Exceptional service in the national interest

OGC | Open Grid Computing, Austin, TX

LDMS Version 4.3 Tutorial Part 1: Basics
https://github.com/ovis-hpc/ovis

Jim Brandt, Ann Gentile, Ben Allan

Sandia National Laboratories

Tom Tucker

Open Grid Computing, Inc.

SAND2017-5153 O

04/2017

Advance Set-up (site specific: wifi UCF_Guest)

We will be using virtual machines hosted at Open Grid Computing

```
$ ssh user<#>@ldmscon.ogc.us
$ password: user<#>
$ ssh user#@compute<#>
$ password: user<#>
```

Schedule

- **9:00 – 10:30 Instruction (Basics)**
- **10:30 – 10:45 Break**
- **10:45 – 12:30 Instruction (Basics)**
- 12:30 – 1:30 Lunch
- 1:30 – 3:00 Instruction
- 3:00 – 3:15 Break
- 3:15 – 5:00 Instruction

Tutorial Format (Basic)

Overview of the Lightweight Distributed Metric Service (LDMS) (9 slides)

- Overview of the LDMS framework
- LDMS architecture description

Setup (3 slides)

- Environment setup description and verification
- Introduction to support programs and helper scripts for use in lab work

Hands-on exercises, instructor walk through, and facilitated student exploration:

- **Exercise 1:** Memeater (1 slide)

Configuring and deploying a distributed monitoring system with storage

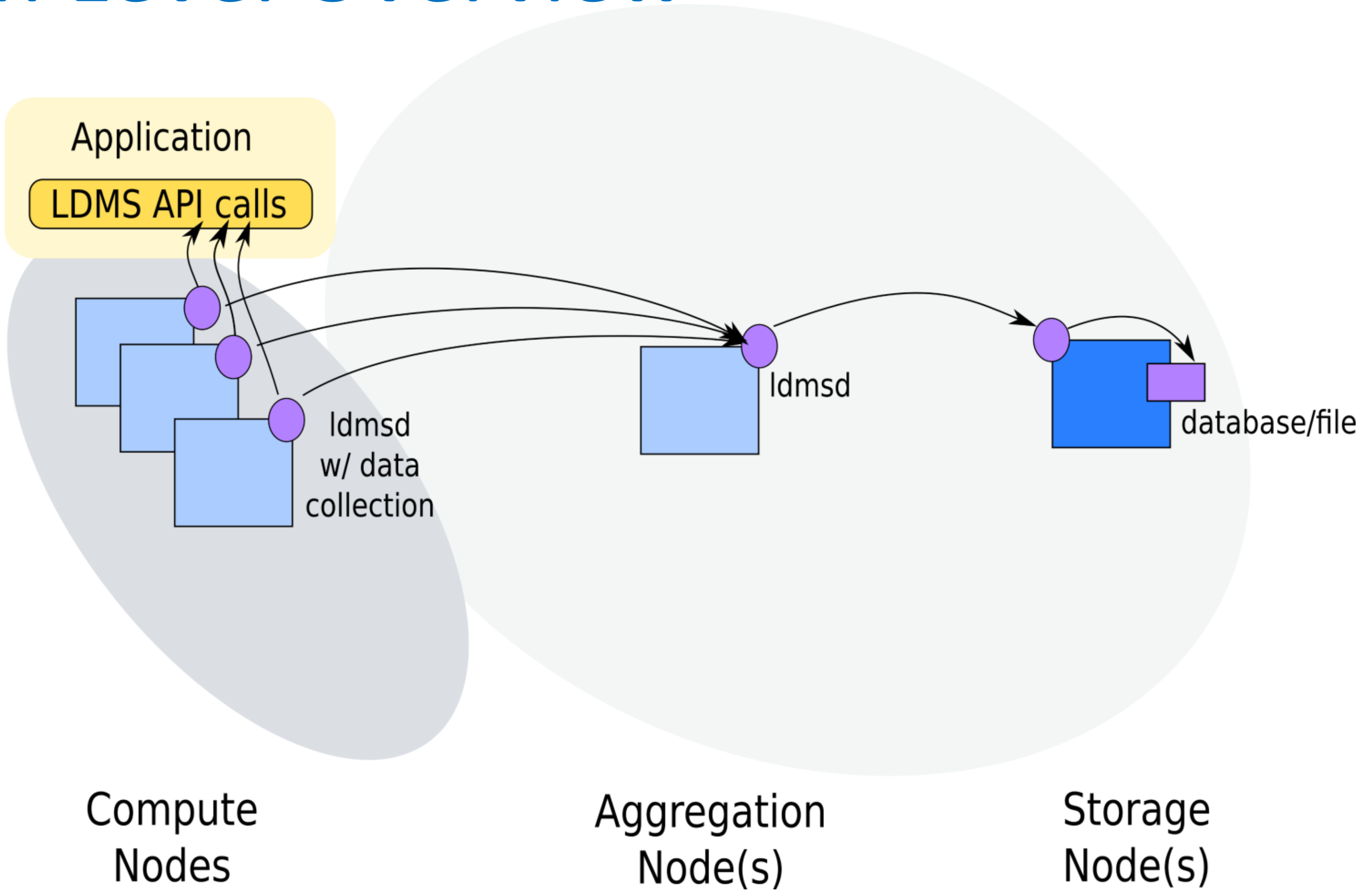
- **Exercise 2:** Configuring and Running Samplers (37 slides – ~1 hour)
  - Sampler startup and local and remote verification
  - Intro to ldmsd_controller and ldms_ls
- **Exercise 3:** Configure Aggregators (13 slides – ~30 min)
  - Aggregation startup and verification using local samplers
  - Aggregation of all other attendees' (remote) samplers
- **Exercise 4:** Aggregating From Remote Hosts: Building a Distributed Monitoring System (4 slides – ~45 min)
- **Exercise 5:** Dynamic Configurations and Resilience (4 slides – ~20 min)
- **Exercise 6:** Storing Data In CSV Format (8 slides – ~20 min)

LDMS Overview

- What is the Lightweight Distributed Metric Service (LDMS)?
  - Daemon based data sampling
    - Collect numeric data
  - Move and aggregate data
  - Store data
  - Analyze data
    - Troubleshooting
    - Optimization
    - Inform future designs
- Typical use cases
  - Identify applications memory (and other resource) utilization behaviors
  - Identify network congestion
  - Determine over-provisioned resources
  - Identify heavy Lustre users

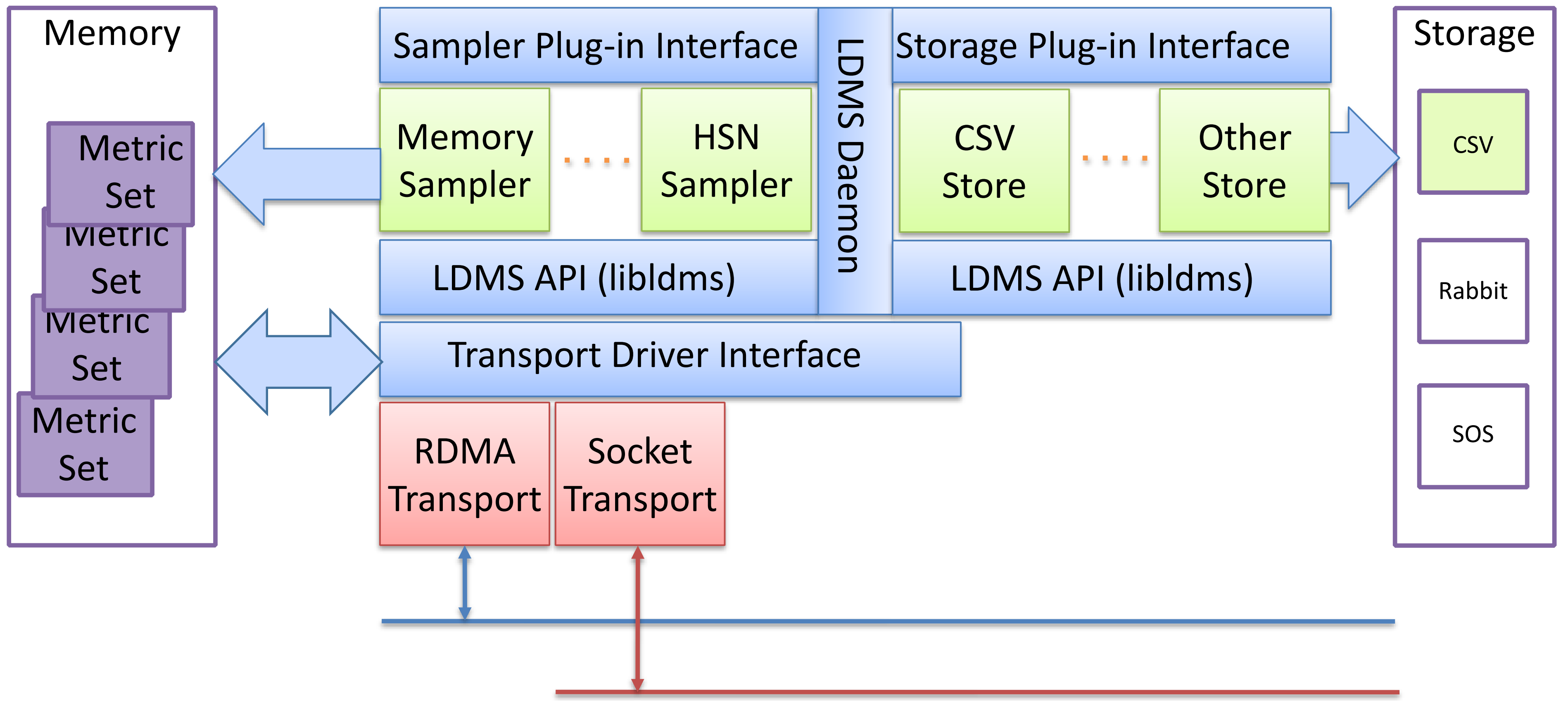
Feature	Generic Monitoring Systems (e.g., Nagios, Ganglia)	Vendor Specific Monitoring Systems (e.g., Cray SEDC)	Application Profilers (e.g., CrayPat, OpenSpeedShop)	**LDMS**
Scalability	Low	Medium	Low	**High**
Typical sampling frequency	Low	Low	High	**Medium**
Overhead	Medium	Low	High	**Low**
Ease of adding new base collectors	High	Low	Medium	**High**
Ease of using data in generic analysis tools	High	Low	Low	**High**
Suitability of data for researchers (e.g., system performance analysis, resilience)	Low	Low	Low	**High**
Suitability of data for use in dynamic feedback (e.g., application or system software can access run-time data for reconfiguration)	Low	Medium	Low	**High**
Portable	Yes	No	Yes	**Yes**
Ability to provide coherent system snapshots	No	No	No	**Yes**
Ability to provide information obtained under production conditions	Yes	Yes	No	**Yes**
Can provide run-time information to platform users	Yes	Yes	No	**Yes**

Lightweight Distributed Metric Service (LDMS) High Level Overview

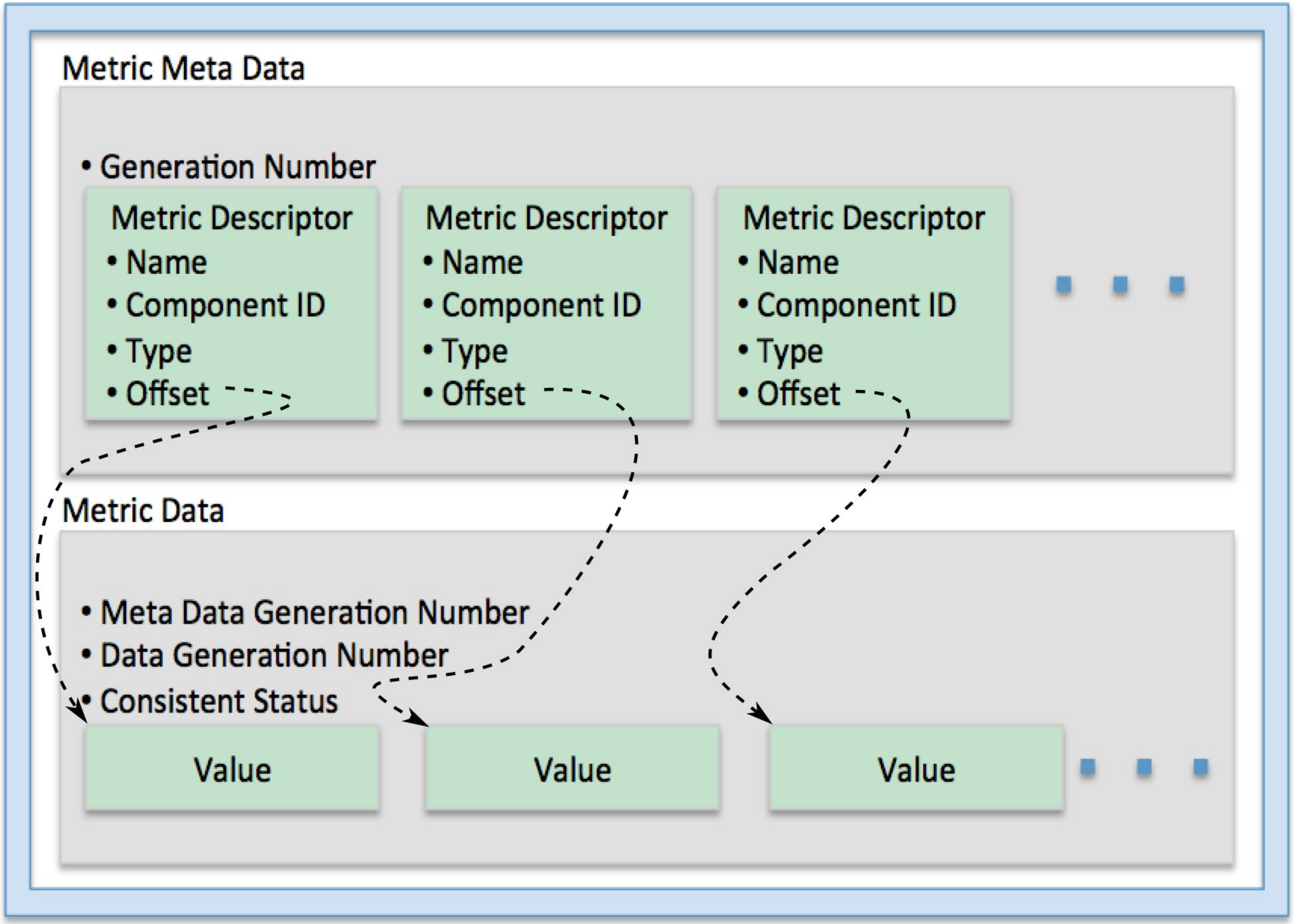


* Only the current data is retained on-node

LDMS Plugin Architecture



Metric Set Memory



Resources

- Documentation (Building, Using)
  - https://github.com/ovis-hpc/ovis/wiki
- Source Code
  - https://github.com/ovis-hpc/ovis
  - git clone https://github.com/ovis-hpc/ovis.git
  - git branch –a # Will show all available branches
  - git branch -a | grep "\-4.3" # Will show all version 4.3 branches
  - git checkout –b OVIS-4.3.<x> origin/OVIS-4.3.<x> # Will check out branch origin/OVIS-4.3.<x> under the name OVIS-4.3.<x>
  - git branch # Will show currently checked out branch
- Publications:
  - https://ovis.ca.sandia.gov
- How you can contribute
  - Post an issue at: https://github.com/ovis-hpc/ovis/issues
- Support
  - Bug reporting and questions: Post an issue at: https://github.com/ovis-hpc/ovis/issues
  - Development services: contact tom@ogc.us
  - Support services: contact tom@ogc.us, ldms@sandia.gov

Supported platforms and networks

- Linux support
  - Rhel 6 and 7
  - SLES 11 & 12
  - Ubuntu
- Vendor hardware platforms running supported software
  - Cray XE6, XK and XC
  - Generic Linux clusters
  - IBM P8 & P9 (both big and little endian)
- Transports
  - Socket
  - Cray ugni
    - Aries
    - Gemini
  - RDMA
    - Infiniband
    - iWarp
    - libfabric

Build dependencies

- Typical compute node environment
  - Autoconf >=2.63, automake, libtool (collectively called autotools)
  - OpenSSH-devel
  - libpapi-devel for papi and syspapi samplers
  - libpfm-devel for syspapi sampler
  - libfabric-devel if applicable transport available

- End use hosts (monitor cluster, special aggregation hosts, etc.)
  - Python 3.x
  - Swig 2.0.x
  - Doxygen for documentation
  - Cython needed for SOS
    - Get from pip
  - libcurl & libcurl-devel if using influx_store

LDMS Installation methods

- Manually build and install using autoconf and automake
- Deployment using RPMs

Note1: For this tutorial, LDMS is pre-installed on student VMs in /opt/ovis

Note2: We will be building and installing to local directories and will use the pre-installed software for all other exercises

Setup

Getting started: Log in and set up your environment

```
ssh user<#>@ldmscon.ogc.us
```

```
user<#>@ldmscon.ogc.us password: user<#>
```

```
$ ssh user<#>@compute<#>
```

Note: “/home/<user>/exercises/ldms/env/ldms-env.sh” is used to set up LDMS environment. You may need to create this file first.

You will want at least 2 terminal windows up for the tutorial

VM directory structure

- VMs include source code, scripts and configuration files for every exercise, helper mini-applications for use in the exercises, and supporting visualization tools (e.g., gnuplot).
- Directory structure:

```
/home/<user>/exercises/                 # Location of exercise related directories
/home/<user>/exercises/ldms/conf/       #  Exercise configuration files
/home/<user>/exercises/ldms/ data/      #  LDMS data
/home/<user>/exercises/ldms/ env/       #  Scripts to configure environment variables
/home/<user>/exercises/ldms/ scripts/   #  Helper scripts for deploying LDMS daemons
/home/<user>/exercises/ldms/ code/      #  memeater code
/home/<user>/exercises/ldms/ logs/      #  Place to write log files
/home/<user>/exercises/ldms/ run/       #  symlink to /tmp/run – place to write pid files
```

Getting started: Set up and verify your environment

- Edit environment configuration file (ldms_env.conf)

```
OVIS_HOME=/opt/ovis

#System environment variables
export PATH=${OVIS_HOME}/bin/:${OVIS_HOME}/sbin/:${PATH}
export LD_LIBRARY_PATH=${OVIS_HOME}/lib/:${LD_LIBRARY_PATH}
export PYTHONPATH=${OVIS_HOME}/lib/python3.6/site-packages/:${PYTHONPATH}

#LDMS environment variables
export ZAP_LIBPATH=${OVIS_HOME}/lib64/ovis-ldms
export LDMSD_PLUGIN_LIBPATH=${OVIS_HOME}/lib/ovis-ldms
```

- Source your environment configuration file

```
$ source ldms_env.conf
```

*A live example of these commands can be found here: Verify Environment Variables

Exercise 1: Memeater

Compile Test Code: memeater.c

- Memeater code repeatedly allocs memory.
- Run in conjunction with LDMS to see changes in memory utilization values reported in /proc/meminfo.
- Code is located at: /home/<user>/exercises/ldms/code/**memeater.c**

```
$ cd /home/<user>/exercises/ldms/code/memeater.c
```

Compile with cc: cc –o memeater memeater.c

Sleep between alloc. Change this wrt sampling frequency.

Periodically increase memory allocated

```
  while (1){
    sleep(2);

    temp = (int*) realloc (keep, ((6144*6144)+count)*sizeof(int));
    if (!temp){
      printf( "Cannot realloc\n");
      break;
      /* malloc will return NULL sooner or later, due to lack of memory */
    }
   ...

 }
 printf("sleeping before exiting\n");
 sleep(60);
 free(keep);
 return 0;
```

Sleep before releasing memory

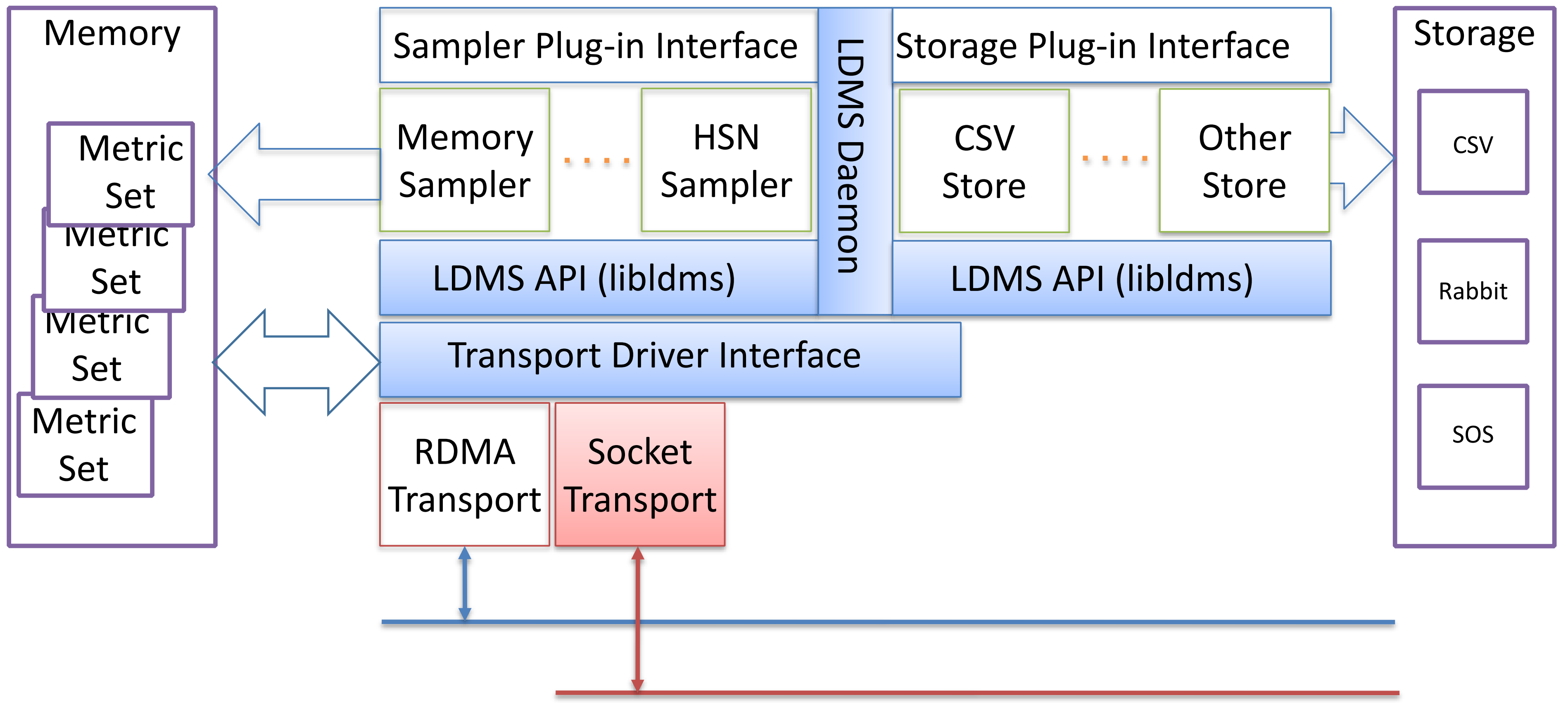
./memeater

```
Active:           231148 kB
alloc: 37748736

adding 1944999541
Active:           378616 kB
alloc: 75497472

  ...

adding 347488691
Active:          1263360 kB
alloc: 301989888

adding 1514442648
adding 1528811800
adding 1877058034
Problems with pipe: Cannot allocate memory
sleeping before exiting
```

OGC Sandia National Laboratories

Exercise 2: Configuring and Running Samplers

LDMS Plugin Architecture



Start and Configure a LDMS Daemon

Exercise Goals:

- Basic LDMS daemon startup and configuration flags/args
  - Manual and run-time configuration options
  - Output options
    - Log files and
  - man pages
    - man ldmsd – displays ldmsd man pages
    - man ldmsd_controller – displays “ldmsd_controller” man pages
- Use of ldms_ls utility as a diagnostic tool
  - man pages
    - man ldms_ls – displays ldms_ls man pages

Start a LDMS daemon

- Start ldmsd with minimum configuration

```
ldmsd -x sock:10001 -l /home/<user>/exercises/ldms/logs/sampler1.log
```

- `-x:` Transport : listening port
- `-l:` Specify the log file path and name(this is not strictly necessary)

NOTES:

- If you receive a “permission denied” error in the “sampler1.log” file, you will need to add “-r ldmsd.pid” at the end of the ldmsd command.
  - -r : The path to the pid file. Please review man page “/ldms/man/ldmsd.man” for more information
- Commands should be **written** in the command prompt window. Copy and paste may cause unnecessary issues with the command line interface

Check ldmsd Running Status

- Using ps

```
ps auxw | grep ldmsd | grep -v grep
```

  - Returns something like:
    “ovis_pu+   3582   0.0   0.1 401604   2204 ?   Ssl 12:51   0:00 ldmsd -x sock:10001“ **if running**
  - Returns: blank line if not running

- Using ldms_ls

```
ldms_ls -h localhost -x sock -p 10001
```

  - Returns: “Connection failed/rejected.” if ldmsd specified does not exist or authentication fails
  - Returns: blank line if the ldmsd specified exists but has no metric sets configured
  - Remote: Replace “localhost” in the command above with another student vm e.g., “ovis-demo-28”

- Also check ports

```
netstat -an | grep 000
```

```
tcp      0      0 0.0.0.0:10001  0.0.0.0:*    LISTEN
```

- Troubleshooting: Also check the log for clues if operation seems wrong.

EXAMPLE: Start and Check LDMS Daemon

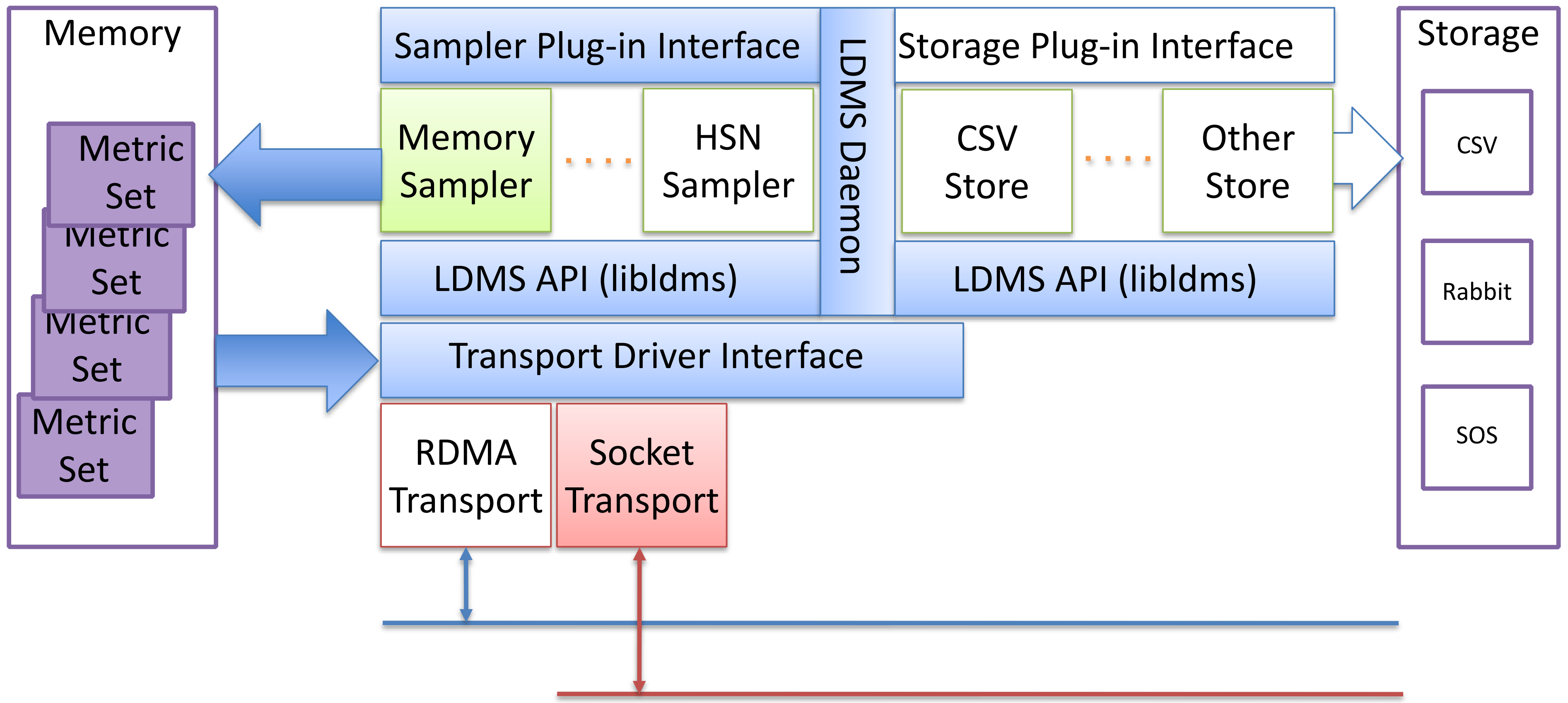
Please see the Start and Check an LDMS daemon to view a live example of these commands (slides 23-24).

Manually Load and Configure a Sampler Plugin

Additional Exercise Goals:

- Basic sampler plugin operation
  - Manual dynamic configuration using the “ldmsd_controller” utility
  - Static configuration using a configuration file
  - man pages
    - man Plugin_meminfo – opens meminfo plugin man pages
    - man Plugin_vmstat – opens vmstat plugin man pages
- Use of ldms_ls utility as a diagnostic tool
  - man pages
    - man ldms_ls – opens ldms_ls man pages

LDMS Plugin Architecture



Configure LDMS Daemon Sampler Plugin

Goals:

- Load the “meminfo” sampler plugin
- Configure loaded “meminfo” sampler plugin
  - Give the set name (instance)
  - Give the node name (producer)
  - Give the component ID
  - Plugin-specific arguments
- Start sampler plugin with a particular sampling interval and offset

optional

Connect ldmsd_controller To An ldmsd

Set up “ldmsd_controller” connection to the aggregator

```
$ldmsd_controller --host localhost --port 10001
```

```
Welcome to the LDMSD control processor
sock:localhost:10001> help
```

See “LDMS HELP” slides starting at slide 77 for help results

- Note 1: The prompt tells you <transport>:<hostname>:<port>
- Note 2: You can use “quit” or Ctrl-d to **exit** or Ctrl-c to **kill** the ldmsd_controller

*A live example of these commands can be found here:
LDMSD Controller Interface Video

Interactive Configuration Using The ldmsd_controller

- Load the "meminfo" sampler plugin:

```
sock:localhost:10001> load name=meminfo
```

- Configure the "meminfo" sampler plugin:

```
sock:localhost:10001> config name=meminfo producer=<$HOSTNAME>
instance=<$HOSTNAME>/meminfo component_id=<host number>
```

- EXAMPLE:

```
sock:localhost:10001> load name=meminfo *enter*
sock:localhost:10001> config name=meminfo producer=ovis-demo-01
instance=ovis-demo-01/meminfo component_id=1 *enter*

producer: Initialize the name of the sampler
Instance: Initialize the name of the node the sampler is running on
component_id: Initialize with a number
```

Query Current Sets On An LDMS Daemon Using “ldms_ls”

- Use ldms_ls to query the current sets available on an LDMS daemon

```
$ ldms_ls -h localhost -x sock -p 10001
```

```
ovis-demo-01/meminfo
```

Get The Set Information Before Starting The "meminfo" Sampler Plugin

```
$ ldms_ls -h localhost -x sock -p 10001 -v ovis-demo-01/meminfo
```

```
Schema         Instance                         Flags  Msize  Dsize  UID    GID   Perm             Update           Duration         Info
-------------- ------------------------         ------ ------ ------ ------ ------ ----------      ---------------- ---------------- --------
meminfo        ovis-demo-01/meminfo                L   1952    416    596  742   -rwxrwxrwx       0.000000         0.000000
"updt_hint_us"="1000000:0"
-------------- ------------------------         ------ ------ ------ ------ ------ ----------      ---------------- ---------------- --------
Total Sets: 1, Meta Data (kB): 1.95, Data (kB) 0.42, Memory (kB): 2.37
```

NOTE: The "`ovis-demo-01/meminfo`" is optional. It is suggested as it will be easier to identify certain sampler daemons when multiple are running on the same host and port.

EXAMPLE: Interactive Configuration Using The ldmsd_controller

Please see Configuration Using LDMSD Controller Interface to view a live example of these commands (slides 30-32).

Query Current Metric Values Before Starting The "meminfo" Sampler Plugin

```
$ ldms_ls -x sock -p 10001 -l ovis-demo-01/meminfo
```

```
ovis-demo-01/meminfo: inconsistent, last update: Wed Dec 31 17:00:00 1969 -0700 [0us]
M u64     component_id        62
D u64     job_id              0
D u64     app_id              0
D u64     MemTotal            0
D u64     MemFree             0
D u64     MemAvailable        0
D u64     Buffers             0
D u64     Cached              0
D u64     SwapCached          0
D u64     Active              0
D u64     Inactive            0
D u64     Active(anon)        0
D u64     Inactive(anon)      0
```

- *Set is "inconsistent"*
- *Values have not yet been collected*

Start The “meminfo” Sampler Plugin

- Start the “meminfo” sampler with a 1 second interval

```
sock:localhost:10001> start name=meminfo interval=1000000
offset=0
```

- This starts the sampler updating the metric values every 1,000,000 micro-seconds = 1 second
- **Note 1:** “offset” defines micro-seconds after the second
- **Note 2:** If offset is not specified the timer starts when the sampler starts

Query Current Metric Values After Starting The "meminfo" Sampler Plugin

```
$ ldms_ls -x sock -p 10001 -l ovis-demo-01/meminfo
```

```
ovis-demo-01/meminfo: consistent, last update: Tue Oct 08 17:52:45 2019 -0600 [2058us]
M u64    component_id          62
D u64    job_id                0
D u64    app_id                0
D u64    MemTotal              131899768
D u64    MemFree               129843340
D u64    MemAvailable          129364708
D u64    Buffers               20076
D u64    Cached                458024
D u64    SwapCached            0
D u64    Active                184380
D u64    Inactive              393140
D u64    Active(anon)          125324
D u64    Inactive(anon)        284684
```

- *Set is "consistent"*
- *Values have been collected*

Periodically Re-Query Sampler and Run "memeater"

```
$ while true; do ldms_ls -h localhost -x sock -p 10001 -l | grep "Active "; sleep 1; done
```

```
D u64      Active                 192308
D u64      Active                 192308
D u64      Active                 191884
D u64      Active                 192396
D u64      Active                 192444
D u64      Active                 192420
D u64      Active                 192528
D u64      Active                 192516
```

- *Note how the values change without/with "memeater" running*

In a separate terminal window, run the "memeater" executable to see both timestamps and values change:

```
$ /home/<user>/memeater/memeater
```

Note: You can edit and re-compile to change the allocation amounts and sleep time to adjust the rate of change.

Check Source (/proc/meminfo) For Reference

```
$ cat /proc/meminfo
MemTotal:          131899768 kB
MemFree:           129828892 kB
MemAvailable:      129350280 kB
Buffers:           20076 kB
Cached:            458076 kB
SwapCached:        0 kB
Active:            192340 kB
Inactive:          393064 kB
Active(anon):      133212 kB
Inactive(anon):    284680 kB
Active(file):      59128 kB
```

EXAMPLE: “meminfo” Sampler Plugin

Please see Meminfo Sampler Daemon to view a live example of these commands (slides 34-38).

Dynamically Change The Sampling Interval

- Using ldmsd_controller, stop the plugin:

```
sock:localhost:10001> stop name=meminfo
```

Note: Querying with ldms_ls will show that the sampler is not updating

Note: We are still using the same sampler daemon from earlier. It should not be killed yet.

- Restart the plugin with a different (5 sec) interval:

```
sock:localhost:10001> start name=meminfo interval=5000000
offset=0
```

Note: Querying with ldms_ls will show that the metric set is now updating only every five seconds

(More on dynamic configuration and resilience in Exercise 3)

Kill Currently Running Daemons

- Kill all of your ldmsd in preparation for the next section

```
$ killall ldmsd
```

- Kill a particular ldmsd

```
$ ps auxw | grep ldmsd | grep -v grep
$ ovis_pu+  3582  0.0  0.1 401604  2204 ?        Ssl
12:51   0:00 ldmsd -x sock:10001 -S samplerd.sock
$ kill 3582
```

- Check to make sure it is dead

```
$ ps auxw | grep ldmsd | grep -v grep
```

EXAMPLE: Change Sample Interval

Please see Change Sample Interval for Meminfo to view a live example of these commands (slides 40-41).

Start a ldmsd and Sampler Plugin Using a Configuration File

- A ldmsd can be started using a configuration file
  - Syntax is identical to that used for manual configuration
  - Can be used to run and configure BOTH sampler and aggregator ldmsd
- Edit the sample configuration file, as appropriate, for the meminfo example:

```
$ cat /home/<user>/exercises/ldms/conf/simple_sampler.conf
```

- NOTE: If the "simple_sampler.conf" is not there, then please create this file in this directory and populate it with the content below:

```
load name=meminfo
config name=meminfo producer=<$HOSTNAME> instance=<$HOSTNAME>/meminfo
component_id=<host number>
start name=meminfo interval=1000000
```

- Run an ldmsd using this configuration file (argument after the –c flag). Modify <user> to your user name.

```
$ ldmsd -x sock:10001 \
-l /home/<user>/exercises/ldms/logs/sampler1.log \
-c /home/<user>/exercises/ldms/conf/simple_sampler.conf
```

Query The Metric Values: The "meminfo" Sampler Is Configured And Running

```
$ ldms_ls -x sock -p 10001 -l ovis-demo-01/meminfo
```

```
ovis-demo-01/meminfo: consistent, last update: Tue Oct 08 17:52:45 2019 -0600 [2058us]
M u64       component_id            62
D u64      job_id                   0
D u64      app_id                   0
D u64      MemTotal                 131899768
D u64      MemFree                  129843340
D u64      MemAvailable             129364708
D u64      Buffers                  20076
D u64      Cached                   458024
D u64      SwapCached                0
D u64      Active                   184380
D u64      Inactive                 393140
D u64      Active(anon)             125324
D u64      Inactive(anon)           284684
```

- *Set is "consistent"*
- *Values have been collected*

Multiple Sampler Plugins

- Uncomment and edit to reflect your host the lines for the vmstat plugin in simple_sampler.conf and restart the ldmsd.

```
load name=vmstat
config name=vmstat producer=<hostname>
instance=<hostname>/vmstat component_id=<hostnum>
start name=vmstat interval=1000000 offset=0
```

Note: hostname is just a string and hostnum is just a uint_64. Example: hostname=ovis-demo-01, hostnum=1

- Query the ldmsd:

```
ldms_ls -h localhost -x sock -p 10001
ovis-demo-01/vmstat
ovis-demo-01/meminfo
```

EXAMPLE: Multiple Sampler Plugins

Please see Multiple Plugin Sampler Deamon to view a live example of these commands (slides 43-45).

Configuration Tools Summary

Dynamic/manual configuration (remote or local)

- ldmsd_controller – Python script that can connect to a ldmsd via a configured network socket or a local Unix Domain Socket (supports command completion)
- ldmsctl – C-based utility that can connect to a ldmsd via a configured network socket or a local Unix Domain Socket (doesn't support command completion)

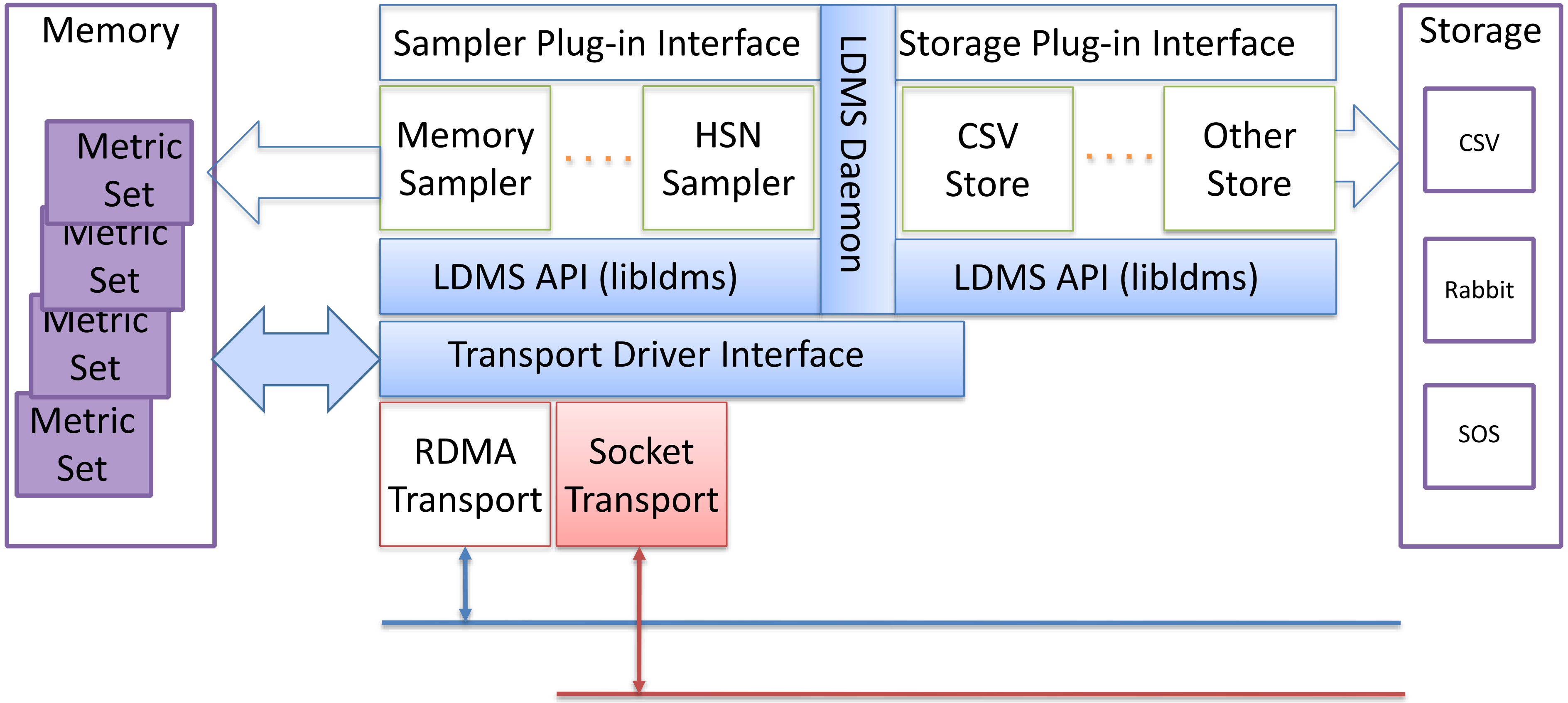
Static configuration (local)

- Configuration file – loaded at ldmsd run time

OGC Sandia National Laboratories

Exercise 3: Configure Aggregators

LDMS Plugin Architecture



Configure a LDMS daemon (ldmsd) to Aggregate metric set(s)

Goals:

- Add list of connections to a ldmsd (connections to sampler ldmsd(s))
- Start the connections
- Create an “update policy”
    - Define an “update policy” update period
    - Define which sets an update policy refers to
- Start the “update policy”

Start a ldmsd That Will Be Used For Aggregation

- (Re)start the sampler ldmsd from the previous exercise (can keep both meminfo and vmstat)
- Start new aggregator ldmsd with minimum configuration:

```
$ ldmsd -x sock:20001 -l /home/<user>/exercises/ldms/logs/agg1.log
```

- `-x:` Transport : listening port
- `-l:` Specify the log file path and name (this is not strictly necessary)

NOTE:

- We will be using a different port number. Instead of 10001 we will be running a daemon on port 20001.
- Please refer to slides 41 & 42 for help in re-creating a sampler daemon

Interactive Aggregator Configuration

- Set up “ldmsd_controller” connection to the aggregator over socket

```
$ ldmsd_controller --host localhost --port 20001
Welcome to the LDMSD control processor
sock:localhost:20002>
```

Simple Aggregator Producer Configuration

- Configure the aggregator to aggregate the “meminfo” set from your sampler daemon (listening on port 10001)

```
sock:localhost:20001> prdcr_add name=prdcr1 host=$HOSTNAME port=10001
xprt=sock type=active interval=20000000

sock:localhost:20001> prdcr_start name=prdcr1
```

- name: policy tag (this is just a string)
- host: hostname for the sampler daemon (e.g. ovis-demo-01)
- port: Listener port of the sampler daemon
- xprt: Transport the sampler daemon listens on
- type: Always “active” (the aggregator will initiate the connection with the sampler)
- interval: Re-connect interval *(not aggregation interval)*

Check Aggregator Status

(**after producer** (prdcr) is started but **before** the **updater** (updtr) is started)

```
sock:localhost:20001> status
```

```
Name         Type         Interval     Offset       Libpath
------------ ------------ ------------ ------------ ------------
Name             Host             Port         Transport    State
---------------- ---------------- ------------ ------------ ------------
prdcr1           nid00062         10001        sock         CONNECTED
    nid00062/meminfo meminfo_x86_ven0000fam0006mod003F START
    nid00062/vmstat  vmstat_x86_ven0000fam0006mod003F START
Name             Interval     Offset       Mode           State
---------------- ------------ ------------ -------------- ------------
Name             Container        Schema           Plugin           State
---------------- ---------------- ---------------- ---------------- ------------
```

Query Current Metric Values On The Aggregator

```
$ ldms_ls -h localhost -x sock -p 20001 -l
$
```

Note: While status (previous slide) shows that the aggregator knows what sets the producer has, the ldms_ls query returns nothing because the updater had not yet been run and the set has not been populated with data.

EXAMPLE: Simple Aggregator Producer Configuration

Please see the simple Aggregator Producer Configuration to view a live example of these commands (slides 51-55).

Simple Aggregator Updater Configuration

OGC Sandia National Laboratories

- Configure the aggregator to update the “meminfo” set

```
sock:localhost:20001> updtr_add name=updtr1 interval=1000000
offset=200000
sock:localhost:20001> updtr_prdcr_add name=updtr1 regex=.*
sock:localhost:20001> updtr_start name=updtr1
```

- `name:` policy tag (string)
- `interval:` update (pull) interval (in usec)
  - Example: interval=1000000 means pull data from sampler every 1 seconds
- `offset:` Target (in us) from <epoc sec>.000000
  - Example: offset=10000 means aggregate every <interval> seconds at 10ms into the second.
- `regex:` regular expression to match the target producers tag(s)

Check Aggregator Status

(after starting both producer (prdcr) and updater (updtr) policies)

```
sock:localhost:20001> status
```

```
Name         Type         Interval     Offset       Libpath
------------ ------------ ------------ ------------ ------------
Name             Host             Port         Transport    State
---------------- ---------------- ------------ ------------ ------------
prdcr1           nid00062         10001        sock         CONNECTED
   nid00062/meminfo meminfo_x86_ven0000fam0006mod003F READY
   nid00062/vmstat  vmstat_x86_ven0000fam0006mod003F READY
Name             Interval     Offset       Mode           State
---------------- ------------ ------------ -------------- ------------
updtr1           1000000      0            Pull           RUNNING
   prdcr1           nid00062         10001 sock       CONNECTED
Name             Container        Schema           Plugin           State
---------------- ---------------- ---------------- ---------------- ------------
```

Query Current Metric Values On The Aggregator

```
$ ldms_ls -h localhost -x sock -p 20001 -l ovis-demo-01/meminfo
```

```
nid00062/meminfo: consistent, last update: Wed Oct 09 18:30:49 2019 -0600 [2093us]
M u64        component_id                    62
D u64       job_id                         0
D u64       app_id                         0
D u64       MemTotal                          131899768
D u64       MemFree                           129834752
D u64       MemAvailable                        129356628
D u64       Buffers                        20228
D u64       Cached                          458892
D u64       SwapCached                          0
D u64       Active                         196708
D u64       Inactive                        393768
D u64       Active(anon)                       137336
```

Check To See That Metrics Change In Both Samplers and Aggregators

- In a third window, run the memeater executable to see changes in the dataset values in both the samplers and aggregators:

```
$ /home/<user>/memeater/memeater
```

- Check sampler using ldms_ls:

```
$  while true; do ldms_ls -h localhost -x sock -p 10001 -l | grep "Active "; sleep 1; done
```

- Check aggregator using ldms_ls:

```
$  while true; do ldms_ls -h localhost -x sock -p 20001 -l | grep "Active "; sleep 1; done
```

EXAMPLE: Simple Aggregator Updater Configuration

Please see Aggregator Updater Configuration to view a live example of these commands (slides 57-60).

Start ldmsd and Aggregation Using a Configuration File

- A ldmsd for performing aggregation can be started using a configuration file in the same manner as a ldmsd for sampling
- Configuration file syntax is identical to that used for manual configuration
- *Edit your* sample configuration file *to reflect your host*:

```
$ cat /home/<user>/exercises/ldms/conf/simple_agg.conf
```

NOTE: If the "simple_agg.conf" is not there, then please create this file in this directory and populate it with the content below:

```
prdcr_add name=prdcr1 host=$HOSTNAME port=10001 xprt=sock type=active interval=20000000
prdcr_start name=prdcr1
updtr_add name=updtr1 interval=1000000 offset=200000
updtr_prdcr_add name=updtr1 regex=.*
updtr_start name=update_all
```

- Kill your aggregator and restart your aggregator ldmsd using this configuration file

```
$ ldmsd -x sock:20001 -l /home/<user>/exercises/ldms/log/aggd.log \
-c /home/<user>/exercises/ldms/conf/simple_agg.conf
```

Query Current Metric Values On The Aggregator

```
$ldms_ls -x sock -p 20001 -l ovis-demo-01/meminfo
```

```
Ovis-demo-01/meminfo: consistent, last update: Wed Oct 09 18:30:49 2019 -0600 [2093us]
M u64      component_id              62
D u64      job_id                    0
D u64      app_id                    0
D u64      MemTotal                  131899768
D u64      MemFree                   129834752
D u64      MemAvailable              129356628
D u64      Buffers                   20228
D u64      Cached                    458892
D u64      SwapCached                0
D u64      Active                    196708
D u64      Inactive                  393768
```

EXAMPLE: Simple Aggregator with Configuration File

Please see Aggregator With Configuration File to view a live example of these commands (slides 62-63).

Exercise 4: Aggregating From Remote Hosts: Building a Distributed Monitoring System

Aggregate From All Student VMs

- Kill aggregator ldmsd
- Edit /home/<user>/exercises/ldms/conf/agg.conf:

```
$ cat /home/<user>/exercises/ldms/conf/agg.conf
#prdcr_add name=prdcr1 type=active host=compute1 port=10001 xprt=sock interval=20000000
#prdcr_add name=prdcr2 type=active host=compute2 port=10001 xprt=sock interval=20000000
#prdcr_add name=prdcr3 type=active host=compute3 port=10001 xprt=sock interval=20000000

prdcr_start_regex regex=.*                                   START (connect to) ALL PRODUCERS

updtr_add name=updtr1 interval=1000000 offset=200000         UPDATE AT 1 SEC INTERVALS
updtr_prdcr_add name=updtr1 regex=.*                         DO THIS ON ALL PRODUCERS
updtr_match_add name=updtr1 match=schema regex=meminfo       RESTRICT TO SETS WITH schema=meminfo
updtr_start name=updtr1                                 START UPDATER POLICY “updtr1”

updtr_add name=updtr2 interval=2000000 offset=200000         UPDATE AT 2 SECOND INTERVALS
updtr_prdcr_add name=updtr2 regex=.*                         DO THIS ON ALL PRODUCERS
updtr_match_add name=updtr2 match=schema regex=vmstat        RESTRICT TO SETS WITH schema=vmstat
updtr_start name=updtr2                                 START UPDATER POLICY “updtr1”
```

Aggregate From All Student VMs (cont'd)

- Restart ldmsd using your edited configuration file

```
$ ldmsd -x sock:20001 -l /home/<user>/exercises/ldms/log/aggd.log \
–c /home/<user>/exercises/ldms/conf/agg.conf
```

NOTE: If the "**agg.conf**" has not yet been populated, then create this file by first copying the "simple_aggregator.conf" and adding the content from the previous slide at the end of the file.

LDMS supports complex topologies:

- Multiple ldmsd (aggregators) can pull from the same ldmsd (sampler or aggregator)
- Can daisy chain aggregators
  - Hierarchical
  - Support both fan-in and fan-out topologies

Check Aggregator Status

```
sock:localhost:20001> status
Name       Type       Interval    Offset     Libpath
------------ ------------ ------------ ------------ ------------
Name            Host           Port        Transport    State
---------------- ---------------- ------------ ------------ ------------
prdcr1          nid00062        10001      sock        CONNECTED
   nid00062/meminfo meminfo_x86_ven0000fam0006mod003F READY
   nid00062/vmstat  vmstat_x86_ven0000fam0006mod003F READY
prdcr2          nid00063        10001      sock        CONNECTED
   nid00063/meminfo meminfo_x86_ven0000fam0006mod003F READY
   nid00063/vmstat  vmstat_x86_ven0000fam0006mod003F READY
prdcr3          nid00012        10001      sock        DISCONNECTED
Name            Interval     Offset       Mode            State
---------------- ------------ ------------ -------------- ------------
updtr1          1000000      200000       Pull            RUNNING
   prdcr1          nid00062              10001 sock        CONNECTED
   prdcr2          nid00063              10001 sock        CONNECTED
   prdcr3          nid00012              10001 sock        DISCONNECTED
Name            Container       Schema          Plugin          State
---------------- ---------------- ---------------- ---------------- ------------
```

Using The Distributed System

- **Exercise** - Loop ldms_ls while running your memeater executable – see your and others data values change

```
$ while true; do ldms_ls -h localhost -x sock -p 20001 -l | grep "Active ” –B9; sleep 1; done
```

Explore basic dynamic configuration changes and resilience in the next exercise

EXAMPLE: Aggregate from Multiple VMs

Please see Aggregate From Multiple VMs to view a live example of these commands (slides 66-69) .

OGC

Sandia National Laboratories

Exercise 5: Basic Dynamic Configurations and Resilience

Basic Dynamic Configuration Changes

- Dynamic configuration
  - Sampler daemons (from exercise 1 slide 40)
    - Stopping sampler plugins
    - Starting sampler plugins with different intervals
  - Aggregator daemons
    - Automatic detection of new metric sets on connected sampler ldmsd
    - Stopping producer (prdcr) and updater (updtr) policies
    - Changing updater intervals

Dynamically Changing a Sampler Plugin's Interval Parameters (also exercise 1 slide 40)

- Using ldmsd_controller, stop the plugin:

```
sock:localhost:10001> stop name=meminfo
```

Note: Querying with ldms_ls will show that the sampler has stopped

- Restart the plugin with a different interval:

```
sock:localhost:10001> start name=meminfo
interval=5000000 offset=0
```

Note: Querying with ldms_ls will show that the metric set is now updating only every five seconds

Dynamic Changes and Aggregator Robustness

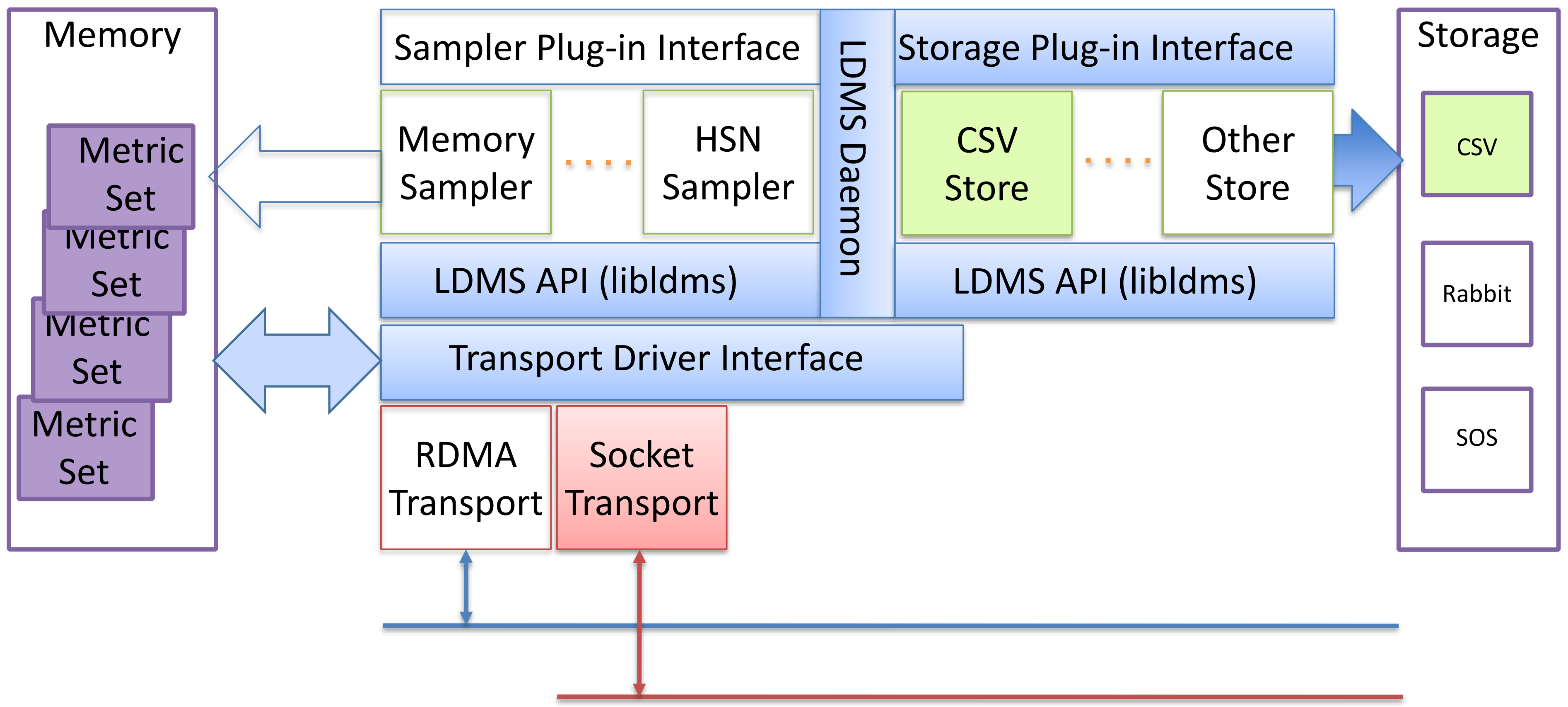
- On-the-fly additions of samplers will be discovered by the aggregating ldmsd
  - **Exercise** – one student will add the vmstat sampler, using ldmsd_controller, to their running sampler ldmsd.
    - All others will see it, using ldms_ls, appear in their aggregators which are pulling from that sampler.
  - **Exercise** – one student will stop their meminfo sampler, using ldmsd_controller, on their running sampler ldmsd.
    - All others will see, using ldms_ls, that the timestamp in that student's metric set ceases to update.
  - **Exercise** – the same student will restart their meminfo sampler, using ldmsd_controller, on their running ldmsd.
    - All others will see, using ldms_ls, that the timestamp in that student's metric set resumes updating.

Dynamic Changes and Robustness (cont'd)

- Samplers and Aggregators can be started in any order
  - **Exercise** – Use your modified configuration files to start the aggregator ldmsd before starting the sampler ldmsd
    - Use ldms_ls to convince yourself that, whether a sampler ldmsd is started before or after an aggregator ldmsd, you are able to see the data generated at the sampler ldmsd on the aggregator ldmsd when both are running
- LDMS collection and transport are robust to Samplers and Aggregators being killed and restarted
  - **Exercise** – one student will kill their sampler ldmsd. All other students will see from ldms_ls timestamp that the student's metric set is removed from the list.
  - **Exercise** – the same student will restart their sampler ldmsd. All other students will see from ldms_ls timestamp that the metric set reappears and resumes updating (after up to the producer reconnect interval of 20 seconds).
  - **Exercise** – Each student will stop and re-start their aggregator ldmsd and verify, using ldms_ls, that they are able to see appropriate data.

Exercise 6: Storing Data In CSV Format

LDMS Plugin Architecture



Storing Data: CSV Store Plugin

- Goals:
  - Configure an aggregator ldmsd with a CSV store plugin using ldmsd_controller
  - Configure an aggregator ldmsd with a CSV store plugin using a configuration file
  - Minimal store options (don’t buffer data)

- Example output from the “meminfo” sampler:

```
#Time,Time_usec,ProducerName,component_id,job_id,MemTotal,MemFree,MemAvailable,Buffers,Cached,SwapCached,Active,Inacti
ve,Active(anon),Inactive(anon),Active(file),Inactive(file),Unevictable,Mlocked,SwapTotal,SwapFree,Dirty,Writeback,AnonPages,Mappe
d,Shmem,Slab,SReclaimable,SUnreclaim,KernelStack,PageTables,NFS_Unstable,Bounce,WritebackTmp,CommitLimit,Committed_AS,V
mallocTotal,VmallocUsed,VmallocChunk,HardwareCorrupted,AnonHugePages,HugePages_Total,HugePages_Free,HugePages_Rsvd,Hu
gePages_Surp,Hugepagesize,DirectMap4k,DirectMap2M

1487105964.002482,2482,ovis-demo-09,9,
0,1884188,571028,1688632,0,1212004,6108,104536,1122496,8276,8580,96260,1113916,0,0,839676,793956,420,0,10552,24812,17
96,52124,40104,12020,1792,3280,0,0,0,1781768,387984,34359738367,7216,34359728128,0,2048,0,0,0,0,2048,47040,2050048

1487105963.002583,2583,ovis-demo-02,2,
0,1884188,1665280,1671132,948,107512,0,71540,80920,44128,8308,27412,72612,0,0,839676,839676,0,0,44000,22264,8436,35680
,24304,11376,1600,2940,0,0,0,1781768,296444,34359738367,7216,34359728128,0,6144,0,0,0,0,2048,34752,2062336

1487105963.001964,1964,ovis-demo-08,8,
0,1884188,1623168,1644996,948,129700,0,89312,101956,60788,8332,28524,93624,0,0,839676,839676,0,0,60620,23912,8500,3645
6,24608,11848,1872,4364,0,0,0,1781768,403252,34359738367,7216,34359728128,0,16384,0,0,0,0,2048,44992,2052096
```

CSV Store: Manual Aggregator Configuration

- Configure the aggregator to **store** the “meminfo” set to a **CSV** file using ldmsd_controller
  - Create a directory for the CSV data
  - Load the store_csv plugin
  - Configure the plugin

```
$ mkdir -p /home/<user>/exercises/ldms/data/CSV
$ ldmsd_controller --host localhost --port 20001
sock:localhost:20001> load name=store_csv
sock:localhost:20001> config name=store_csv path=/home/<user>/exercises/ldms/data/CSV buffer=0
```

- Check status

```
sock:localhost:20001> status
Name        Type        Interval    Offset      Libpath
------------ ------------ ------------ ------------ ------------
csv         store         1000000     0       /home/<user>/Build/OVIS-4.3.1/lib/ovis-ldms/libstore_csv.so
```

- name: plugin name
- path: Path to the base directory for the csv file container. This directory must pre-exist.
- buffer: ‘0’ to disable buffering # USE WITH CAUTION!
- man page:
  - man Plugin_store_csv – opens store_csv plugin man pages

CSV Store: Manual Aggregator Configuration (cont.)

- Configure the aggregator to **store** the “meminfo” set to a csv file.

```
sock:localhost:20001> strgp_add name=meminfo_store_csv plugin=store_csv
container=memory_metrics schema=meminfo
```

- Check status

```
sock:localhost:20001> status
Name               Container        Schema          Plugin          State
----------------   ---------------- --------------- --------------- ------------
meminfo_store_csv  memory_metrics   meminfo         store_csv       STOPPED
   producers:
   metrics:
```

- name: storage policy tag
- plugin: store plugin used for storing metric set data
- container: the storage backend container name. For csv, this is the directory where the output file will go. This will be created.
- schema: metric set schema to be stored

CSV Store: Manual Aggregator Configuration (cont.)

```
sock:localhost:20001> strgp_start name=meminfo_store_csv
```

- Check status

```
sock:localhost:20001> status
Name             Container        Schema           Plugin           State
---------------- ---------------- ---------------- ---------------- ------------
meminfo_store_csv memory_metrics   meminfo          store_csv        RUNNING
    producers:
    metrics: component_id job_id app_id MemTotal MemFree MemAvailable Buffers Cached
SwapCached Active Inactive Active(anon) Inactive(anon) Active(file) Inactive(file) Unevictable
Mlocked SwapTotal SwapFree Dirty Writeback AnonPages Mapped Shmem Slab SReclaimable
SUnreclaim KernelStack PageTables NFS_Unstable Bounce WritebackTmp CommitLimit
Committed_AS VmallocTotal VmallocUsed VmallocChunk HardwareCorrupted HugePages_Total
HugePages_Free HugePages_Rsvd HugePages_Surp Hugepagesize DirectMap4k DirectMap2M
DirectMap1G
```

- name: storage policy tag

CSV Store: LDMSD Status

```
sock:localhost:20001> status
```

```
Name         Type         Interval     Offset       Libpath
------------ ------------ ------------ ------------ ------------
csv          store        1000000            0 /home/<user>/Build/OVIS-4.3.1/lib/ovis-ldms/libstore_csv.so
Name             Host             Port         Transport    State
---------------- ---------------- ------------ ------------ ------------
prdcr1           nid00052                10001 sock         CONNECTED
    nid00052/meminfo meminfo          READY
    nid00052/vmstat  vmstat           READY
prdcr2           nid00053                10001 sock         CONNECTED
    nid00053/meminfo meminfo          READY
    nid00053/vmstat  vmstat           READY
Name             Interval     Offset       Mode            State
---------------- ------------ ------------ --------------- ------------
updtr1           1000000      200000       Pull            RUNNING
    prdcr1           nid00052                10001 sock         CONNECTED
    prdcr2           nid00053                10001 sock         CONNECTED

Name                             Container                        Schema           Plugin           State
----------------                 ----------------                 ---------------- ---------------- ------------
meminfo_store_csv                memory_metrics                   meminfo          store_csv        RUNNING
    producers:
    metrics: component_id job_id app_id MemTotal MemFree MemAvailable Buffers Cached SwapCached Active Inactive Active(anon)
Inactive(anon) Active(file) Inactive(file) Unevictable Mlocked SwapTotal SwapFree Dirty Writeback AnonPages Mapped Shmem Slab
SReclaimable SUnreclaim KernelStack PageTables NFS_Unstable Bounce WritebackTmp CommitLimit Committed_AS VmallocTotal VmallocUsed
VmallocChunk HardwareCorrupted HugePages_Total HugePages_Free HugePages_Rsvd HugePages_Surp Hugepagesize DirectMap4k
DirectMap2M DirectMap1G
```

Examining The CSV File

Exercise: Check the CSV file

```
$ head /home/<user>/exercises/ldms/data/CSV/memory_metrics/meminfo
$ tail -f /home/<user>/exercises/ldms/data/CSV/memory_metrics/meminfo
```

- If aggregating from others' vm's, you will see multiple hosts in the output

Exercise: View data changes:

- Run the memeater executable

```
$ /home/<user>/exercises/ldms/code/memeater
```

- Compare the live memeater output using tail –f "…/meminfo" values

EXAMPLE: CSV Store - Manual Aggregator Configuration

Please see Manual CSV Store to view a live example of these commands (slides 78-83).

CSV Store: Start and Configure Aggregator Using a Configuration File

- Edit aggregator configuration file, as appropriate, at: /home/<user>/exercises/ldms/conf/agg.conf

```
load name=store_csv
config name=store_csv path=/home/<user>/exercises/ldms/data/CSV  buffer=0
strgp_add name=meminfo-store_csv plugin=store_csv container=memory_metrics schema=meminfo
strgp_start name=meminfo-store_csv
```

- Restart your aggregator using: /home/<user>/exercises/ldms/scripts/start_agg.conf

EXAMPLE: CSV Store - Start and Configure Aggregator Using a Configuration File

Please see CSV Store Using Configuration File to view a live example of these commands (slide 85).

Basics End

LDMS HELP

ldms_controller: “help” Topics

```
sock:localhost.10001> help

Documented commands (type help <topic>):
========================================
EOF                    logrotate            setgroup_del        term
comment                oneshot              setgroup_ins        udata
config                 plugn_sets           setgroup_mod        udata_regex
connect                prdcr_add            setgroup_rm         updtr_add
daemon_exit            prdcr_del            shell               updtr_del
daemon_status          prdcr_hint_tree      source              updtr_match_add
env                    prdcr_set_status     start               updtr_match_del
failover_config        prdcr_start          stop                updtr_prdcr_add
failover_peercfg_start prdcr_start_regex    strgp_add           updtr_prdcr_del
failover_peercfg_stop  prdcr_status         strgp_del           updtr_start
failover_start         prdcr_stop           strgp_metric_add    updtr_status
failover_status        prdcr_stop_regex     strgp_metric_del    updtr_stop
failover_stop          prdcr_subscribe      strgp_prdcr_add     updtr_task
greeting               publish              strgp_prdcr_del     usage
help                   quit                 strgp_start         version
include                say                  strgp_status
load                   set_route            strgp_stop
loglevel               setgroup_add         subscribe

Undocumented commands:
======================
example                plugn_status         script              status              try
```

Definitely use for samplerd (start)
Definitely use for aggregators (prdcr_add, prdcr_start, updtr_add, updtr_start)
Definitely use for aggregators that store (strgp_add, strgp_start)
Use to load and config plugin (config, load)
Get help and daemon status (help, usage)

ldmsd_controller: Command Help

```
sock:localhost:10001> help prdcr_add
```

```
Add an LDMS Producer to the Aggregator
Parameters:
name=     A unique name for this Producer
xprt=     The transport name [sock, rdma, ugni]
host=     The hostname of the host
port=     The port number on which the LDMS is listening
type=     The connection type [active, passive]
interval= The connection retry interval (us)
```