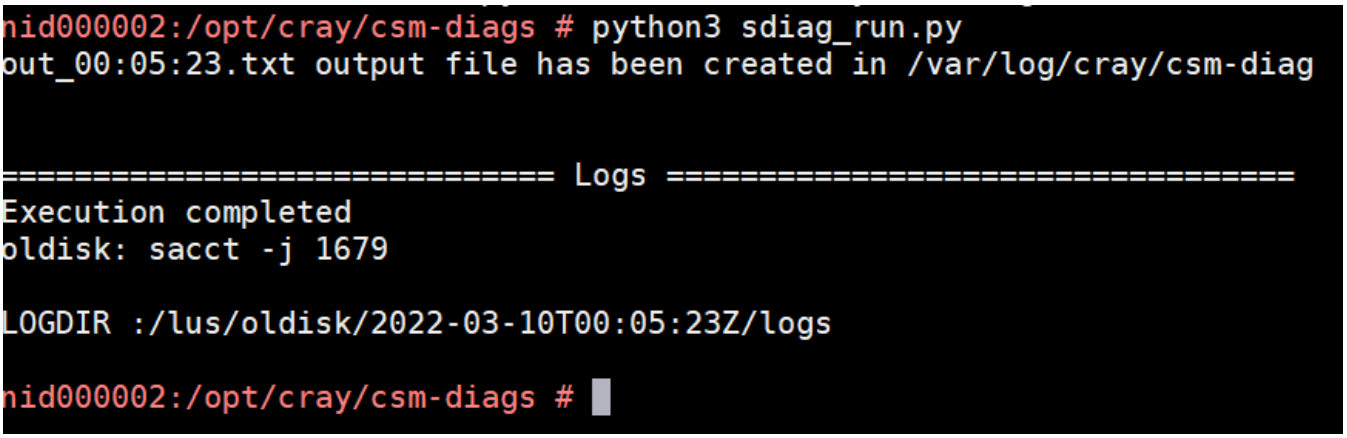
   ```
   nid# ./sdiag_run.py
   ```

   Output of the executed command on PBS framework:



   The `qstat -x 19179.pbs-service-nmn` can be used for monitoring the submitted job. The LOGDIR can be used for further analysis of the executed diagnostics.

   Output of the executed command on SLURM framework:



   The `sacct -j 1679 (job_id)` can be used for monitoring the submitted job. The LOGDIR can be used for further analysis of the executed diagnostics.

9 Adding a custom diagnostics using badger

Use badger to launch a custom diagnostic on any compute nodes using the WLM and log it using the mounted shared filesystem. By default, `badger_run.sh` and `badger_analyze.sh` scripts are required in the badger gumball to run the custom scripts.

`badger_run.sh`

Badger launches this script using the WLM.

Example script: `#!/bin/bash  echo Hostname: $(hostname)  ./custom_script  exit $?`

`badger_analyze.sh`

Badger launches this script to determine the exit status of the test whether it is a success or not. This script can also be used to run the post-run diagnosis to go through the generated log files and derive a key diagnosis.

Example script:

```
#!/usr/bin/env bash
echo "This is analysis stage"
FILE_MANIFEST="$1"
```

```
if ! [[ -s $FILE_MANIFEST ]]; then
  echo "${GUMBALL_NAME} has FAILED: Empty manifest file"
  exit 1
fi

 STDOUT_FILES=( $(cat $FILE_MANIFEST | grep "\.out\$") )
 STDERR_FILES=( $(cat $FILE_MANIFEST | grep "\.err\$") )

 analysisSuccess=true

 for ((i=0; i<${\#STDOUT_FILES[*]}; i++)); do
  curRunStdoutFile=${STDOUT_FILES[$i]}
  curRunName=$(basename $curRunStdoutFile .out)

 if grep -q -I "PASS" $curRunStdoutFile; then
   echo "${curRunName}: TEST has PASSED"
 else
   curRunStderrFile=$\{STDERR_FILES[$i]}
   analysisSuccess=false

  echo "${curRunName}: TEST has FAILED"
   if ! [[ -s $curRunStderrFile ]]; then
    echo ".err file is not available"
    else
    cat $curRunStderrFile
   fi
fi
done

  if [[ ${analysisSuccess} == false ]]; then
    exit 1
    fi

exit 0
```

To launch the diagnostics, use the following steps:

1. Create the gumball using the following command:

   ```
   tar -cvf tarball.tar badger_run.sh badger_analyze.sh file1 file2
   comment: tar your badger_run.sh and badger_analyze.sh (these are the minimum)
   ```

2. Upload the gumball as an artifact so that the badger can access the tarball using the following command:

   ```
   cray artifacts create badger tarball.tar /location/of/the/tarball.tar
   ```

3. Check the gumball availability using the following command:

   ```
   cray artifacts list badger
   ```

4. Create a custom application from the gumball using the following badger command:

   ```
   cray badger applications create --name="name-of-the-application" --description="give your description" \
   --link-to-exe="tarball.tar" --invocation-per-node="True" --output-file-per-node="False" \
   --group-by-core-count="False" --group-by-memory-size="False" \
   #: (" --invocation-per-node="True" if it's a non-mpi diag, \
   " --invocation-per-node="False" if it's a mpi diag)
   ```

5. Check the application created using the following badger command:

   ```
   cray badger applications describe "application-id"
   ```

6. Create a custom suite from the gumball using the following badger command:

```
cray badger suites create --name="name-of-your-suite" \
--description="description-of-your-suite" --application-order-application-id="application-id"
```

7. Check the suite created using the following badger command:

   ```
   cray badger suites describe "suite-id" --format=json
   ```

8. Create a custom session from the gumball using the following badger command:

   ```
   cray badger sessions create --suite-id="suite-id" --start-process-timestamp="sampletime" \
   --working-directory="directory"
   #: time should be in format "2020-07-01T05:52:04.910822Z" - date +"%Y-%m-%dT%TZ"
   ```

9. Verify the session created using the following badger command:

   ```
   cray badger sessions list
   ```

10. Monitor the session using the following badger command:

    ```
    cray badger sessions describe "session-id"
    ```

For more customizations and options available to customize your application, suite, or session, see the ***Badger user guide*** or badger help commands in the Cray CLI.

10 CSM Diags uninstall

The `uninstall.sh` script is used for uninstalling CSM Diags. This script is present in the tarball from where the CSM Diags product was installed (worker node).

The following command uninstalls CSM Diags:

```
ncn-w# ./uninstall.sh
```

Sample output:

```
Beginning csm-diags(Badger) Uninstall...
release "cray-hms-badger-db-util" uninstalled
release "cray-hms-badger-api" uninstalled
release "cray-hms-badger-job-api" uninstalled
release "cray-hms-badger-service" uninstalled
release "cray-hms-badger-loader" uninstalled
release "cray-fox" uninstalled
done
```

To uninstall the Automation framework use the following command:

```
ncn-w# rpm -e $(rpm -qa| grep -i csm-diags-automation)
```

11 Diagnostics

CSM Diags provides the following levels of diagnostics:

- Consistency checks
- System level diagnostics
- Nvidia diagnostics
- AMD GPU diagnostics
- Fabric diagnostics
- OSU benchmark

11.1 Consistency Checks

Note: Click the individual CSM Diags name to find more information.

Diagnostic name	CSM Diags name
cpuchk	sdiag_nhc_gumball
memchk	sdiag_nhc_gumball
fabricchk	sdiag_nhc_gumball
netchk	sdiag_nhc_gumball
fschk	sdiag_nhc_gumball
mpichk	sdiag_mpichk_gumball

11.1.1 cpuchk

Syntax

```
sdiag_nhc_gumball
```

Description

Checks for the uniformity in the CPU configuration on selected nodes by verifiying CPU model, vendor, architecture, number of online or offline cores, number of threads and sockets on each node, and reports inconsistencies. Also checks the CPU operating frequency on the nodes and reports outlier nodes based on CPU operating frequency. An outlier is defined by `median +/- 3 * median absolute deviations`.

Arguments

`-c consistency-configs` Name of the directory to be created within the lustre mount path where the configuration files for this check are stored.

11.1.2 memchk

Syntax

```
sdiag_nhc_gumball
```

Description

Checks for the uniformity in the memory configuration on selected nodes by verifiying physical memory size, swap space, DIMM size, speed, DIMMS per socket and the balanced memory configuration on the nodes, and reports inconsistencies.

Arguments

`-c consistency-configs` Name of the directory to be created within the lustre mount path where the configuration files for this check are stored.

11.1.3 fabricchk

Syntax

```
sdiag_nhc_gumball
```

Description

Checks for uniformity in the slingshot fabric configuration on selected nodes by verifiying HSN device states, HSN device checks, PCI link status and speed, NUMA, NIC version, and reports inconsistencies. It also checks for any errors in IP and MAC address configuration, DNS configuration, PCIe errors and lmon errors, and reports inconsistencies. It also performs ping check to the specified node through the specified HSN device.

Arguments

`-c consistency-configs` Name of the directory to be created within the lustre mount path where the configuration files for this check are stored.

`-i <HSN Device>` Name of the HSN device on the compute node on which `fabrichchk` runs, through which the compute node specified with `-h` is pinged. The default HSN device is hsn0.

`-h <compute node>` Name of the compute node which is pinged. The default compute node is the first compute node listed by the workload manager.

11.1.4 netchk

Syntax

```
sdiag_nhc_gumball
```

Description

Checks for the uniformity in the ethernet configuration on selected nodes by verifying driver name, version, firmware version, ethernet PCI value, duplex value, advertised link speed and auto-negotiation value on the nodes, and reports inconsistencies.

Arguments

`-c consistency-configs` Name of the directory to be created within the lustre mount path where the configuration files for this check are stored.

11.1.5 fschk

Syntax

```
sdiag_nhc_gumball
```

Description

Checks for the lustre and nfs mount point on selected nodes. Also checks for file system size and usage, and reports if the file system usage exceeds a certain threshold.

Arguments

`-c consistency-configs` Name of the directory to be created within the lustre mount path where the configuration files for this check are stored.
`-t <threshold value>` Threshold value in percentage for file system usage check. The default value is 90.
`-f <lustre path>` Path at which `fschk` checks if lustre is mounted. This is an optional argument. The default path to perform checks is the lustre mount point on the system.
`-n <nfs path>` Path used by `fschk` checks if nfs is mounted. This is an optional argument. Mount point check for nfs is not performed unless this parameter is provided.

11.1.6 mpichk

Syntax

```
sdiag_mpichk_gumball
```

Description

Runs the mpi “hello world!” program to determine the mpi status of the system.

Arguments

`-rankspernode 10` Number of ranks to be created and used for mpichk on each compute node. The default value is 10.

11.2 System level diagnostics

Note: Click the individual CSM Diags name to find more information.

Diagnostic name	CSM Diags name
linpack	sdiag_linpack_gumball
cwlinpack	sdiag_cwlinpack_gumball
nodeperf	sdiag_nodeperf_gumball
stream	sdiag_stream_gumball
olcmt	sdiag_olcmt_gumball
oldisk	sdiag_oldisk_gumball

Diagnostic name	CSM Diags name
olconf	sdiag_olconf_gumball
cwolconf	sdiag_cwolconf_gumball
rank	sdiag_rank_gumball
pandora	sdiag_pandora_gumball
cwhpcc	sdiag_cwhpcc_gumball
fsperf	sdiag_fio_gumball
diskperf	sdiag_fio_gumball

11.2.1 linpack

Syntax

```
sdiag_linpack_gumball
```

Description

Uses High Performance Linpack (HPL), an industry standard test to check CPU performance in terms of GFLOPS for each node in the cluster.

Arguments

`--usage` Prints the help menu
`--MEM_PCT=[MEM_PCT]` Percentage of memory on which test is intended to be run.

11.2.2 cwlinpack

Syntax

```
sdiag_cwlinpack_gumball
```

Description

`cwlinpack` is an MPI application that uses HPL to check CPU performance in terms of GFLOPS for a group of nodes or an entire Cluster.

Arguments

`--usage` Prints the help menu
`--MEM_PCT=[MEM_PCT]` Percentage of memory on which test is intended to be run.
`--NPROCS=[NPROCS]` Number of processes for the test (the "P*Q" value in the HPL.dat file).
`--MEM_TOTAL=[MEM_TOTAL]` Total amount of memory to run the test (the "N" value in the HPL.dat file).

11.2.3 nodeperf

Syntax

```
sdiag_nodeperf_gumball
```

Description

`nodeperf` starts a job on a CPU and reports CPU performance. `nodeperf` is a DGEMM routine that run on the compute node to perform double precision matrix-matrix multiply operations on each physical CPU.

Arguments

`-h` Display help screen.
`-m` Number of megabytes to consume (Default 0). This option is passed on to `cpu_perf`.
`-g` Number of gigabytes to consume (Default 1). This option is passed on to `cpu_perf`.
`-i` Number of iterations (Default 5). This option is passed on to `cpu_perf`.
`-v` verbose levels 1 through 7

11.2.4 stream

Syntax

```
sdiag_stream_gumball
```

Description

Uses Stream, an industry standard memory bandwidth performance tool. It measures sustained memory bandwidth (in MB/s).

Arguments

`-P` coverage in percentage

`-m` :run time in minutes

`-N` problem size
`-Werror` treat negative time as an error
`-c` use monotonic clock

11.2.5 olcmt

Syntax

```
sdiag_olcmt_gumball
```

Description

Runs `olcmt`, a diagnostic designed to test memory and cache components. Several memory test algorithms are used to ensure a broad test platform.

Arguments

`-marchC` Runs memory `marchC` algorithm. Single test word pattern that "marches" as a complement from cell to cell.
`-masest` Runs memory `masest` algorithm. Alternating multiple-address selection test which is used to detect faulty operation of the device address decoders.
`-address` Runs `address` pattern. Standard address pattern test to detect faulty address bits. Uses address and address complement.
`-random` Runs `random` memory test. Uses random data and complement.
`-l1cache` Runs `l1cache` test designed to exercise only the l1 cache. Implicitly the l1 cache is tested through the memory algorithms.
`-l2cache` Runs `l2cache` test designed to exercise only the l2 cache. Implicitly the l2 cache is tested through the memory algorithms.
`-dcache` Runs special `dcache` test to stress the L1 data cache. Runs a special code sequence known to create data cache errors. There is no data compare.
`-z` This option forces memory to be locked in place. This routine enforces the memory faulting routine to be single threaded which is substantially slower on larger systems. This option prevents any pages under test from being swapped out to disk while being tested.
`-r` Special option to force all processes to rotate by 4 CPUs. Period of rotation by default is set to every 3 minutes.
`-R <Rotation Period>` Enables rotation for the period specified in minutes.
`PERCENT=amount` Test amount percent of free memory. Each CPU malloc allocates only the free memory local to the node which is divided among the CPUs enabled on the local node.

11.2.6 oldisk

Syntax

```
sdiag_oldisk_gumball
```

Description

Uses oldisk, a diagnostic to test system disk by writing and reading data patterns to the disk and comparing the data.

Arguments

`-filename file` Use file as the location for testing. This option is required. Note that the file should not be a pre-existing file, as oldisk destroys and delete the file.
`-h -help` - run with an interactive help menu. Causes all other command line options to be ignored. No tests will be run.
`-filesize size` Use size megabytes of disk space for testing. The default is 128.
`-async-direct` Access the drive asynchronously, bypassing the operating systems internal buffer. (This is the default test.)
`-async-buffered` Access the drive asynchronously, using the operating systems internal buffer.
`-sync-direct` Access the drive synchronously, bypassing the operating systems internal buffer.
`-sync-buffered` Access the drive synchronously, using the operating systems internal buffer.
`-patterns patterns` Patterns is a comma separated list of data patterns to use. Choices are zeros, ones, checker, quadword, halfword, counting, and random. (Default is all).

`-paranoid` Be paranoid about the systems memory by double checking every data buffer as it is filled. This option helps prevent a memory error from showing up as a disk error, but slightly increases execution time.

11.2.7 olconf

Syntax

`sdiag_olconf_gumball`

Description

An MPI application that is intended as a system wide confidence test which runs on each single node. This test will test the sanity of CPU, memory.

Arguments

`-help` Print the help menu

`[-do_short] [-do_int] [-do_long] [-do_float] [-do_float_scatter] [-do_double] [-do_double_scatter] [-do_strid`
`[-do_all_to_all] [-bi_sect] [-timecpus] [-time_network][MegBytesPerRank=size][VARIANCE=%variance]`

11.2.8 cwolconf

Syntax

`sdiag_cwolconf_gumball`

Description

An MPI application that is intended as a system wide confidence test which runs on a clustered system. This test will test the sanity of CPU, memory and inter cluster connectivity.

Arguments

`-help` Print the help menu

`[-do_short] [-do_int] [-do_long] [-do_float] [-do_float_scatter] [-do_double] [-do_double_scatter]`
`[-do_stride] [-do_all_to_all] [-bi_sect] [-timecpus] [-time_network][MegBytesPerRank=size][VARIANCE=%variance`

11.2.9 rank

Syntax

`sdiag_rank_gumball`

Description

Rank is an MPI application that runs on a clustered system. It communicates with other ranks in the cluster through interconnect and can hence can used to stress the interconnect.

Note: Rank when run on the HPE Slingshot 11 based system, `libfabric CXI provider` does not allow to spawn more than 253 ranks per node for a single CASSINI NIC. This is a restriction introduced in HPE Slingshot 11. You must restrict the 'n' tasks per node in `sdiag-arguments.json`, "invocation_string":"–number-of-tasks-per-node=@auto" for rank.

Arguments

`-help` Print the help menu.
`-t` Type of test, either `b` for bandwidth or `l` for latency.
`-o` Type of output, either `p` for results for each rank pair (default), `t` for a table of results, or `s` for statistics per rank including average, minimum, and maximum.
`-l` Number of loops per every round, the default is 200.
`-s` Message size exchanged between rank pairs, defaults to 1000000 bytes for bandwidth tests and 4 bytes for latency tests.
`-r` Run time, number of minutes the test is to run.
`limit` If defined, specifies that only bandwidth results less than the limit, or latency results greater than the limit, are reported.
`-rankspernode` Number of processes to start per node (Default is one processes for each core).

11.2.10 pandora

Syntax

`sdiag_pandora_gumball`

Description

Pandora is a system level stress test used to stress the entire system consisting of processors, memory, I/O and network.

Arguments

`-h /-help` Print the help menu
`-v` Print pandora version and exit
`-dpca` Dumps Pre-Computed Array

`-runtime` Test time in minutes

`-mount` Mount optional drives

`-tloops` Number of test loops to execute (must be used with "-runtime 0")

`-ntwkHosts` Comma separated list of target hosts for network tests

`-stats` Print detailed process statistics

`-cpu</strong>` Run on selected CPU's only
`-noIO` Do not perform IO tests
`-noFPU` Do not perform FPU tests
`-noMEM` Do not perform MEM tests
`-noNTWK` Do not perform NTWK tests only for debugging (because these reduce test coverage)
`PERCENT=n` Percentage of free memory to test
`DATASIZE=n` Size [bits] of data used by memory tests (must be equal to 8, 16, 32, or 64)
`ALGOTYPE=` Mem RW tests type (ONTHEFLY or PRECOMPUTE)
`ADDRALGO=` Addr generation algorithm (mem RW). Could be: INC, DEC, STRIDEINC, STRIDEDEC, STRIDECNVRG, or RAND
`DATAALGO=` Data generation algorithm (mem RW). Could be: RAND, VECT, BIAS0, BIAS1, BKGRND0, or BKGRND1

11.2.11 cwhpcc

Syntax

`sdiag_cwhpcc_gumball`

Description

cwhpcc uses HPC Challenge, which is a benchmark suite that combines several benchmarks to test a number of independent attributes of the performance of HPC systems.

Arguments

`--usage` Prints the help menu
`--MEM_PCT=[MEM_PCT]` Percentage of memory on which test is intended to be run.
`--NPROCS=[NPROCS]` Number of processes for the test (the "P*Q" value in the HPL.dat file).
`--MEM_TOTAL=[MEM_TOTAL]` Total amount of memory to run the test (the "N" value in the HPL.dat file).

11.2.12 fsperf

Syntax

`sdiag_fio_gumball`

Description

`fsperf` uses FIO (Flexible I/O tester). `fsperf` checks bandwidth, IOPS, and latency by performing read or write operations to a file. `fsperf` runs on selected nodes and reports any outlier nodes. An outlier is defined by `median +/- 3 * median absolute deviations`.

Arguments

`-s` Disk space used for testing. Size can be specified in K, M, or G for kilobytes, megabytes, or gigabytes. Minimum supported size is 4M and the default size is 128M.
`-f` File on which the test runs. The default file is `fsperftestfile`.
`-r` Type of I/O pattern performed in the test.

Note: Write operations on existing files are destructive to those files and it must be performed with a caution. The supported I/O types are, read, write, readwrite(rw), randread, randwrite, randrw. The default operation is read.

`-d` Avoids performing I/O operations out of the system cache when set to '1'. The test reads or writes directly to the disk. If the value is set to '0', the test performs read or write operations through the system cache. The default value is 1.

11.2.13 diskperf

Syntax

```
sdiag_fio_gumball
```

Description

Disk uses FIO (Flexible I/O tester), `fsperf` checks bandwidth, IOPS, and latency by performing read or write operations to disk. Runs on selected nodes and reports any outlier nodes. An outlier is defined by `median +/- 3 * median absolute deviations`.

Arguments

`-s` Disk space used for testing. Size can be specified in K, M, or G for kilobytes, megabytes, or gigabytes. Minimum supported size is 4M and the default size is 128M.
`-f` Block device on which test runs.

Note: Write operations on disk partitions are destructive. The default value is `disable`.

`-r` Type of I/O pattern performed in the test.

Note: Write operations on existing files are destructive to those files and hence it must performed with a caution. The supported I/O types are, read, write, readwrite(rw), randread, randwrite, randrw. The default operation is read.

`-d` Avoids performing I/O operations out of the system cache when set to '1'. The test reads or writes directly to the disk. If the value is set to '0', the test performs read or write operations through the system cache. The default value is 1.
`-a` When set to 1, write operations are allowed on mounted devices or partitions. **Note**: Write on disk partitions is a destructive operation and it must be used with a caution. The default value is 0.

11.3 Nvidia GPU diagnostics

Prerequisites

1. The node must contain Nvidia GPUs. To verify, run this command `lspci -d 10de:*`
2. Verify if Cuda is installed.

Note: Click the individual CSM Diags name to find more information.

Diagnostic name	CSM Diags name
gpu-burn	sdiag_gpuburn_gumball
Xkbandwidth	sdiag_xkbandwidth_gumball
Xkcheck	sdiag_xkcheck_gumball
Xkdgemm	sdiag_xkdgemm_gumball
Xkmemtest	sdiag_xkmemtest_gumball
Xkmemtest	sdiag_xkbandwidth_gumball
Xkstress	sdiag_xkstress_gumball
dgnettest	N/A (present on compute nodes)

11.3.1 gpu-burn

Syntax

```
sdiag_gpuburn_gumball
```

Description

GPU CUDA stress test. Performs single or double precision matrix multiplications to stress the GPUs and report their respective health status.

Arguments

-d Numeric number If -d is supplied then it does multiplication in doubles else its float by default.

11.3.2 Xkbandwidth

Syntax

```
sdiag_xkbandwidth_gumball
```

Description

Measures memory bandwidth between CPU and GPU, and within the GPU memory to memory.

Arguments

-h Display this menu:

-m mode : Set memory mode to use (default pinned) mode: pageable - Pageable memory pinned - Non-pageable system memory

-t test Set the test to run (default quick) tests: quick - Quick measurements. See quick test options.

range - A user-specified range of values. See range test options.

shmoo - An intense shmoo of a large range of values.

```
  `-w` - Allocate pinned memory as write-combined.

  `-i iter` - Number of iterations to run (default 1).

  `-v level` - Display all output level 0-3 (default 0).

  `-p path` - Path to ini file.

  `-a`- Disable performance check.

  `-f`- Show performance failures due to noise. Applicable to DTOD transfers only.

  `-o prct` - Performance tolerance level in percentage (default 6).

  `--version` - Displays version number.
```

Quick test options only

-d: size Set default size in MB (default 64).

Range test options only -s size : starting transfer size in MB (default 32).

-e size ending transfer size in MB (default 64).
-g size increment transfer size in MB from start to end (default 4).
-n ptrn use data with data pattern (default 0 - no pattern)

```
ptrn:
0x00000001 Random data pattern
0x00000002 Sparse zero random position
0x00000004 Sparse one random position
0x00000008 Block of all ones random length
0x00000010 Block of all zeros random length
0x00000020 Solid ones right/left justified
0x00000040 Random data right/left justified
0x00000080 Groups of 2 identical adjacent bits
0x00000100 Groups of 4 identical adjacent bits
```

```
0x00000200 Groups of 8 identical adjacent bits
0x00000400 Groups of 16 identical adjacent bits
0x00000800 Groups of identical inner/outer two bits
0x00001000 Groups of identical upper three adjacent bits
0x00002000 Groups of identical lower three adjacent bits
0x00004000 0x0000000000000000
0x00008000 0xFFFFFFFFFFFFFFFF
0x00010000 0xAAAAAAAAAAAAAAAA
0x00020000 0x5555555555555555
0x00040000 0xCCCCCCCCCCCCCCCC
0x00080000 0x3333333333333333
0x00100000 0x6666666666666666
0x00200000 0x9999999999999999
0x00400000 0x7777777777777777
```

--cpu num Default CPU core to run on for single device.
--time t Time duration in seconds

11.3.3 Xkcheck

Syntax

`sdiag_xkcheck_gumball`

Description

Checks for the consistency of the GPU properties across a selected range of nodes.

Arguments

-f Make all comparisons (default is a shortened list).
-v Print test result when all nodes report the same value.

-l :Print a list of all node ids, xnames, and GPU serial numbers, one per line.

-c :Print a list of node xnames for each comparison.

-n :Print a list of node ids for each comparison.

-s :Print a list of GPU serial numbers for each comparison.

-d :Include comparisons from the CUDA Driver API.

-m :Include comparisons from the NVML API (default).

-r :Include comparisons from the CUDA Runtime API (default).

-D :Do not print CUDA Driver API comparisons (default).

-M :Do not print NVML API comparisons.

-R Do not print CUDA Runtime API comparisons.

11.3.4 Xkdgemm

Syntax

`sdiag_xkdgemm_gumball`

Description

Runs dgemm performance measurements on the GPUs.

Arguments

-h : Display this menu

-m size: Size of dimension m (default 8192)
-n size Size of dimension n (default 8192)

`-k size` Size of dimension k (default 8192)
`-i iterations` Number of iterations (default 1)
`-l loops` Number of loops (default 1)
`-p path` Path to ini file
`-a` Disable performance check
`-b` Disable results validation check
`-g` Enable MPI mode. Compares results across other nodes using same input data.
`-v level` Display output level 0-3 (default 0)
`--cpu num` Default CPU core to run on for single device.

11.3.5 Xkmemtest

Syntax

`sdiag_xkmemtest_gumball`

Description

Memory tests on the GPU memory.

Arguments

`--silent` Do not print out progress message (default)
`--device <idx>` Designate one device for test (default 0)
`--interactive` Progress info is printed in the same line.
`--disable_all` Disable all tests
`--enable_test <test_idx>` Enable the test
`--disable_test <test_idx>` Disable the test
`--max_num_blocks <n>` Set the maximum of blocks of memory to test. 1 block = 1 MB in here.
`--exit_on_error` When finding error, print error message and exit
`--monitor_temp <interval>` Monitoring temperature, the temperature is updated every `<interval>` seconds

This feature is experimental `--pattern <pattern>` : Manually set test pattern for test4/test8/test10

`--list_tests` : List all test descriptions

`--num_iterations <n>` Set the number of iterations (only effective on test0 and test10)
`--num_passes <n>` Set the number of test passes (this affects all tests)

`--disable_serial_number` : Disable reporting serial number

`--verbose <n>` Set verbose level

0 - Print out test start and end message only (default)

1 - Print out pattern messages in test

2 - Print out progress messages

`--stress` : Stress test. Equivalent to `--disable_all --enable_test` 10

`--num_iterations` : 1000 `--num_passes` 40 `--exit_on_error`

`--cpu num` : Default CPU core to run on for single device

`--help` Print this message

11.3.6 Xkmemtest

Syntax

`sdiag_xkbandwidth_gumball`

Description

Measures memory bandwidth between CPU and GPU, and within the GPU memory to memory.

Arguments

`--silent` :Do not print out progress message (default)

`--device <idx>` Designate one device for test (default 0)

`--interactive` : Progress info is printed in the same line.

`--disable_all` : Disable all tests

`--enable_test <test_idx>` Enable the test
`--disable_test <test_idx>` Disable the test
`--max_num_blocks <n>` Set the maximum of blocks of memory to test. 1 block = 1 MB in here

`--exit_on_error` : When finding error, print error message and exit

`--monitor_temp <interval>` Monitoring temperature, the temperature is updated every `<interval>` seconds. This feature is experimental.
`--pattern <pattern>` Manually set test pattern for test4/test8/test10

`--list_tests` : List all test descriptions

`--num_iterations <n>` Set the number of iterations (only effective on test0 and test10)
`--num_passes <n>` Set the number of test passes (this affects all tests)

`--disable_serial_number` : Disable reporting serial number

`--verbose <n>` Set verbose level

0 - Print out test start and end message only (default)

1 - Print out pattern messages in test

2 - Print out progress messages

`--stress` : Stress test. Equivalent to `--disable_all --enable_test` 10

`--num_iterations` : 1000 `--num_passes` 40 `--exit_on_error`

`--cpu num` : Default CPU core to run on for single device.

`--help` : Print this message

11.3.7 Xkstress

Syntax

`sdiag_xkstress_gumball`

Description

Performs performance measurements on the GPU and does memory transfers right after to stress both the GPU processor and the PCIe link.

Arguments

`-c` : Print a list of node xnames for each comparison.

`-d delay` Set delay between passes, i.e. cool down period in seconds. (Default 0).

`-h` : Print this help message.

`-i iter` Set test iteration repeat count. (Default 10).
`-l list` Set the test list. Default is all tests “stream,gemm,pcie”. gemm=“dgemm,sgemm,zgemm,cgemm”, stream=“copy,scale,add,triad”
`-m size` Set matrix size for matrix multiplication tests. (Default 4096).

`-n` : Print a list of node ids for each comparison.

`-o` : Print a list of node nids or xnames that are not within tolerance and underperforming relative to other nodes.

`-p` : Set the number of passes to run the test. (Default 3).

`-q` : Print test result only when tolerance is exceeded.

`-t pct` Percent tolerance for performance variation between nodes. (Default 5, whole number).

`-v` : Print results for each iteration of a test.

`-x size` PCIe transfer size, in megabytes. (Default 64).
`--cpu n` Default CPU core to run on for single device.

11.4 AMD GPU diagnostics

Prerequisites

1. The node must contain AMD GPUs. To verify, run this command `lspci | grep "Display controller"`
2. Verify if ROCM is installed.

Note: Click the individual CSM Diags name to find more information.

Diagnostic name	CSM Diags name
amdgpubandwidth	sdiags-amdgpubandwidth-gumball
amdgpuproperties	sdiags-amdgpuproperties-gumball
amdgpuedpp	sdiags-amdgpuedpp-gumball
amdgpufilechk	sdiags-amdgpufilechk-gumball
amdgpukernelchk	sdiags-amdgpukernelchk-gumball
amdgpulinkchk	sdiags-amdgpulinkchk-gumball
amdgpumonitor	sdiags-amdgpumonitor-gumball
amdgpup2pchk	sdiags-amdgpup2pchk-gumball
amdgpupciechk	sdiags-amdgpupciechk-gumball
amdgpupciemonitor	sdiags-amdgpupciemonitor-gumball
amdgpupkgchk	sdiags-amdgpupkgchk-gumball
amdgpusbioschk	sdiags-amdgpusbioschk-gumball
amdgpustresstest	sdiags-amdgpustresstest-gumball
amdgpuuserchk	sdiags-amdgpuuserchk-gumball

11.4.1 amdgpubandwidth

Syntax

`sdiags-amdgpubandwidth-gumball`

Description

The PCIe bandwidth benchmark attempts to saturate the PCIe bus with DMA transfers between system memory and a target GPU card's memory. These are known as host-to-device or device-to-host transfers, and can be either unidirectional or bidirectional transfers. The maximum bandwidth obtained is reported.

Arguments

`--host_to_device` This key indicates if host to device transfers are considered. The default value is true.
`--device_to_host` This key indicates if device to host transfers are considered. The default value is true.
`--parallel` This option is only used if the `test_bandwidth` key is true. - true – Run all test transfers in parallel. - false – Run test transfers one by one.
`--duration` This option is only used if `test_bandwidth` is true. This key specifies the duration a transfer test should run, given in milliseconds. If this key is not specified, the default value is 10000 (10 seconds).
`--log_interval` This option is only used if `test_bandwidth` is true. This is a positive integer, given in milliseconds, that specifies an interval over which the moving average of the bandwidth is calculated and logged. The default value is 1000 (1 second). It must be smaller than the duration key. If this key is 0 (zero), results are displayed as soon as the test transfer is completed.
`--block_size` **(Optional)** Defines list of block sizes to be used in transfer tests.

`--b2b_block_size` : This option is only used if both `test_bandwidth` and `parallel` keys are true. This is a positive integer indicating size in bytes of a data block to be transferred continuously ("back-to-back") for the duration of one test pass. If the key is not present, ordinary transfers with size indicated in `block_size` key are performed.

`--link_type` : This is a positive integer indicating type of link to be included in bandwidth test. Numbering follows that listed in `hsa_amd_link_info_type_t` in `hsa_ext_amd.h` file.

11.4.2 amdgpuproperties

Syntax

`sdiags-amdgpuproperties-gumball`

Description

The GPU Properties module queries the configuration of a target device and returns the device's static characteristics. These static values can be used to debug issues such as device support, performance, and firmware problems.

Arguments

`--properties` The properties key specifies what configuration property or properties the query is interested in.
`--io_links-properties` The properties key specifies what configuration property or properties the query is interested in.

11.4.3 amdgpuedpp

Syntax

`sdiags-amdgpuedpp-gumball`

Description

The Input EDPp test can be used to characterize the peak power capabilities of a GPU to different levels of use. This tool can leverage the functionality of the GST to drive the compute load on the GPU, but the test uses different configuration and output keys and should focus on driving power usage rather than calculating compute load. The purpose of the IET module is to bring the GPU(s) to a pre-configured power level in watts by gradually increasing the compute load on the GPUs until the desired power level is achieved. This verifies that the GPUs can sustain a power level for a reasonable amount of time without problems like thermal violations arising.

Arguments

`--target_power` This is a floating point value specifying the target sustained power level for the test.
`--ramp_interval` This is an time interval, specified in milliseconds, given to the test to determine the compute load that sustains the target power. The default value is 5000 (5 seconds). This time is counted against the duration of the test.
`--tolerance` A value indicating how much the target_power can fluctuate after the ramp period for the test to succeed. The default value is 0.1 or 10%.
`--max_violations` The number of tolerance violations that can occur after the ramp_interval for the test to still pass. The default value is 0.
`--sample_interval` The sampling rate for target_power values given in milliseconds. The default value is 100 (0.1 seconds).
`--log_interval` This is a positive integer, given in milliseconds, that specifies an interval over which the moving average of the bandwidth is calculated and logged.

11.4.4 amdgpufilechk

Syntax

`sdiags-amdgpufilechk-gumball`

Description

This feature checks for the existence of a file, its owner, group, permissions and type. The primary purpose of this module is to check that the device interfaces for the driver and the KFD are available, but it can also be used to check for the existence of important configuration files, libraries, and executables.

Arguments

`--file` The value of this key should satisfy the file name limitations of the target operating system and specifies the file to check. This key is required.
`--owner` The expected owner of the file. If this key is specified ownership is tested.
`--group` If this key is specified, group ownership is tested.
`--permission` If this key is specified, the permissions on the file are tested. The permissions are expected to match the permission value given.
`--type` If this key is specified the file type is checked.
`--exists` If this key is specified and set to false all optional parameters will be ignored and a check will be made to make sure the file does not exist. The default value for this key is true

11.4.5 amdgpukernelchk

Syntax

`sdiags-amdgpukernelchk-gumball`

Description

The `rcqt-kernelcheck` module determines the version of the operating system and the kernel installed on the platform and compares the values against the list of supported values.

Arguments

`--os_versions` A collection of strings corresponding to operating systems names, such as {“Ubuntu 16.04.3 LTS”, “Centos 7.4”, and so on.}

`--kernel_versions` A collection of strings corresponding to kernel version names, such as, {“4.4.0-116-generic”, “4.13.0-36-generic”, and so on.}

11.4.6 amdgpulinkchk

Syntax

`sdiags-amdgpulinkchk-gumball`

Description

This feature checks that a search by the linker or loader for a library finds the correct version in the correct location. The check should include a `SONAME` version of the library, the expected location, and the architecture of the library.

Arguments

`--soname` This is the `SONAME` of the library for the check. An `SONAME` library contains the major version of the library in question.

`--arch` This value qualifies the architecture expected for the library.

`--ldpath` :This is the fully qualified path where the library is expected to be located.

11.4.7 amdgpumonitor

Syntax

`sdiags-amdgpumonitor-gumball`

Description

The GPU monitor module is used to monitor and characterize the response of a GPU to different levels of use. This module is intended to run concurrently with other actions, and provides a `start` and `stop` configuration key to start the monitoring and then stop it after testing has completed. The module can also be configured with bounding box values for interested GPU parameters. If any of the GPU's parameters exceed the bounding values on a specific GPU an INFO warning message will be printed to stdout while the bounding value is still exceeded.

Arguments

`--monitor` If this this key is set to true, the GM module starts monitoring on specified devices. If this key is set to false, all other keys are ignored and monitoring of the specified device is stopped.

`--metrics` The set of metrics to monitor during the monitoring period.

`--sample_interval` If this key is specified metrics are sampled at the given rate. The units for the `sample_interval` are milliseconds. The default value is 1000.

`--log_interval` If this key is specified informational messages is emitted at the given interval, providing the current values of all parameters specified. This parameter must be equal to or greater than the sample rate. If this value is not specified, no logging occurs.

`--terminate` If the terminate key is `true`, the GM monitor terminates the RVS process when a bounds violation is encountered on any of the metrics specified.

`--force` If `true` and terminate key is also `true`, the RVS process terminates immediately. **Note**: This may cause resource leaks within GPUs.

11.4.8 amdgpup2pchk

Syntax

`sdiags-amdgpup2chk-gumball`

Description

The P2P qualification tool is designed to provide the list of all GPUs that support P2P and characterize the P2P links between peers. In addition to testing for P2P compatibility, this test performs a peer-to-peer throughput test between all unique P2P pairs for performance evaluation. These are known as device-to-device transfers, and can be either unidirectional or bidirectional. The average bandwidth obtained is reported to help debug low bandwidth issues.

Arguments

`--peers` This is a required key, and specifies the set of GPU(s) considered being peers of the GPU specified in the action. If `all` is specified, all other GPU(s) on the system are considered peers. Otherwise, only the GPU IDs specified in the list are considered.

`--peer_deviceid` This is an optional parameter, but if specified it restricts the peers list to a specific device type corresponding to the `deviceid`.

`--test_bandwidth` If this key is set to true, the P2P bandwidth benchmark runs if a pair of devices pass the P2P check.

`--bidirectional` This option is only used if `test_bandwidth` key is true.

`--parallel` This option is only used if the `test_bandwidth` key is true.

`--duration` This option is only used if `test_bandwidth` is true. This key specifies the duration a transfer test must run, given in milliseconds. If this key is not specified, the default value is 10000 (10 seconds).

`--log_interval` This option is only used if `test_bandwidth` is true. This is a positive integer, given in milliseconds, that specifies an interval over which the moving average of the bandwidth is calculated and logged. The default value is 1000.

`--block_size` Defines list of block sizes to be used in transfer tests.

`--b2b_block_size` This option is only used if both `test_bandwidth` and `parallel` keys are true. This is a positive integer indicating size in bytes of a data block to be transferred continuously (`back-to-back`) for the duration of one test pass.

`--link_type` This is a positive integer indicating type of link to be included in bandwidth test. Numbering follows that listed in `hsa_amd_link_info_type_t` in `hsa_ext_amd.h` file.

11.4.9 amdgpupciechk

Syntax

`sdiags-amdgpupciechk-gumball`

Description

PCI express qualification tool module targets and qualifies the configuration of the platforms PCIe connections to the GPUs.

Arguments

`--capability` The PCIe capability key contains a collection of structures that specify which PCIe capability to check and the expected value of the capability. A check structure must contain the PCIe capability value, but an expected value may be omitted. The value of all valid capabilities that are a part of this collection are entered into the `capability_value` field.

11.4.10 amdgpupciemonitor

Syntax

`sdiags-amdgpupciemonitor-gumball`

Description

The PCIe State Monitor (PESM) tool is used to actively monitor the PCIe interconnect between the host platform and the GPU. The module registers `listener` on a target GPU's PCIe interconnect and log a message whenever it detects a state change. The PESM can detect the following state changes:

- PCIe link speed changes
- GPU device power state changes

Arguments

`--monitor` If this key is set to true, the PESM module starts monitoring specified devices. If this key is set to false, all other keys are ignored and monitoring is stopped for all devices.

11.4.11 amdgpupkgchk

Syntax

`sdiags-amdgpupkgchk-gumball`

Description

This feature is used to check installed packages on the system. It provides checks for installed packages and the currently available package versions, if applicable.

Arguments

`--package` Specifies the package to check. This key is required.
`--version` This is an optional key specifying a package version. If it is provided, the tool checks if the version is installed, failing otherwise. If it is not provided any version matching the package name results in success.

11.4.12 amdgpusbioschk

Syntax

`sdiags-amdgpusbioschk-gumball`

Description

The GPU SBIOS mapping qualification tool is designed to verify that a platform's SBIOS has satisfied the BAR mapping requirements for VDI and Radeon Instinct products for ROCm support.

Arguments

`--bar1_req_size` This is an integer specifying the required size of the BAR1 frame buffer region.
`--bar1_base_addr_min` This is an integer specifying the minimum value of the BAR1 base address.
`--bar1_base_addr_max` This is an integer specifying the maximum value of the BAR1 base address.
`--bar2_req_size` This is an integer specifying the required size of the BAR2 frame buffer region.
`--bar2_base_addr_min` This is an integer specifying the minimum value of the BAR2 base address.
`--bar2_base_addr_max` This is an integer specifying the maximum value of the BAR2 base address.
`--bar4_req_size` This is an integer specifying the required size of the BAR4 frame buffer region.
`--bar4_base_addr_min` This is an integer specifying the minimum value of the BAR4 base address.
`--bar4_base_addr_max` This is an integer specifying the maximum value of the BAR4 base address.
`--bar5_req_size` This is an integer specifying the required size of the BAR5 frame buffer region.

11.4.13 amdgpustresstest

Syntax

`sdiags-amdgpustresstest-gumball`

Description

The GPU stress test modules purpose is to bring the CPUs of the specified GPU(s) to a target performance level in `gigaflops` by performing large matrix multiplications using SGEMM or DGEMM (Single or Double-precision General Matrix Multiplication) available in a library like `rocBlas`. The GPU stress module may be configured so it does not copy the source arrays to the GPU before every matrix multiplication. This allows the GPU performance to not be capped by the device to host bandwidth transfers. The module calculates how many matrix operations per second are necessary to achieve the configured performance target and fails if it cannot achieve that target.

Arguments

`--target_stress` The maximum relative performance the GPU attempts to achieve in `gigaflops`. This parameter is required.
`--copy_matrix` This parameter indicates if each operation should copy the matrix data to the GPU before executing. The default value is true.
`--ramp_interval` This parameter indicates a time interval, specified in milliseconds, given to the test to reach the given `target_stress` `gigaflops`. The default value is 5000 (5 seconds)

`--tolerance` A value indicating how much the `target_stress` can fluctuate after the ramp period for the test to succeed. The default value is 0.1 or 10%.
`--max_violations` The number of tolerance violations that can occur after the `ramp_interval` for the test to still pass. The default value is 0.
`--log_interval` This is a positive integer, given in milliseconds, that specifies an interval over which the moving average of the bandwidth is calculated and logged.
`--matrix_size` Size of the matrices of the SGEMM operations. The default value is 5760.

11.4.14 amdgpuuserchk

Syntax

`sdiags-amdgpuuserchk-gumball`

Description

This feature checks for the existence of a user and the user's group membership.

Arguments

`--user` Specifies the username to check. This key is required.
`--groups` This is an optional key specifying a collection of groups that the user is associated. The user's membership in each group is checked.

11.5 Fabric diagnostics

- dgnettest
- check_excessive_pause

11.5.1 dgnettest

Syntax

dgnettest_run.sh (present on application nodes in `/usr/local/diag/bin`)

Description

Rosetta network diagnostic to measure bandwidth and latency. Performs all-to-all and bisection bandwidth test and latency validation.

Usage: `./dgnettest_run.sh [options] [all2all | bisect | latency]`

Note: The script `dgnettest_run.sh` supports only ***SLURM*** Workload manager. For more information on usage of `dgnettest_run.sh`, see ***Network Diagnostics*** in ***Slingshot Troubleshooting Guide***.

Arguments

`-b` Sets slurm to broadcast out `dgnettest` to the nodes. The default is no broadcasting of files.
`-d` Debugs
`-c class` Sets network class type of 0-4. The default value is 2.
`-i` Ignores hostfile and MAC address errors.
`-l latency` Sets the high CV threshold for the latency test. Value must be between 0 and 1. The default value is 0.05.
`-m size` Specifies size of messages used in bytes. The default value is 131072.
`-n nodelist` Selects nodes to use. The default value is ***all***.
`-N NIC` Runs over a single, user selected NIC (must select 0 to 1).
`-s set-size` Runs test over sets of nodes of a given size. The default value is 512.
`-t time` Specifies runtime for each test. The default value is default 30.
`-T threshold` Sets the low bandwidth threshold. Value must be between 0 and 1. The default value is 0.9.
`-C` Runs the test over both NICs concurrently or, if using multiple NICs and changing -p, the NICs to ranks mapping is required.
`-M mpi_mode` Sets mpi mode for srun. The default value is ***none***.
`-p PPN` Sets the number of processes per node. The default value is selected by ***NUMA***.
`-r reps` Sets the number of repetitions to run each test. The default value is ***auto***.
`-v` Verbose

11.5.2 Check Excessive Pause

Syntax

check_excessive_pause.sh (present on fabric manager)

Description

Checks all ports on the fabric that are reachable from the fabric manager. It checks for an error flag that indicates that there are excessive amounts of PAUSE frames on a port.

Arguments

N/A

11.6 OSU Benchmark

Currently, only OSU Benchmark MPI binaries are supported:

CSM Diags name	Category
osu_startup	startup
osu_bw_bibw	pt2pt
osu_single_multi_latency	pt2pt
osu_multiplebw_message_rate	pt2pt
osu_multithread_multiprocess_latency	pt2pt
osu_bw_latency_ops	one-sided
osu_put_bibw	one-sided
osu_get_acc_latency	one-sided
osu_collective_blocking_barrier	collective
osu_collective_MPI_blocking_ops	collective
osu_collective_MPI_non_blocking_ibarrier	collective
osu_collective_MPI_non_blocking_ops	collective

11.6.1 osu_startup

Syntax

`osu_hello osu_init`

Arguments

N/A

11.6.2 osu_bw_bibw

Syntax

`osu_bw osu_bibw`

Arguments

`-m, --message-size [MIN:]MAX` set the minimum or maximum message size to MIN or MAX bytes respectively.

Examples:

```
`-m 128 // min = default, max = 128`

`-m 2:128 // min = 2, max = 128`

`-m 2: // min = 2, max = default`
```

`-M, --mem-limit SIZE` set per process maximum memory consumption to SIZE bytes(default 536870912).
`-i, --iterations ITER` set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
`-x, --warmup ITER` set number of warmup iterations to skip before timing (default 200).

-W, --window-size SIZE set number of messages to send before synchronization (default 64).
-h, --help print this help.
-v, --version print version info.

11.6.3 osu_single_multi_latency

Syntax

osu_latency osu_multi_lat

Arguments

-m, --message-size [MIN:]MAX set the minimum or the maximum message size to MIN or MAX bytes respectively.

Examples

```
`-m 128 // min = default, max = 128`

`-m 2:128 // min = 2, max = 128`

`-m 2: // min = 2, max = default`
```

-M, --mem-limit SIZE set per process maximum memory consumption to SIZE bytes(default 536870912).
-i, --iterations ITER set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
-x, --warmup ITER set number of warmup iterations to skip before timing (default 200).
-h, --help print this help.
-v, --version print version info.

11.6.4 osu_multiplebw_message_rate

Syntax

osu_mbw_mr

Arguments

-R=<0,1>, --print-rate Print unidirectional message rate (default 1).
-p=<pairs>, --num-pairs Number of pairs involved (default np / 2).
-W=<window>, --window-size Number of messages sent before acknowledgement (default 64). This argument cannot be used with -v.
-V, --vary-window Vary the window size (default no). (This argument cannot be used with -W).
-m, --message-size [MIN:]MAX Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

```
-m 128 // min = default, max = 128

-m 2:128 // min = 2, max = 128

-m 2: // min = 2, max = default
```

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes (default 536870912).
-h, --help Print this help

11.6.5 osu_multithread_multiprocess_latency

Syntax

osu_latency_mp osu_latency_mt

Arguments

-m, --message-size [MIN:]MAX Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

`-m 128 // min = default, max = 128`

`-m 2:128 // min = 2, max = 128`

`-m 2: // min = 2, max = default`

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes (default 536870912).
-i, --iterations ITER Set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
-x, --warmup ITER Set number of warmup iterations to skip before timing (default 200).
-t, --num_processes SEND:[RECV] Set the sender and receiver number of processes.

min: 1

default: (receiver processes: 2 sender processes: 1)

max: 128.

Examples

`-t 4 // receiver processes = 4 and sender processes = 1`

`-t 4:6 // sender processes = 4 and receiver processes = 6`

`-t 2: // not defined`

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes
-h, --help Print this help
-v, --version Print version info

11.6.6 osu_bw_latency_ops

Syntax

osu_acc_latency osu_fop_latency osu_get_latency osu_put_latency osu_cas_latency osu_get_bw osu_put_bw

11.6.6.0.1 Arguments

-w, --win-option <win_option> can be any of the following:

create Use MPI_Win_create to create an MPI Window object.

allocate Use MPI_Win_allocate to create an MPI Window object.

dynamic Use MPI_Win_create_dynamic to create an MPI Window object.

-s, --sync-option <sync_option> can be any of the follows:

pscw Use Post, Start, Complete, or Wait synchronization calls.

fence Use MPI_Win_fence synchronization call.

lock Use MPI_Win_lock/unlock synchronizations calls.

flush Use MPI_Win_flush synchronization call.

flush_local Use MPI_Win_flush_local synchronization call.

lock_all Use MPI_Win_lock_all/unlock_all synchronization calls.

-m, --message-size[MIN:]MAX Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

-m 128 // min = default, max = 128

-m 2:128 // min = 2, max = 128

-m 2: // min = 2, max = default

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes (default 536870912).

-x, --warmup ITER Number of warmup iterations to skip before timing (default 100).

-i, --iterations ITER Number of iterations for timing (default 10000).

-h, --help Print this help message

11.6.7 osu_put_bibw

Syntax

osu_put_bibw

Arguments

-w, --win-option <win_option> can be any of the following:

create : Use MPI_Win_create to create an MPI Window object.

allocate : Use MPI_Win_allocate to create an MPI Window object.

dynamic : Use MPI_Win_create_dynamic to create an MPI Window object.

-s, --sync-option <sync_option> can be any of the following:

pscw : Use Post, Start, Complete, or Wait synchronization calls.

fence : Use MPI_Win_fence synchronization call.

-m, --message-size [MIN:]MAX Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

```
-m 128 // min = default, max = 128

-m 2:128 // min = 2, max = 128

-m 2: // min = 2, max = default
```

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes (default 536870912).
-x, --warmup ITER Number of warmup iterations to skip before timing (default 100).
-i, --iterations ITER Number of iterations for timing (default 10000).
-h, --help Print this help message.

11.6.8 osu_get_acc_latency

Syntax

osu_get_acc_latency

Arguments

-m, --message-size [MIN:]MAX Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

```
-m 128 // min = default, max = 128

-m 2:128 // min = 2, max = 128

-m 2: // min = 2, max = default
```

-M, --mem-limit SIZE Set per process maximum memory consumption to SIZE bytes (default 536870912).
-x ITER Number of warmup iterations to skip before timing (default 100).
-i ITER Number of iterations for timing (default 10000).

win_option:

create : Use MPI_Win_create to create an MPI Window object.

allocate : Use MPI_Win_allocate to create an MPI Window object.

`dynamic` : Use `MPI_Win_create_dynamic` to create an MPI Window object.

`sync_option`:

`lock` : Use `MPI_Win_lock/unlock` synchronizations calls.

`flush` : Use `MPI_Win_flush` synchronization call.

`flush_local` : Use `MPI_Win_flush_local` synchronization call.

`lock_all` : Use `MPI_Win_lock_all/unlock_all` synchronization calls.

`pscw` : Use `Post`, `Start`, `Complete`, or `Wait` synchronization calls.

`fence` : Use `MPI_Win_fence` synchronization call.

11.6.9 osu_collective_blocking_barrier

Syntax

`osu_collective_blocking_barrier`

Arguments

`-i, --iterations ITER` Set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
`-x, --warmup ITER` Set number of warmup iterations to skip before timing (default 200).
`-f, --full` Print full format listing (MIN/MAX latency and ITERATIONS displayed in addition to AVERAGE latency).
`-h, --help` Print this help.
`-v, --version` Print version info.

11.6.10 osu_collective_MPI_blocking_ops

Syntax

`osu_allgatherv` `osu_alltoallv` `osu_gatherv` `osu_scatter` `osu_allreduce` `osu_bcast` `osu_reduce` `osu_scatterv`

Arguments

`-m, --message-size [MIN:]MAX` Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

```
-m 128 // min = default, max = 128

-m 2:128 // min = 2, max = 128

-m 2: // min = 2, max = default
```

`-M, --mem-limit SIZE` Set per process maximum memory consumption to SIZE bytes (default 536870912).
`-i, --iterations ITER` Set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
`-x, --warmup ITER` Set number of warmup iterations to skip before timing (default 200).
`-f, --full` Print full format listing (MIN/MAX latency and ITERATIONS displayed in addition to AVERAGE latency).
`-h, --help` Print this help.
`-v, --version` Print version info.

11.6.11 osu_collective_MPI_non_blocking_ibarrier

Syntax

`collective osu_ibarrier`

Arguments

`-i, --iterations ITER` Set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
`-x, --warmup ITER` Set number of warmup iterations to skip before timing (default 200).
`-f, --full` Print full format listing (MIN/MAX latency and ITERATIONS displayed in addition to AVERAGE latency).
`-t, --num_test_calls CALLS` Set the number of `MPI_Test()` calls during the dummy computation, set CALLS to 100, 1000, or any number greater than 0.

`-h, --help` Print this help
`-v, --version` Print version info.

11.6.12 osu_collective_MPI_non_blocking_ops

Syntax

`osu_iallgatherv osu_ialltoallv osu_igather osu_iscatter osu_iallreduce osu_ialltoallw osu_igatherv osu_iscatterv osu_iallgather osu_ialltoall osu_ibcast osu_ireduce`

Arguments

`-m, --message-size [MIN:]MAX` Set the minimum and/or the maximum message size to MIN and/or MAX bytes respectively.

Examples

```
`-m 128 // min = default, max = 128`

`-m 2:128 // min = 2, max = 128`

`-m 2: // min = 2, max = default`
```

`-M, --mem-limit SIZE` Set per process maximum memory consumption to SIZE bytes (default 536870912).
`-i, --iterations ITER` Set iterations per message size to ITER (default 1000 for small messages, 100 for large messages).
`-x, --warmup ITER` Set number of warmup iterations to skip before timing (default 200).
`-f, --full` Print full format listing (MIN/MAX latency and ITERATIONS displayed in addition to AVERAGE latency).
`-t, --num_test_calls CALLS` Set the number of `MPI_Test()` calls during the dummy computation, set CALLS to 100, 1000, or any number greater than 0.
`-h, --help` Print this help.
`-v, --version` Print version information.

11.7 Restore Postgres

The following steps are required to restore data to a Postgres cluster:

11.7.1 Restore Postgres for Cray hms badger

Assuming that a dump of the database exists. If the badger Postgres cluster is in a state that the cluster must be rebuilt and the data is restored, the following procedures are recommended:

1. Copy the database dump to an accessible location.
    - If the database is manually backed up, verify that the dump file exists in a location of the Postgres cluster. This location is required in the subsequent steps.
    - If the database is automatically backed up, the most recent version of the dump and secrets must exist in the `postgres-backup` S3 bucket. These are required in the subsequent steps. List the files in the `postgres-backup` S3 bucket and if the files exist, download the dump and secrets out of the S3 bucket. The following Python3 scripts are used to list and download the files.

    Note: The `.psql` file contains the database dump and `.manifest` file contains the secrets. The `aws_access_key_id` and `aws_secret_access_key` must be set based on the `postgres-backup-s3-credentials` secret.

    ```
    ncn-w001# export S3_ACCESS_KEY=`kubectl get secrets postgres-backup-s3-credentials \
    -ojsonpath='{.data.access_key}' | base64 --decode`

    ncn-w001# export S3_SECRET_KEY=`kubectl get secrets postgres-backup-s3-credentials \
    -ojsonpath='{.data.secret_key}' | base64 --decode`
    ```

    list.py:

    ```
    import io
    import boto3
    import os

    # postgres-backup-s3-credentials are needed to list keys in the postgres-backup bucket
    ```

```
s3_access_key = os.environ['S3_ACCESS_KEY']
s3_secret_key = os.environ['S3_SECRET_KEY']

s3 = boto3.resource(
    's3',
    endpoint_url='http://rgw-vip.nmn',
    aws_access_key_id=s3_access_key,
    aws_secret_access_key=s3_secret_key,
    verify=False)

backup_bucket = s3.Bucket('postgres-backup')
for file in backup_bucket.objects.filter(Prefix='cray-hms-badger-postgres'):
    print(file.key)
```

download.py:

Update the script for the specific `.manifest` and `.psql` files that must be downloaded from S3.

```
import io
import boto3
import os

# postgres-backup-s3-credentials are needed to download from postgres-backup bucket

s3_access_key = os.environ['S3_ACCESS_KEY']
s3_secret_key = os.environ['S3_SECRET_KEY']

s3_client = boto3.client(
    's3',
    endpoint_url='http://rgw-vip.nmn',
    aws_access_key_id=s3_access_key,
    aws_secret_access_key=s3_secret_key,
    verify=False)

response = s3_client.download_file('postgres-backup', \
'cray-hms-badger-postgres-2022-01-13T14:45:12.manifest', \
'cray-hms-badger-postgres-2022-01-13T14:45:12.manifest')

response = s3_client.download_file('postgres-backup', \
'cray-hms-badger-postgres-2022-01-13T14:45:12.psql', \
'cray-hms-badger-postgres-2022-01-13T14:45:12.psql')
```

2. Copy the database dump file to the Postgres member.

```
ncn-w001# DUMPFILE=cray-hms-badger-postgres-2022-01-13T14:45:12.psql
ncn-w001# NAMESPACE=services
ncn-w001# POSTGRESQL=cray-hms-badger-postgres

ncn-w001# kubectl cp ./${DUMPFILE} "${POSTGRESQL}-0":/home/postgres/${DUMPFILE} \
-c postgres -n ${NAMESPACE}
```

3. Restore the data.

```
ncn-w001# kubectl exec "${POSTGRESQL}-0" -c postgres -n ${NAMESPACE} \
-it -- psql -U postgres < ${DUMPFILE}
```