SAND2021-13644C

Exceptional service in the national interest

Sandia National Laboratories

OGC | Open Grid Computing, Austin, TX

OGC

LDMS Version 4.3.8 Advanced Tutorial: Part 1

https://github.com/ovis-hpc/ovis

Jim Brandt, Ann Gentile

Sandia National Laboratories

Tom Tucker

Open Grid Computing, Inc.

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Configuration Options and Parameter Tuning

Environment Variables

The following environment variables must be set:

LD_LIBRARY_PATH

Path to ovis libraries if not defined in a system default path (**/opt/ovis/lib64 & /opt/ovis/lib64/ovis-ldms** for this tutorial).

PATH

Include the path to sbin directory containing ldmsd (**/opt/ovis/sbin** in this tutorial)

Include the path to bin directory containing python modules (/opt/ovis/bin) **# Necessary If using ldmsd_controller**

ZAP_LIBPATH

Path to ovis/lib/ovis-ldms (**/opt/ovis/lib64/ovis-ldms** for this tutorial)

LDMSD_PLUGIN_LIBPATH

Path to ovis/lib/ovis-ldms (**/opt/ovis/lib64/ovis-ldms** for this tutorial)

The following environment variables may be set to override compiled-in defaults:

LDMSD_PIDFILE

Full path name of pidfile (override the **default of "/var/run/ldmsd.pid"**

Both default and environment variable can be overridden by use of the command line argument "-r pidfilepath"

LDMSD_MEM_SZ

Define the size of memory reserved for metric sets (override the **default of 512KB**).

This can be overridden by use of the command line argument "-m <size>"

ZAP_IO_MAX sets the maximum number of IO threads that can be brought up (up to NUM_PROCS (including if hyperthreading))

ZAP_IO_BUSY sets the busy threshold for spinning up a new I/O thread (float between 0 and 1.0 default=0.80)

LDMSD_UPDTR_OFFSET_INCR

The increment to the offset hint in microseconds. **This is only for updaters that determine the update interval and offset automatically. E**xample: if the offset hint is **100000 (100 millisecond),** the updater offset will be 100000 + LDMSD_UPDTR_OFFSET_INCR (**Default is 100000 (100 milliseconds)**).

General/Configuration Options

Sandia National Laboratories

OGC

-F Run in foreground mode; don't daemonize the program. **Default is false**

-B version-file-mode

When run in daemon mode, controls the existence of the banner file. Mode 0 suppresses the version file. **Mode 1 deletes it at daemon exit**. Mode >= 2 leaves it in place for debugging after daemon exit. **Default mode is 1**. The banner contains the software and protocol versions information, which is also logged at the INFO level. The banner file name is always the pidfile name with .version appended.

-c CONFIG_PATH

The path to configuration file (optional, **default: <none>**). The configuration file contains a batch of ldmsd controlling commands, such as `load` for loading a plugin, and `prdcr_add` for defining a ldmsd producer to aggregate from (see **ldmsd_controller**(8) for a complete list of commands, or simply run help from inside the **ldmsd_controller**). The commands in the configuration file are executed sequentially, except for **prdcr_start**, **updtr_start**, **strgp_start**, and **failover_start that will be deferred**. The **failover service, if present, will start first (among the deferred)**. Then, upon failover pairing success or failure, the other deferred configuration objects will be started. Note: while the failover service **active**, prdcr, updtr, and strgp cannot be altered (start, stop, or reconfigure).

-m MEMORY_SIZE

MEMORY_SIZE is the maximum size of pre-allocated memory for metric sets. The given size must be less than 1 petabytes. For example, 20M or 20mb are 20 megabytes. The default is adequate for most ldmsd acting in the sampling role. For aggregating ldmsd, a rough estimate of pre-allocated memory needed is (Number of nodes aggregated) x (Number of metric sets per node) x 4k. Data sets containing arrays may require more. The estimate can be checked by enabling DEBUG logging and examining the mm_stat bytes_used+holes value at ldmsd exit.

-n NAME

The name of the daemon. **Default is "HOSTNAME:PORT"**.

-r pid_file

The path to the pid file and prefix of the .version banner file for daemon mode. **Default is /var/run/ldmsd.pid**

-V Display LDMS version information and then exit.

-u plugin_name – This is unreliable, unnecessary, and will be removed.

Display the usage for named plugin. Special names all, sampler, and store match all, sampler type, and store type plugins, respectively.

General/Configuration Options Cont.

Sandia National Laboratories

OGC

-x XPRT:PORT

Specifies the transport type to listen on. May be specified more than once for multiple transports. The XPRT string is one of 'rdma', 'sock', or 'ugni' (CRAY XE/XK/XC). A transport specific port number must be specified following a ':', e.g. rdma:10000.

The listening transports can also be specified in the configuration file using listen command, e.g. `listen xprt=sock port=1234`.

-a AUTH

Define the **default LDMS Authentication method for the LDMS connections in this daemon** (when the connections do not specify authentication method/domain). See man pages on **ldms_authentication**(7) for more information. **Default is "none" (no authentication)**.

-A NAME**=**VALUE

Passing the NAME=VALUE option to the LDMS Authentication plugin. This command line option can be given multiple times. Please see **ldms_authentication**(7) for more information, and consult the plugin manual page for plugin-specific options.

-v LOG_LEVEL

LOG_LEVEL can be one of DEBUG, INFO, ERROR, CRITICAL or QUIET (no output). **Default is ERROR**

-l LOGFILE

LOGFILE is the path to the log file for status messages. **Default is stdout if not specified and –v is NOT QUIET**. The syslog facility is used if LOGFILE is exactly "syslog".

-P THR_COUNT

THR_COUNT is the number of event threads available for **updaters and setting up connections**

Sampler Configuration Considerations

Sandia National Laboratories

OGC

- Set Memory – default is 512kB
  - Typically this is large enough but use ldms_ls with the –v flag to check what is being used
    - Can save a small amount by reducing to be in alignment with what is actually being used
  - May need to increase if using large vectors of sets or sets with large numbers of metrics
  - Rough rule of thumb: (number of metrics x 8) + 5x overhead for scalars and 0.25 x overhead for vectors
  - **ENOMEM (ERRNO 12) error in log file if set memory capacity is exceeded**
- Update interval
  - Depends on what makes sense for each data set being collected
    - Examples: power-0.1sec, meminfo-1sec, link_status-60sec
- Update offset
  - Typically set to 0 for sampler
  - **If not set will** be some based on when the sampler started but will gradually **drift**
    - This will be changed to default to 0 but currently should always be set to something

Aggregator Considerations

- Memory need assessment (See **Exercise 1.7**)
  - Should be the sum of sampler sets with some headroom for local sets and possible addition of remote sets
  - Should take into account possible failover configuration and additional associated load
- ulimit –n (See **Exercise 1.8**)
  - Should be ~2x number of connections + number of stores + failover + some headroom
  - Some defaults are set to 1024
- Fan-in considerations (See **Exercise 1.9**)
  - Number of cores available on aggregator host (run one event thread per core)
  - Number of sets per sampler ldmsd
  - Number of metrics per set
  - Transport (RDMA is generally more efficient than sock)
  - Failover capacity
  - Re-connect interval and expected number of down nodes vs. I/O thread pool size

ZAP Transport Components and Tuning

Sandia National Laboratories

OGC

- **ZAP_IO_MAX sets the maximum number of IO threads that can be brought up (up to NUM_PROCS (including if hyperthreading)). These handle update completions and storage**
- **ZAP_IO_BUSY sets the busy threshold for spinning up a new I/O thread (float between 0 and 1.0 default=0.80)**
- **ZAP_EVENT_QDEPTH [2048]** – Queue depth in event threads. Each event worker has an event queue. This queue has the depth specified by this environment variable. When this queue depth is reached, attempts to queue additional events will block. This is to avoid having sender/receiver overrun the daemons ability to process events and the queue depth growing unbounded.
- **ZAP_UGNI_CQ_DEPTH [2048]** – Completion queue depth for Cray ugni transport. For high connection counts, this depth should be increased to avoid overrunning the CQ.
- Transport type
  - rdma – Remote Direct Memory Access (RDMA)
  - sock -- Socket
  - ugni – Cray ugni RDMA
  - fabric – libfabric RDMA

Start of Hands-On Exercises

- Set up two terminals and change directory (cd) to ~/ldmscon2021/advanced/exercises/ldms in each
- Set up your environment

```
$ source env/ldms-env.sh
```

- Check your LDMS environment

```
$ ldmsd-check-env #dump currently set environment variables that may influence ldmsd and plugin behavior
```

- HOSTNAME=node-64
- LD_LIBRARY_PATH=/opt/ovis/lib/:/opt/ovis/lib64/:/opt/gcc/10.2.0/snos/lib64
- LDMSD_PLUGIN_LIBPATH=/opt/ovis/lib64/ovis-ldms
- PYTHONPATH=/opt/ovis/lib/python3.6/site-packages
- ZAP_LIBPATH=/opt/ovis/lib64/ovis-ldms

Sandia National Laboratories

OGC

Exercise 1.1 – 1.4: Specifying a pid File

Exercise 1.1: Specifying a pid file

Scripts for running these exercises can be found in:

/home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/E1.1

```
$ ldmsd -x sock:10001 -r /tmp/run/ldmsd.pid
$ ps auxw | grep ldmsd
```

user1 **21116** 0.0 0.0 123336 1620 ? Ssl 10:34 0:00 **ldmsd** -x sock:10001 -r /tmp/run/ldmsd.pid

```
$ ls -l /tmp/run/
```

total 8
-rw------- 1 user1 user1 6 Oct 20 10:34 ldmsd.pid
-rw------- 1 user1 user1 168 Oct 20 10:34 ldmsd.pid.version

```
$ cat /tmp/run/ldmsd.pid
```

21116

```
$ cat /tmp/run/ldmsd.pid.version
```

Started LDMS Daemon with authentication version 4.3.4-alpha.1.24-46b6. LDMSD Interface Version 3.2.2.0. LDMS Protocol Version 4.2.0.0. git-SHA 46b691565fa9cfa3a890fcae7fc0b5b00f6b983c

Exercise 1.2: Using –B0 version-file-mode

Scripts for running exercises 1.2 can be found in:

/home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/E1.2

- In the previous example ldmsd was run without the –B option which uses the default of –B 1
- Kill your ldmsd from the previous exercise

```
$ kill 21116 # or pkill ldmsd
```

- Check to see that both the pid and pid.version files are deleted

```
$ ls -ltr /tmp/run
total 0
```

- Run ldmsd again using –B 0 to suppress banner file creation

```
$ ldmsd -x sock:10001 –B 0 -r /tmp/run/ldmsd.pid
$ ls -ltr /tmp/run/
total 4
-rw------- 1 user1 user1 6 Oct 20 15:26 ldmsd.pid
```

Exercise 1.3: Using –B2 version-file-mode

- Kill your previous ldmsd and then run a new ldmsd using –B 2 to keep the version file when ldmsd is killed

```
$ pkill ldmsd
$ ldmsd -x sock:10001 –B 2 -r /tmp/run/ldmsd.pid
$ ls -ltr /tmp/run/
total 8
-rw------- 1 user1 user1 168 Oct 20 15:36 ldmsd.pid.version
-rw------- 1 user1 user1   6 Oct 20 15:36 ldmsd.pid
```

- Kill your ldmsd and then check for your version file

```
$ pkill ldmsd
$ ls -ltr /tmp/run/
total 4
-rw------- 1 user1 user1 168 Oct 20 15:36 ldmsd.pid.version
```

Note: This file will be overwritten if you re-run your previous ldmsd command so move to preserve

Exercise 1.4: ldmsd -V

```
$ ldmsd -V
```

LDMSD Version: 4.3.4-alpha.1.24-46b6

LDMS Protocol Version: 4.2.0.0

LDMSD Plugin Interface Version: 3.2.2.0

git-SHA: 46b691565fa9cfa3a890fcae7fc0b5b00f6b983c

For comparison:

```
$ cat /tmp/run/my.pid.version
```

Started LDMS Daemon with authentication version 4.3.4-alpha.1.24-46b6. LDMSD Interface Version 3.2.2.0. LDMS Protocol Version 4.2.0.0. git-SHA 46b691565fa9cfa3a890fcae7fc0b5b00f6b983c

Exercise 1.6: Aggregating Using Push, Pull, Auto-interval

Exercise 1.6: Aggregator Using Data Push, Pull, Auto-Interval

/home/<user>/ldmscon2021/advanced/exercises/ldms/conf/**E1.6/agg11_push.conf**

```
prdcr_add name=prdcr1_1 host=<node> type=active xprt=sock port=10001 interval=20000000
prdcr_start name=prdcr1_1
prdcr_add name=prdcr2_1 host=<node> type=active xprt=sock port=10002 interval=20000000
prdcr_start name=prdcr2_1

updtr_add name=updtr1 interval=10000000 offset=100000
updtr_prdcr_add name=updtr1 regex=prdcr1.*
updtr_start name=updtr1

updtr_add name=updtr2 interval=10000000 push=onchange auto_interval=false
updtr_prdcr_add name=updtr2 regex=.*
updtr_match_add name=updtr2 match=inst regex=.*/meminfo
updtr_start name=updtr2

updtr_add name=updtr3 interval=10000000 auto_interval=true          #(Honor "hints" if true)
updtr_prdcr_add name=updtr3 regex=prdcr2.*
updtr_match_add name=updtr3 match=schema regex=.*vmstat.*
updtr_start name=updtr3
```

Note: interval=<some interval> must be specified to even when **registering for push mode** but has no effect on data collection

- Push/Pull/auto_interval **configuration** is performed on the aggregator only
- Can mix push and pull within a single aggregator.

Exercise 1.6: Aggregator Using Data Push Cont.

Sandia National Laboratories

OGC

- Run two ldmsd using ports 10001 and 10002 using: /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/**E1.6/start_multi_samplers.sh**
- Run pull aggregator ldmsd using **agg_push_pull.conf:** /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/**E1.6/start_agg11_push_pull.sh**

start_agg11_push_pull.sh

```
#!/bin/bash
ME=$(whoami)
ldmsd -x sock:20001 -c /home/${ME}/ldmscon2021/advanced/exercises/ldms/conf/E1.5/agg11_push_pull.conf \
        –v DEBUG -l /home/${ME}/ldmscon2021/advanced/exercises/ldms/logs/agg11_push_pull.log \
        -r /tmp/run/agg11_push_pull.pid
```

Exercise 1.6: Aggregator Using Data Push

- Verify, using ldms_ls that data is being pushed/pulled as defined in **agg11_push.conf**

$ ldms_ls -h localhost -x sock -p 20001 -v

```
$ ldms_ls -h localhost -x sock -p 20001 -v
Schema          Instance                  Flags  Msize  Dsize  UID    GID    Perm       Update            Duration          Info
--------------- ------------------------ ------ ------ ------ ------ ------ ---------- ----------------- ----------------- --------
vmstat          node-2_10002/vmstat          CR   6536   1016   1005   1005 -rwxrwxrwx 1596712615.004823          0.000208 "updt_hint_us"="5000000:100000"
meminfo         node-2_10002/meminfo         CR   2496    440   1005   1005 -rwxrwxrwx 1596712620.004248          0.002316 "updt_hint_us"="1000000:0"
vmstat          node-2_10001/vmstat          CR   6536   1016   1005   1005 -rwxrwxrwx 1596712610.002842          0.000200 "updt_hint_us"="10000000:100000"
meminfo         node-2_10001/meminfo         CR   2496    440   1005   1005 -rwxrwxrwx 1596712620.003005          0.001364 "updt_hint_us"="10000000:100000"
--------------- ------------------------ ------ ------ ------ ------ ------ ---------- ----------------- ----------------- --------
Total Sets: 4, Meta Data (kB): 18.06, Data (kB) 2.91, Memory (kB): 20.98
```

Sandia National Laboratories

OGC

Exercise 1.7: Too Little Memory

Exercise 1.7: Too Little Memory (Sampler, Aggregator)

- This exercise explores what happens (failure modes) when the amount of memory allocated to a daemon for hosting metric sets is set too low
- cd to /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/**E1.7** for this exercise
- Examine the sampler configuration file for this exercise and note that the amount of memory being allocated for hosting metric sets on the sampler is set to 10k

$ cat start_ldms_e1.7_sampler.sh

- Run this script and note that the sampler daemon fails to start

$ ./start_ldms_e1.7_sampler.sh

- cat the log file to find the problem **(insufficient memory to load all plugins results in daemon failure)**
- Modify the memory allocation (try 16k) in the start script and try again

$ ./start_ldms_e1.7_sampler.sh

- Success! Note, using ldms_ls –v that the memory required is ~11k

Too Little Memory (Sampler, Aggregator)

- Examine the aggregator configuration file for this exercise and note that the amount of memory being allocated for hosting metric sets on the aggregator is also set to 10k

$ cat start_ldms_e1.7_sampler.sh

- Run this script and see that the aggregator daemon successfully starts

$ ./start_ldms_e1.7_sampler.sh

$ ps axuw | grep ldmsd | grep 20001

- Now check that the aggregator is indeed acquiring the metric sets data

$ ldms_ls –h localhost –x sock –p 20001 –v

- Note that the aggregator is only hosting the meminfo metric set
- cat the log file to find the problem **(insufficient memory to aggregate all sets from producers does NOT result in daemon failure but does result in logging errors along with the likely fix) Note that if your verbosity were set to CRITICAL you would not see this error**
- Modify the memory allocation (try 16k) in the start script and try again

$ ./start_ldms_e1.7_sampler.sh

- Success! Note, using ldms_ls –v that the memory required is ~11k

Too Little Memory (Sampler, Aggregator)

- Modify the memory allocation (try 16k) in the start script and try again

$ ./start_ldms_e1.7_aggregator.sh

- Success! Note, using ldms_ls –v that the memory required is ~11k

- Memory need assessment
  - For a daemon only hosting sampler plugins this should be the sum of sampler sets with some headroom in case you want to add some additional sampler plugins
  - For aggregators this should be the sum of metric sets being aggregated with some headroom in case additional samplers are added to remote sampler daemons
  - For aggregators this should also take into account possible failover configurations and the possible additional associated load

Sandia National Laboratories

OGC

Exercise 1.8: Open File Descriptor Limits

Exercise 1.8: Aggregator Open File Descriptor Limit Set Too Low

- When the ratio of sampler daemons to aggregators is large the upper bound on the number of open file descriptors may need to be modified to accommodate all of the connections using ulimit –n <num>
  - num should be ~2x number of connections + number of stores + failover + some headroom
  - Some defaults are set to 1024
- This exercise explores what happens when ulimit is set too low for the number of connections between an aggregator and samplers
- Check your current ulimit –n

$ ulimit –n

1024

Exercise 1.8: Aggregator Open File Descriptor Limit Set Too Low (Best Practices)

- Start 10 sampler daemons

$ /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/**E1.8/start_multi_samplers.sh**

- Start your aggregator to connect to these sampler daemons

$ /home/<user>/ldmscon2021/advanced/exercises/ldms/scripts/**E1.8/start_multi_samplers.sh**

Sandia National Laboratories

OGC

Exercise 1.9: Fan-in & Reconnect Intervals

Exercise 1.9: Fan-in, Reconnect Intervals

Sandia National Laboratories

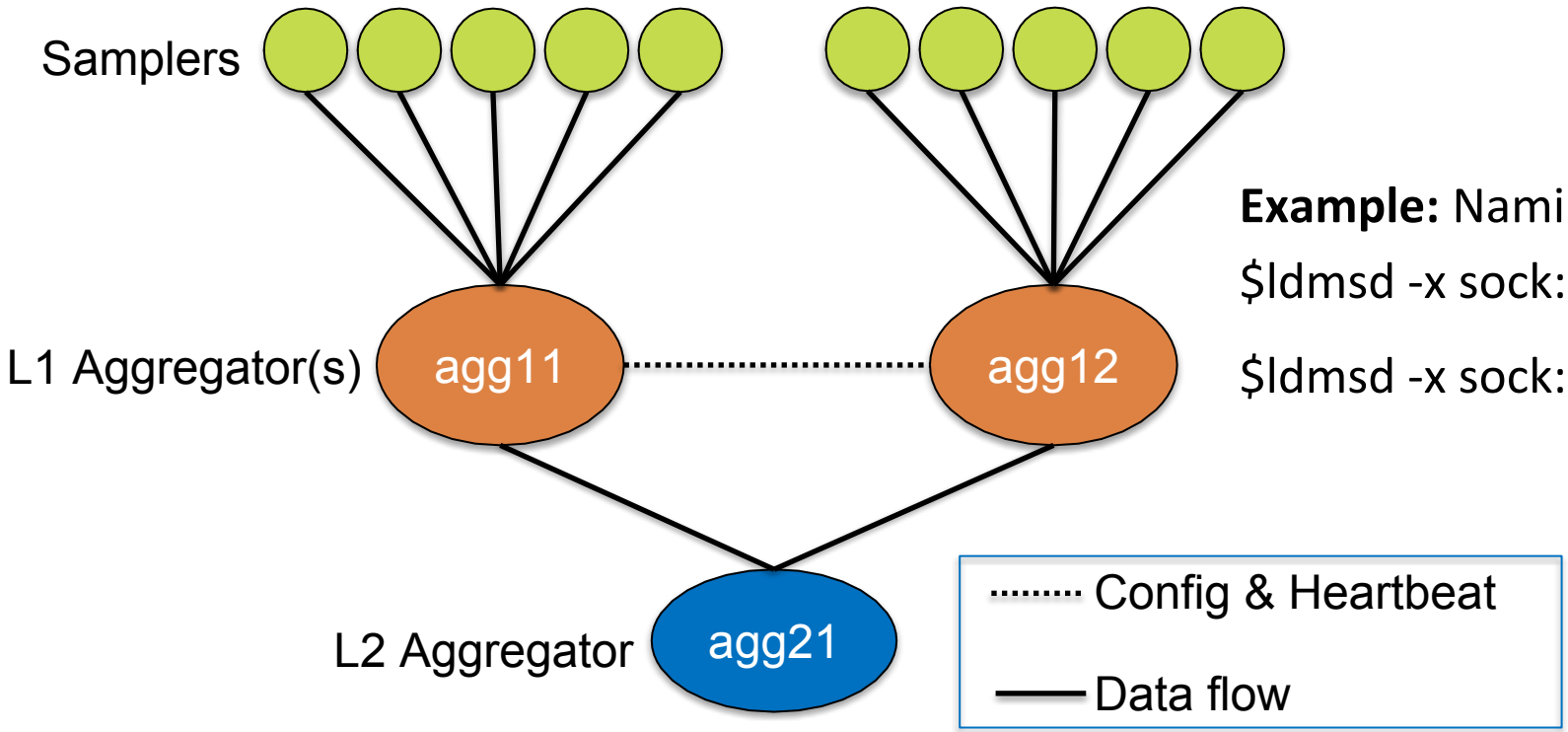
OGC

- Fan-in considerations
  - Number of cores available on aggregator host (run one event thread per core)
  - Number of sets per sampler ldmsd
  - Number of metrics per set
  - Transport (RDMA is generally more efficient than sock)
  - Failover capacity
  - Re-connect interval and expected number of down nodes vs. thread pool size

Exercise 1.10: Aggregator Failover Pairs

Exercise 1.10: Aggregator Failover Pairs

Naming your daemons: The reason for naming a ldmsd is for ease of reference when setting up aggregator "failover pairs". We typically refer to aggregators by level and number though the names are just strings that we use for bookkeeping. In this diagram we refer to the aggregators directly communicating with the "samplers" as level 1 (or L1) aggregators. We typically use the notation **aggxy** where x refers to the level and y refers to the instance at level x (e.g., agg12 would refer to the second L1 aggregator).



Example: Naming daemons

$ldmsd -x sock:10001 **-c agg11.conf** -v ERROR **-n agg11**

$ldmsd -x sock:10002 **-c agg12.conf** -v ERROR **-n agg12**

Exercise 1.10: Two Aggregators Configured as Failover Pairs

Sandia National Laboratories

OGC

Run multiple (10) ldmsd using ports 11001 through 11010 using /home/<user>/exercises/ldms/scripts/**E1.10/start_multi_samplers.sh** script.

Make a configuration file (use ~/ldmscon2021/advanced/exercises/ldms/conf/**E1.10/agg11_failover.conf**) to aggregate from five samplers

==========

env MYHOST=$(eval hostname)
prdcr_add name=prdcr1 host=${MYHOST} type=active xprt=sock port=10001 interval=20000000
prdcr_start name=**prdcr1**

…

prdcr_add name=prdcr5 host=${MYHOST} type=active xprt=sock port=10005 interval=20000000
prdcr_start name=**prdcr5**

updtr_add name=updtr1 interval=1000000 offset=100000
updtr_prdcr_add name=updtr1 regex=prdcr1_.*
updtr_start name=updtr1

failover_config host=${MYHOST} port=20002 xprt=sock interval=1000000 peer_name=**agg12** timeout_factor=2

failover_start

==========

Note: ${MYHOST} is used here because the exercises are all on the same host. **This would be the agg12 host**

Two Aggregators Configured as Failover Pairs Cont.

Make a configuration file (use ~/ldmscon2021/advanced/exercises/ldms/conf/**E1.9/agg12_failover.conf**) to aggregate from another five samplers:

==========

```
env MYHOST=$(eval hostname)
prdcr_add name=prdcr6 host=${MYHOST} type=active xprt=sock port=10006 interval=20000000
prdcr_start name=prdcr6
…
prdcr_add name=prdcr10 host=${MYHOST} type=active xprt=sock port=10010 interval=20000000
prdcr_start name=prdcr10

updtr_add name=updtr1 interval=1000000 offset=100000
updtr_prdcr_add name=updtr1 regex=prdcr1_.*
updtr_start name=updtr1

failover_config host=${MYHOST} port=20001 xprt=sock interval=1000000 peer_name=agg11 timeout_factor=2

failover_start
```

==========

Note: ${MYHOST} is used here because the exercises are all on the same host. **This would be the agg11 host.**

Two Aggregators Configured as Failover Pairs Cont.

Sandia National Laboratories

OGC

/home/<user>/ldmscon2021/advanced/ exercises/ldms/scripts/**E1.9/start_agg11_failover.sh**

```
#!/bin/bash
ME=$(whoami)
ldmsd -x sock:20001 -c /home/${ME}/ldmscon2021/advanced/exercises/ldms/conf/E1.9/agg11_failover.conf \
        –v DEBUG -l /home/${ME}/ldmscon2021/advanced/exercises/ldms/logs/agg11_failover.log \
        -n agg11 -r /tmp/run/agg11_failover.pid
```

/home/<user>/exercises/ldms/scripts/**E1.9/start_agg12_failover.sh**

```
#!/bin/bash
ME=$(whoami)
ldmsd -x sock:20002 -c /home/${ME}/ldmscon2021/advanced/exercises/ldms/conf/E1.9/agg12_failover.conf \
        –v DEBUG -l /home/${ME}/ldmscon2021/advanced/exercises/ldms/logs/agg12_failover.log \
        -n agg12 -r /tmp/run/agg12_failover.pid
```

Two Aggregators Configured as Failover Pairs Cont.

- Run failover aggregator ldmsds using **agg11_failover.conf and agg12_failover.conf**

```
$ ~/ldmscon2021/advanced/exercises/ldms/scripts/E1.9/start_agg11_failover.sh
$ ~/ldmscon2021/advanced/exercises/ldms/scripts/E1.9/start_agg12_failover.sh
```

- Verify that agg11 is aggregating from samplers 1-5 and agg12 is aggregating from samplers 6-10

```
$ ldms_ls –h localhost –x sock –p 20001 –v
$ ldms_ls –h localhost –x sock –p 20002 –v
```

- Kill agg11

```
$ ~/ldmscon2021/advanced/exercises/ldms/scripts/E1.9/stop_agg11_failover.sh
```

- Verify that agg12 is now aggregating from samplers 1-10

```
$ ldms_ls –h localhost –x sock –p 20002 –v
```

- Restart agg11 and verify that agg11 & agg12 are again aggregating from samplers 1-5 & 6-10 respectively

```
$ ldms_ls –h localhost –x sock –p 20001 –v
$ ldms_ls –h localhost –x sock –p 20002 –v
```

Sandia National Laboratories

OGC

Exercise 1.11: Maestro

Sandia National Laboratories

OGC

Exercise 1.12: Vector of Sets

Exercise 1.12: Vector (Buffer) of Sets Sampler Configuration

~/ldmscon2021/advanced/exercises/ldms/conf/**E1.12/vector_sampler.conf**

==========

```
env MYHOST=$(eval hostname)

load name=meminfo
config name=meminfo producer=${MYHOST} instance=${MYHOST}/meminfo schema=meminfo \
set_array_card=10

start name=meminfo interval=1000000 offset=0
```

==========

Note: set_array_card=n configures a ring buffer of size (n x set data size)

Vector of Sets Sampler Configuration

~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/start_vector_sampler.sh**

==========

```
#!/bin/bash
ME=$(whoami)
ldmsd  -x sock:10001 –v DEBUG –l /home/${ME}/ldmscon2021/advanced/exercises/ldms/logs/sampler_vector.log \
            -c /home/${ME}/ldmscon2021/advanced/exercises/ldms/conf/E1.12/sampler_vector.conf \
            -r /home/${ME}/ldmscon2021/advanced/exercises/ldms/run/sampler_vector.pid
```

==========

~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/stop_vector_sampler.sh**

==========

```
#!/bin/bash
ME=$(whoami)
kill $(cat /home/${ME}/exercises/ldms/run/sampler_vector.pid)
```

==========

Vector of Sets Aggregator Configuration

Sandia National Laboratories

OGC

~/ldmscon2021/advanced/exercises/ldms/conf/**E1.12/vector_agg.conf**

==========

```
env MYHOST=$(eval hostname)
env ME=$(whoami)

prdcr_add name=prdcr1 host=${MYHOST} xprt=sock port=10001 type=active interval=20000000
prdcr_start name=prdcr1

# 10 sec update interval of newly populated sets in ring
updtr_add name=updtr interval=10000000 offset=200000
updtr_prdcr_add name=updtr regex=.*
updtr_start name=updtr

load name=store_csv
config name=store_csv path=/home/${ME}/ldmscon2020/advanced/exercises/ldms/data/CSV buffer=0

strgp_add name=meminfo_vector-store_csv plugin=store_csv container=meminfo schema=meminfo
strgp_prdcr_add name=meminfo_vector-store_csv regex=.*
strgp_start name=meminfo_vector-store_csv
```

==========

Vector of Sets Aggregator Start/Stop Scripts

Sandia National Laboratories

OGC

~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/start_vector_agg.sh**

```
#!/bin/bash
ME=$(whoami)
ldmsd  -x sock:20001 –v DEBUG \
        –l /home/${ME}/ldmscon2021/advanced/exercises/ldms/logs/vector_agg.log \
        -c /home/${ME}/ldmscon2021/advanced/exercises/ldms/conf/E1.12/vector_agg.conf \
        -r /tmp/run/vector_agg.pid
```

~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/stop_vector_aggregator.sh**

```
#!/bin/bash
ME=$(whoami)
kill $(cat /tmp/run/vector_agg.pid)
```

Run Vector of Sets Aggregator

- Run vector sampler and aggregator

  ~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/start_vector_sampler.sh**

  ~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/start_vector_agg.sh**

- Tail the store file and see that data comes in chunks of 10 while a ldms_ls of the aggregator only shows one instance

  $ tail -f ~/ldmscon2021/advanced/exercises/ldms/data/CSV/meminfo/meminfo_vector

- Stop vector sampler and aggregator

  ~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/stop_vector_sampler.sh**

  ~/ldmscon2021/advanced/exercises/ldms/scripts/**E1.12/stop_vector_agg.sh**

Sandia National Laboratories

OGC

END Advanced Part 1