SAND2023-05015C

OGC | Open Grid Computing, Austin, TX

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

Sandia National Laboratories: Jim Brandt, Sara Walton, Ben Allan

Open Grid Computing, Inc.: Tom Tucker

The *LDMS Version 4.3+ Basics Tutorial* is an updated and expanded version of previous tutorials: SAND2017-5153 O and SAND2021-13510 C

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-NA0003525.

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.

Tutorial Format (basic)

Overview of the Lightweight Distributed Metric Service (LDMS)

- Overview of the LDMS framework
- LDMS architecture description

Setup

- 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:

Configuring and deploying a distributed monitoring system with storage

- **Exercise 1:** Configuring and Running Samplers (~20 min)
  - Sampler startup and local and remote verification
  - Intro to ldmsd_controller and ldms_ls
- **Exercise 2:** Configure Aggregators (~20 min)
  - Aggregation startup and verification using local samplers
- **Exercise 3:** Storing Data In CSV Format (~20 min)

LDMS Overview

What is the Lightweight Distributed Metric Service (LDMS)?

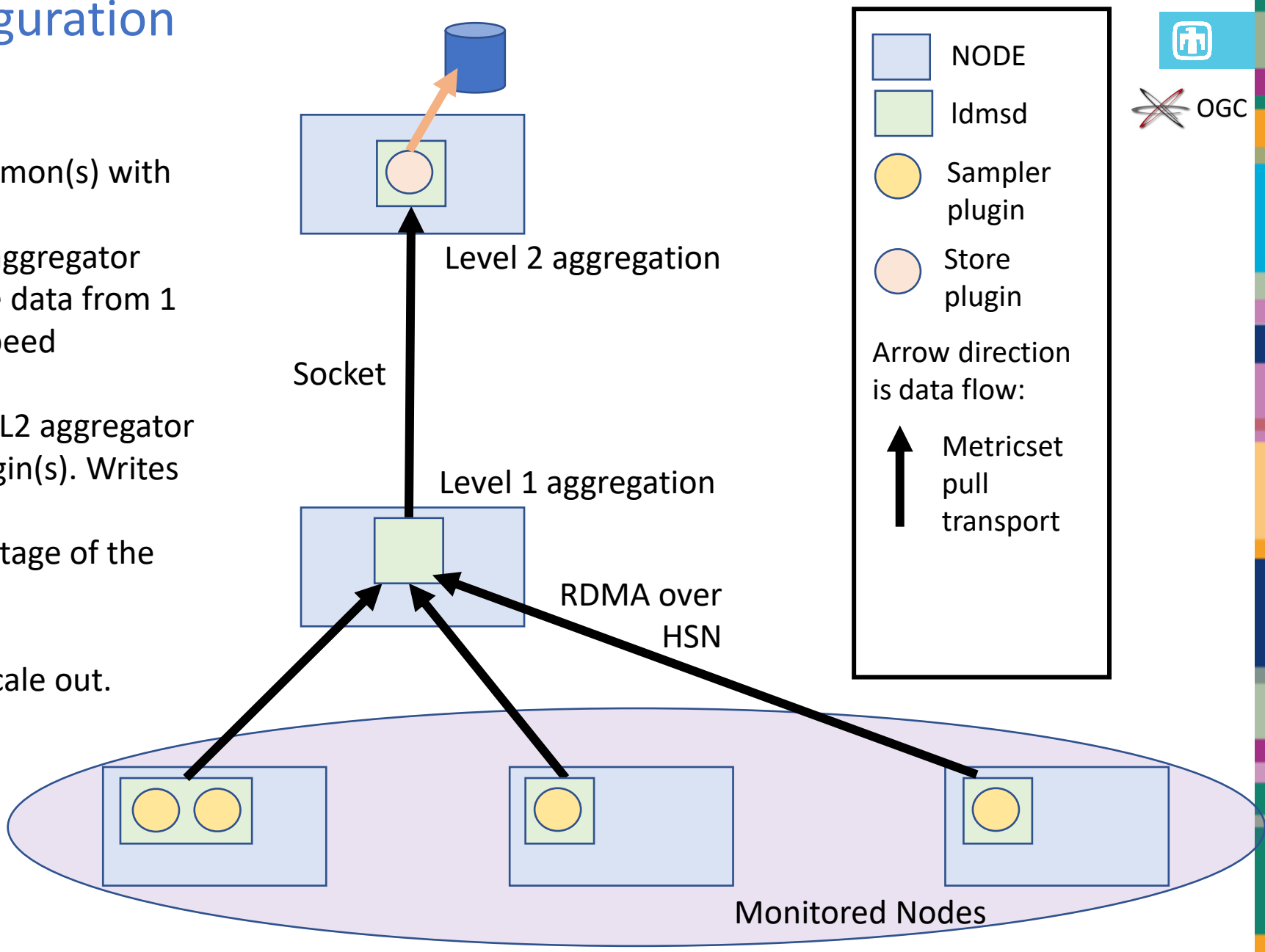
- Daemon based data collection
  - Plugin architecture
  - Sample numeric data
  - Sample application data
- Transport and aggregate data
- Store data

Typical use cases for information “stored” by LDMS

- Identify application execution behaviors
- Identify applications memory (and other resource) utilization behaviors
- Identify network congestion
- Identify heavy Lustre users

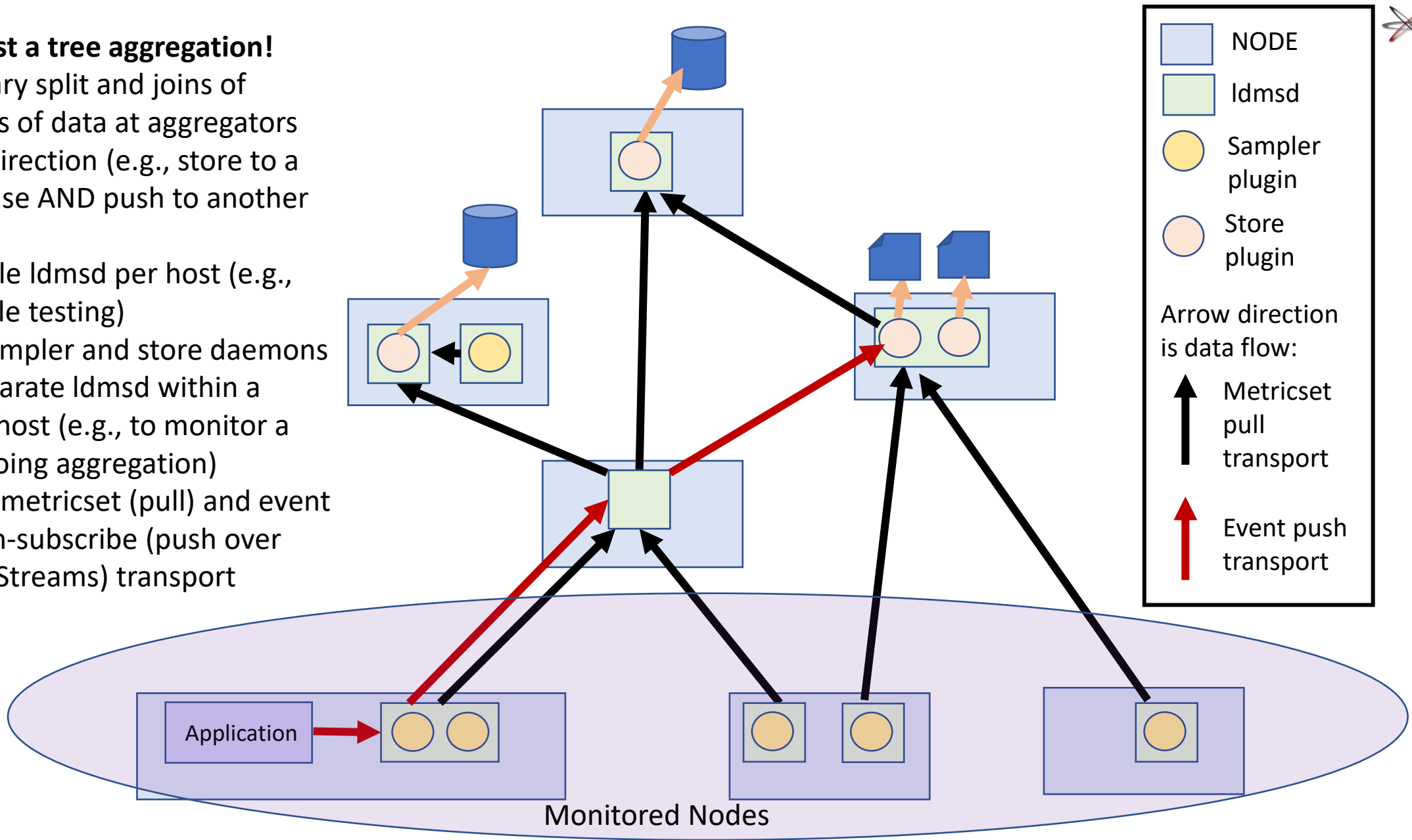
Common Simple Configuration

- Setup:
  - Compute Nodes have ldms daemon(s) with one or more sampler plugins
  - Cluster service node(s) has L1 aggregator ldms daemon(s). Aggregate the data from 1 or more nodes over the high speed transport. No plugins.
  - Off cluster storage node(s) has L2 aggregator ldms daemon(s) with store plugin(s). Writes out the aggregated data
  - Mixed transports to take advantage of the HSN
  - RDMA is a preference
- This topology can be replicated to scale out. Examples:
  - Replicate per “Scalable Unit”
  - Note that fan-in of ~thousands to one are possible

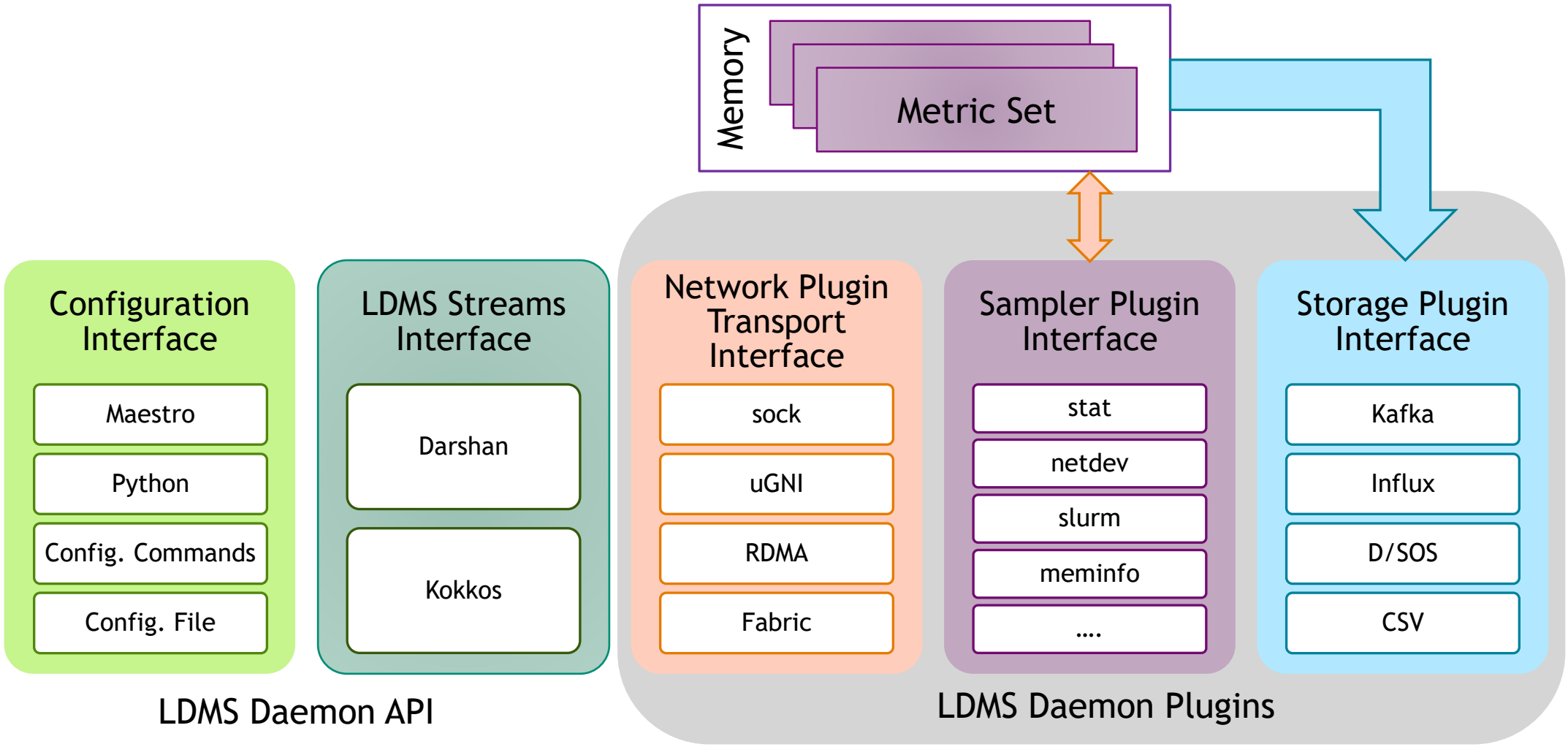


Anything is possible!

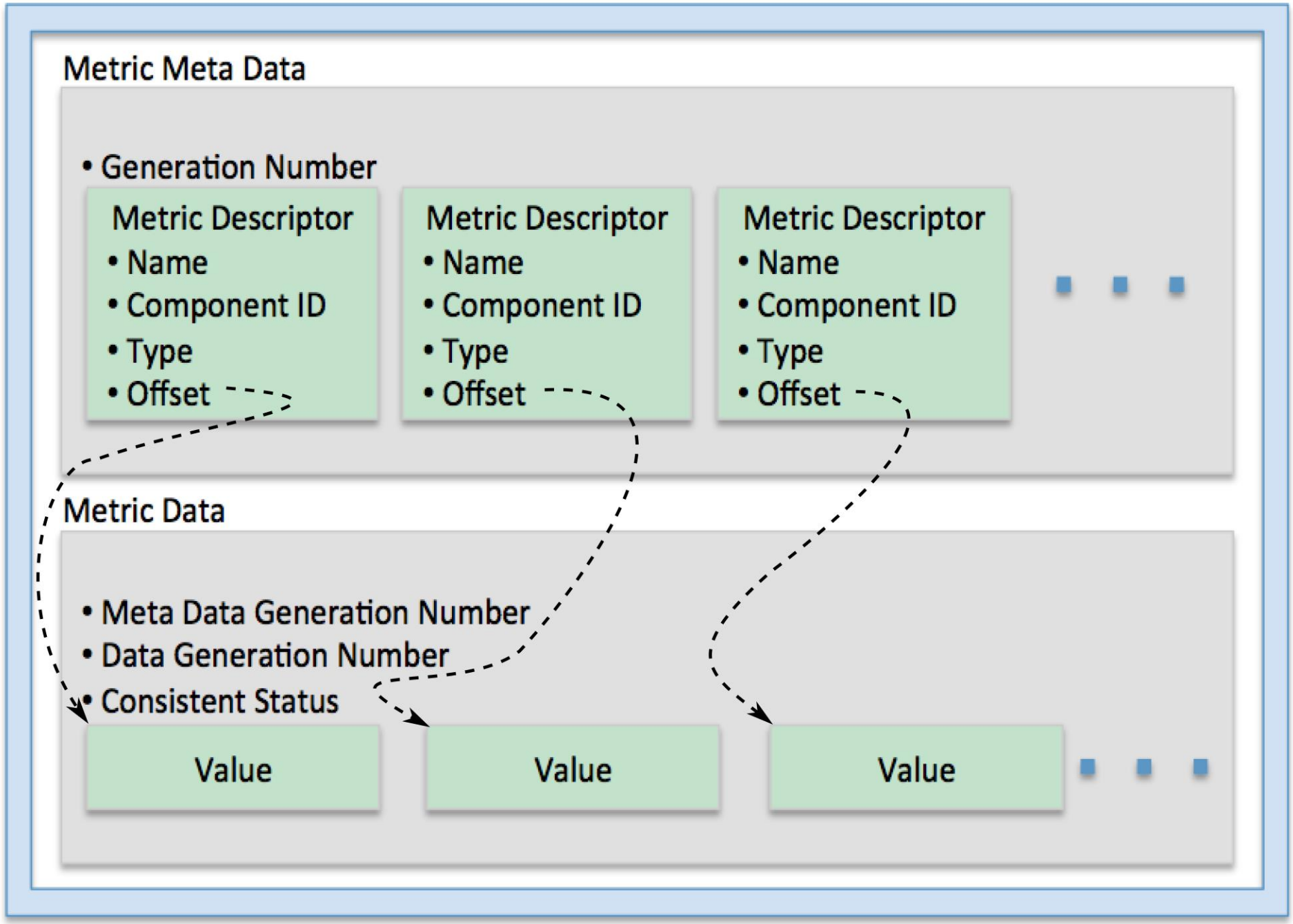
- **Not just a tree aggregation!** Arbitrary split and joins of subsets of data at aggregators for redirection (e.g., store to a database AND push to another ldmsd)
- Multiple ldmsd per host (e.g., for scale testing)
- Run sampler and store daemons on separate ldmsd within a single host (e.g., to monitor a host doing aggregation)
- Mixed metricset (pull) and event publish-subscribe (push over LDMS Streams) transport



LDMS Plugin Architecture



Metric Set Memory



OGC

- Meta-data only transferred upon initialization or change – this reduces both CPU and network burden
- Separation of data and meta-data blocks enables efficient RDMA transfer of data

Resources

Documentation (Building, Using)
- https://ovis-hpcreadthedocs.readthedocs.io/en/latest/

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 community support:** Post an issue at https://github.com/ovis-hpc/ovis/issues
- **Development and support:** contact tom@ogc.us
- **Support services:** contact tom@ogc.us

Supported Platforms and Networks & Dependencies

OGC

Linux support

- RHEL 7 & 8
- SLES 11 – 15
- TOSS
- Ubuntu
- RedHat (7-9)

Vendor hardware platforms supported software

- ARM
- Cray XC and EX
- Generic Linux clusters (may not have all dependencies)
- IBM Power (both big and little endian)

Transport Support

- Socket
- Cray ugni – Aries & Gemini
- RDMA – Infiniband, iWarp, libfabric

Typical compute node environment dependencies

- Autoconf >=2.63, automake, libtool (collectively called autotools)
- OpenSSH-devel
- lib*-devel for specific plugins that are enabled

LDMS Installation Methods

1. LDMS Development Containers
   - Dependencies and pre-requisites already installed
   - Easy to build and install LDMS
   - Pull & run development container
     - docker pull ovishpc/ldms-dev:4.3.11 #(based on Ubuntu 22.04)
     - docker run -it --name dev -hname dev ovishpc/ldms-dev:4.3.11

Example: Quick build in the dev container

- git pull https://github.com/ovis-hpc/ovis
- cd ovis
- ./autogen.sh
- mkdir build && cd build
- ../configure --prefix=/opt/ovis
- make && make install

For more info, see https://github.com/ovis-hpc/ovis

Other LDMS Installation Methods

1. Build and install using VM's or containers
2. Manually build and install using autoconf and automake
3. Build and install RPMs

Note 1: For this tutorial, LDMS is pre-installed on attendee containers in /opt/ovis

Note 2: We will be using the pre-installed software, scripts and configuration files for the exercises

Setup

Getting started: Log in and set up your environment

We will be using a container cluster provided by Open Grid Computing.

Instructions to download and access the LDMSCON2023 containers are shown below.

An email with an SSH identity file attached was sent out to those who registered to LDMSCON2023 via Eventbrite.

1. Download the SSH identity file (id_rsa) that was sent to your email address
2. Login to the container

```
$ ssh –i <path-to-id_rsa>/id_rsa root@<email-username>.ldms-ug.dev
```

Note: The “-i” option tells 'ssh' to use the specified id_rsa file instead of the default '~/.ssh/id_rsa’. This file will only work on your designated host so do not share with others.

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

Getting started: Directory structure

Containers include source code, scripts and configuration files for every exercise, helper mini-applications for use in the exercises

Directory structure:

/root/ldmscon2023/basic/	# Location of exercise related directories
/root/ldmscon2023/basic/conf/e*/	# Exercise configuration files
/root/ldmscon2023/basic/data/	# LDMS data
/root/ldmscon2023/basic/env/	# Scripts to configure environment variables
/root/ldmscon2023/basic/scripts/e*/	# Helper scripts for deploying LDMS daemons
/root/ldmscon2023/basic/logs/	# Place to write log files

Getting started: Set up and verify your environment

Source your environment configuration file (ldms-env.sh):

```
$ source /root/ldmscon2023/basic/env/ldms-env.sh
```

Contents of ldms-env.sh:

```
#!/bin/bash
TOP=/opt/ovis
export LD_LIBRARY_PATH=$TOP/lib64/:$LD_LIBRARY_PATH
export LDMSD_PLUGIN_LIBPATH=$TOP/lib/ovis-ldms
export ZAP_LIBPATH=$TOP/lib/ovis-ldms
export PYTHONPATH=$TOP/lib/python3.10/site-packages/:$PYTHONPATH
export PATH=$TOP/sbin:$TOP/bin:$PATH
```

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

Note: Container environments are already set so sourcing this file is optional.

Exercise 1: Configuring and Running Samplers

Start and Check Status of a LDMS Daemon

Exercise Goals:

Basic LDMS daemon startup and flags/args

- Command line configuration
- Run-Time Configuration

Use of ldms_ls utility as a diagnostic tool

For more help and information please visit the man pages

- man ldmsd – displays ldmsd man pages
- man ldmsd_controller – displays “ldmsd_controller” man pages
- man ldms_ls – displays ldms_ls man pages

Start a LDMS daemon

- Start ldmsd with minimum configuration

```
$ ldmsd -x sock:10001 -l ~/ldmscon2023/basic/logs/sampler1.log
```

-x: transport : listening port

-l: Specify the log file path and name(this is not strictly necessary)

Commands should be **written** in the command prompt window. Copy and paste may introduce non-printing characters and unexpected results

Terminal #1

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

Note: This is only a single method in checking the LDMS daemon process and is not an LDMS related command.

- Using ldms_ls

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

- 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

- Check the log file. This can be used to find clues when troubleshooting.

```
$ cat ~/ldmscon2023/basic/logs/sampler1.log
```

Note: The default values for transport (-x) and hostname (-h) are "sock" and "localhost", respectively.

Terminal #1

Manually Configure a Sampler Plugin

Exercise Goals:

Basic sampler plugin operation

- Manual configuration using the “ldmsd_controller” utility
- Static configuration using a configuration file
- Plugin_meminfo – Collect memory metrics from the memory usage report in /proc/meminfo
- Plugin_vmstat – Collect system metrics from the monitoring utility “vmstat”

Use of ldms_ls utility as a diagnostic tool

Man Pages

- man Plugin_meminfo – opens meminfo plugin man pages
- man Plugin_vmstat – opens vmstat plugin man pages
- man ldms_ls – opens ldms_ls man pages

Configuring a LDMS Daemon Sampler Plugin

Goals:

Load a sampler plugin

Configure loaded 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.**

Interactive Configuration Using The ldmsd_controller

Using a separate terminal (terminal #2), connect to your ldmsd using the "ldmsd_controller" utility

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

```
Welcome to the LDMSD control processor

sock:localhost:10001> help
```

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

*An example of running these commands can be found here: LDMSD Controller Interface Video

Terminal #2

Load and Configure the “meminfo” Sampler 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=${COMP_ID}
```

name: plugin name
producer: By convention set to host name (can be any string). Source of the ldms data.
Instance: By convention set to producer/<sampler name> (unique string). Uniquely identifies the metric set (e.g. where and what type of data is being produced).
component_id: By convention some unique numeric identifier (any uint_64). Another unique identifier for faster comparison (e.g. number)

- Optionally copy & paste above contents from the following text file under **# load and configure**:

```
$ cat ~/ldmscon2023/basic/scripts/e1/sampler_plugins.txt
```

Terminal #2

Query Current Sets Using “ldms_ls”

- In terminal #1, use ldms_ls to query the sets currently available on an LDMS daemon

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

```
${HOSTNAME}/meminfo
```

Terminal #1

Get The Set Meta-Data Before Starting The "meminfo" Sampler Plugin Using -v Flag


```
$ldms_ls -h localhost -x sock -p 10001 -v ${HOSTNAME}/meminfo
```

```
Schema          Instance                                    Flags  Msize  Dsize  UID    GID    Perm
                Update          Duration          Info
--------------- ----------------------                      ------ ------       ------  ------ ------
----------              ---------------- ---------------- --------

meminfo      ${HOSTNAME}/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 1: The "${HOSTNAME}/meminfo" is optional. Leaving it off will display the meta-data for all metric sets resident on this LDMS daemon.

Note 2: If you re-run this command multiple times and observe the timestamp, you will see the sampler is not updating. This means it has not started collecting "meminfo" data yet.

Terminal #1

Query Current Metric Values Before Starting The "meminfo" Sampler Plugin using -l flag


```
$ldms_ls -x sock -p 10001 -l ${HOSTNAME}/meminfo
```

```
${HOSTNAME}/meminfo: inconsistent, last update: Wed Dec 31 17:00:00 1969 -0700 [0us]
M u64      component_id                1
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
```

- *Values (set to 0) have not yet been collected*

- **Note 1:** We did not have to specify the hostname with –h because the default is "localhost".
- **Note 2**: The "${HOSTNAME}/meminfo" is optional. Leaving it off will display the data for all metric sets resident on this LDMS daemon.
- **Note 3**: The timestamp is the start of epoch time

Terminal #1

Start The “meminfo” Sampler Plugin Using the ldmsd_controller

- In terminal #2, start the “meminfo” sampler with a 1 second (1,000,000us) interval

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

- This starts the sampler updating the metric values every 1,000,000 micro-seconds = 1 second

Note 1: “interval” defines the number of micro-seconds between successive samples

- Optionally copy & paste above contents from the following text file under **# start**:

```
$ cat ~/ldmscon2023/basic/scripts/e1/sampler_plugins.txt
```

Terminal #2

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

```
$ ldms_ls -x sock -p 10001 -l ${HOSTNAME}/meminfo
```

```
${HOSTNAME}/meminfo: consistent, last update: Tue Oct 08 17:52:45 2019 -0600 [2058us]
M u64        component_id                1
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
```

- *Values are being collected*

- **Note 1:** We did not have to specify the hostname with –h because the default is "localhost".
- **Note 2**: The "${HOSTNAME}/meminfo" is optional. Leaving it off will display the data for all metric sets resident on this LDMS daemon.
- **Note 3:** The number in microseconds is when the sampling is scheduled to start. We have it to set to zero (every second on the second) and is a few milliseconds off due to the scheduler.

Terminal #1

Check Source (/proc/meminfo) For Reference

- Validate that the LDMS daemon is collecting “meminfo” data

```
$cat /proc/meminfo
```

```
MemTotal:       131899768 kB
MemFree:        129828892 kB
MemAvailable:   129350280 kB
Buffers:            20076 kB
Cached:            458076 kB
SwapCached:             0 kB
Active:            184380 kB
Inactive:          393064 kB
Active(anon):      125324 kB
Inactive(anon):    284680 kB
Active(file):       59128 kB
```

```
$ ldms_ls -x sock -p 10001 -l ${HOSTNAME}/meminfo
```

```
${HOSTNAME}/meminfo: consistent, last update: Tue Oct 08 17:52:45 2019 -0600 [2058us]
M u64     component_id            1
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                  184384
D u64     Inactive                393140
D u64     Active(anon)            125324
D u64     Inactive(anon)          284684
```

Terminal #1

Change The Sampling Interval

- Using ldmsd_controller, stop the plugin:

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

Note: Querying with ldms_ls –v 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
```

Note: Querying with ldms_ls -v multiple times will show that the metric set is now updating only every five seconds.

- **Exit** out of the ldmsd_controller with “quit” or Ctrl-d **OR kill** the ldmsd_controller with Ctrl-c

Terminal #2

Kill Currently Running Daemons

- Option 1: Kill all ldms daemons

```
$pkill ldmsd
```

- Option 2: Kill a particular ldms daemon

```
$ps auxw | grep ldmsd | grep -v grep
root        1577  0.0  0.0 763812  6088 ?        Ssl  18:14   0:00 ldmsd -x sock:10001 -l ~/ldmscon2023/ldms-basic/logs/sampler1.log
root        1585  0.0  0.0 102660 20768 pts/1    Sl+  18:15   0:00 python3 /opt/ovis/bin/ldmsd_controller -h localhost -x sock -p 10001
```

```
$kill 1577
$kill 1585
```

Note 1: You can alternatively stop the ldmsd_controller process by exiting ("quit" or Ctrl-d) or killing it (Ctrl-c) in terminal #2

- Check to make sure they are dead

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

Note 2: If you are having issues killing the daemon, you can always run "kill -9 <pid>"

Terminal #1 or #2

Start a ldmsd and Configure a Sampler Plugin Using a Configuration File

- Syntax is identical to that used for manual configuration
- Examine the sample configuration file for the meminfo example:

```
$cat ~/ldmscon2023/basic/conf/e1/simple_sampler.conf
```

- Alternatively create this file with the content shown below and filling in appropriate information:

```
load name=meminfo
config name=meminfo producer=${HOSTNAME} instance=${HOSTNAME}/meminfo component_id=${COMP_ID}
start name=meminfo interval=1000000
```

- Run an ldmsd using this configuration file (argument after the –c flag).
  - Using configuration files because bash variables are supported (i.e. $HOSTNAME)

```
$ldmsd -x sock:10001 -l ~/…/logs/sampler1.log -c ~/…/conf/e1/simple_sampler.conf
```

Terminal #1

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


```
$ ldms_ls -x sock -p 10001 -l ${HOSTNAME}/meminfo
```

${HOSTNAME}/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

- *Values are being collected*

- **Note 1:** We did not have to specify the hostname with –h because the default is "localhost".
- **Note 2**: The "${HOSTNAME}/meminfo" is optional. Leaving it off will display the data for all metric sets resident on this LDMS daemon.

Terminal #1

Multiple Sampler Plugins Running on a Single ldmsd

- Copy and paste the contents located in ~/…/scripts/e1/sampler_plugins.txt in the ldmsd_controller interface.
  - Alternatively modify the “simple_sampler.conf” file to include the above contents at the end of the file and then restart your ldmsd using -c

```
load name=vmstat
config name=vmstat producer=${HOSTNAME} instance=${HOSTNAME}/vmstat
component_id=${COMP_ID}
start name=vmstat interval=1000000
```

- Optionally copy & paste above contents from the following text file under **# vmstat plugin**:

```
$cat ~/ldmscon2023/basic/scripts/e1/sampler_plugins.txt
```

- Query the ldmsd using ldms_ls:

```
$ldms_ls -h localhost -x sock -p 10001
${HOSTNAME}/vmstat
${HOSTNAME}/meminfo
```

- **Do not kill this LDMS daemon** as we will be using it for the next exercise(s).

Terminal #1

Configuration Methods Summary

- Dynamic configuration using ldmsd_controller
  - ldmsd_controller is a Python script that can connect to a ldmsd via a configured network socket (supports command completion)
- Startup configuration via configuration file
  - Configuration file – loaded at ldmsd runtime
- All commands from this tutorial can be automated
  - Example: “echo config.file | ldmsd_controller –x sock –p 10001”

Sandia National Laboratories

OGC

Exercise 2: Configure LDMS Aggregators

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 and offset
  - Define which sets an update policy refers to (or all)
- Start the “update policy”

Start a ldmsd That Will Be Used For Aggregation

- Configure a **second** ldmsd sampler that listens on a **different host** with a **different component id** by first logging into another node and setting the $COMP_ID variable to 2:

```
$ ssh node-2
```

- Start this ldms sampler with the configuration file and have it **listen on port 10001** with log file "sampler2.log"

```
$ldmsd -x sock:10001 -l ~/ldmscon2023/basic/logs/sampler2.log -c
~/…/conf/e1/simple_sampler.conf
```

- Start a new aggregator ldmsd with minimum configuration:

```
$ldmsd -x sock:20001 -l ~/ldmscon2023/basic/logs/aggd.log
```

-x: Transport **:** listening port

-l: Specify the log file path and name (optional)

-c: Specify the configuration file path and name

Note 1: We are using **10001** for our **samplers** and port **20001** for our **aggregator**.
Note 2: The sampler would need to listen on a different port if it was started on the same node (i.e. 10001).
Note 3: The sampler listening on the second node (i.e. **node-2**) will only sample **meminfo**. If you wish for to sample **vmstat**, then please repeat the instructions in the previous exercise for this node **OR** uncomment the section **vmstat for second sampler** in ~/simple_sampler.conf and start it using this file.

Terminal #1

Interactive Aggregator Configuration Using the ldmsd_controller

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

```
$ldmsd_controller -h localhost –x sock -p 20001
Welcome to the LDMSD control processor
sock:localhost:20001>
```

Note: We need set the port to **20001** to connect to our **aggregator.**

Terminal #2

Simple Aggregator Producer Configuration

- Configure the aggregator to aggregate from your sampler daemons (listening on port 10001/10001)
  - prdcr_add: defines the name of the producer (i.e. daemon collecting the data) to aggregate from
  - prdcr_start: Initiates connection with a producer. It does NOT start a producer (which the name implies)

```
sock:localhost:20001> prdcr_add name=prdcr1 host=${HOSTNAME1} port=10001 xprt=sock type=active interval=20000000
sock:localhost:20001> prdcr_start name=prdcr1
sock:localhost:20001> prdcr_add name=prdcr2 host=${HOSTNAME2} port=10001 xprt=sock type=active interval=20000000
sock:localhost:20001> prdcr_start name=prdcr2
```

Note 1: The sequence of commands can be changed (i.e. the prdcr_add's can be inputted before the prdcr_start's)

name: policy tag (this is just a string)
host: hostname of the **sampler daemon** that the aggregator is connecting to
port: Port the sampler daemon listens on
xprt: Transport the sampler daemon listens on
type: Choose "active" (aggregator will initiate the connection with the sampler)
interval: Re-connect interval (set to 20 seconds) (*This is **NOT the sampler interval**)*

- **Optionally** copy & paste contents located in the text file under **# producers**:

```
$ cat ~/ldmscon2023/basic/scripts/e2/agg_samplers.txt
```

Terminal #2

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        node-1        10001 sock        CONNECTED
   node-1/meminfo meminfo         START
   node-1/vmstat vmstat         START
prdcr2        node-2        10001 sock        CONNECTED
   node-2/meminfo meminfo         START
Name          Interval:Offset  Auto  Mode          State       Skipped counter Oversampled counter
------------- ---------------- ----- ------------- ----------- --------------- ------------------
Name          Container       Schema          Plugin          flush(sec)      State
------------- --------------- --------------- --------------- ----------- ---------------
```

Terminal #2

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 there is no update policy associated with the connected prdcr and the sets have not yet been created and populated with data at the aggregator.

Terminal #1

Configure and Start Aggregator Updater Policy

- Configure the aggregator to **update** all schemas (i.e. meminfo and vmstat).

```
sock:localhost:20001> updtr_add name=update_all interval=1000000 auto_interval=true
sock:localhost:20001> updtr_prdcr_add name=update_all regex=.*
sock:localhost:20001> updtr_start name=update_all
```

name: policy tag (string)

interval: update (pull) interval (in usec)

- Example: interval=1000000 means it will pull data from associated prdcr every 1 second

auto_interval: Automatically aggregate according to update hint (default false)

- Example: This daemon will aggregate every <interval> seconds set in the sampler (i.e. the "`updt_hint_us`" from ldms_ls -v)
- If false, then it will default to the interval specified on the aggregator.

regex: regular expression to match the target producers tag(s)

- prdcr1 in this case (see slide 50)

- **Optionally** copy & paste contents located in the text file under **# updater**:

```
$ cat ~/ldmscon2023/basic/scripts/e2/agg_samplers.txt
```

Terminal #2

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          node-1          10001 sock        CONNECTED
   node-1/meminfo meminfo         READY
   node-1/vmstat vmstat           READY
prdcr2          node-2          10001 sock        CONNECTED
   node-2/meminfo meminfo         READY
Name            Interval:Offset  Auto   Mode          State       Skipped counter Oversampled counter
--------------- ---------------- ------ ------------- ----------- --------------- ------------------
updtr1          1.0s:0.0ms       true   Pull          RUNNING                   0                  0
   prdcr1          node-1          10001 sock        CONNECTED
   prdcr2          node-2          10001 sock        CONNECTED
Name            Container       Schema          Plugin          flush(sec)      State
--------------- --------------- --------------- --------------- ----------- ---------------
```

Terminal #2

Query Current Metric Values On The Aggregator


```
$ldms_ls -h localhost -x sock -p 20001 -l ${HOSTNAME}/meminfo
```

```
${HOSTNAME}/meminfo: consistent, last update: Wed Oct 09 18:30:49 2023 -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
```

Note: You can query the other sampler with the set instance name: ${HOSTNAME}/meminfo
Or all of them with –l only.

Terminal #1

Start Aggregator ldmsd Using a Configuration File

- A ldmsd for performing aggregation can also be started using a configuration file in the same manner as a ldmsd for sampling (see slide 40)
- Configuration file syntax is identical to that used for manual configuration
- Check out your sample configuration file:

```
$cat ~/ldmscon2023/basic/conf/e2/simple_agg.conf
```

Alternatively create a conf file in this directory and populate it with the contents below:

```
prdcr_add name=prdcr1 host=${HOSTNAME1} port=10001 xprt=sock type=active interval=20000000
prdcr_start name=prdcr1
prdcr_add name=prdcr2 host=${HOSTNAME2} port=10001 xprt=sock type=active interval=20000000
prdcr_start name=prdcr2
updtr_add name=update_all interval=1000000 auto_interval=true
updtr_prdcr_add name=update_all regex=.*
updtr_start name=update_all
```

Terminal #1

Start Aggregator ldmsd Using a Configuration File

- **Exit** out of the ldmsd_controller with “quit” or Ctrl-d **OR kill** the ldmsd_controller with Ctrl-c
- Kill **ONLY** your **aggregator** ldmsd (i.e. port 2001).

```
$ps auxw | grep ldmsd | grep -v grep
ovis_pu+  3582  0.0  0.1 401604  2204 ?        Ssl  12:51   0:00
ldmsd -x sock:20001 -S samplerd.sock
$kill 3582
```

- Restart your aggregator using the “simple_agg.conf” configuration file

```
$ldmsd -x sock:20001 -l ~/ldmscon2023/basic/logs/aggd.log -c
~/ldmscon2023/basic/conf/e2/simple_agg.conf
```

Note: If you kill all LDMS daemons (samplers and aggregator) you can easily restart the samplers using a configuration file (exercise 1).

Terminal #1

Query Current Metric Values On The Aggregator


```
$ldms_ls -x sock -p 20001 -l ${HOSTNAME}/meminfo
```

${HOSTNAME}/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

Note: You can query the other sampler with the set instance name: ${HOSTNAME}/meminfo
Or all of them with –l

Terminal #1

Exercise 3: Storing Data In CSV Format

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)
- **Note:** The scripts/e1 - e3 directories contain scripts to start ldmsd using associated configuration files

Example output from the "meminfo" sampler:

#Time,Time_usec,ProducerName,component_id,job_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,AnonHugePages,HugePages_Total,HugePages_Free,HugePages_Rsvd,HugePages_Surp,Hugepagesize,DirectMap4k,DirectMap2M

1487105964.002482,2482,node-3,3,
0,1884188,571028,1688632,0,1212004,6108,104536,1122496,8276,8580,96260,1113916,0,0,839676,793956,420,0,10552,24812,1796,52124,40104,12020,1792,3280,0,0,0,1781768,387984,34359738367,7216,34359728128,0,2048,0,0,0,0,2048,47040,2050048

1487105963.002583,2583,${HOSTNAME}0,10,
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,${HOSTNAME}2,12,
0,1884188,1623168,1644996,948,129700,0,89312,101956,60788,8332,28524,93624,0,0,839676,839676,0,0,60620,23912,8500,36456,24608,11848,1872,4364,0,0,0,1781768,403252,34359738367,7216,34359728128,0,16384,0,0,0,0,2048,44992,2052096

CSV Store: Manual Aggregator Configuration

OGC

- DO NOT kill the LDMS daemons (i.e. samplers and aggregator)
- 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 and configure the plugin

```
$ldmsd_controller -h localhost -x sock -p 20001
```

```
sock:localhost:20001> load name=store_csv
sock:localhost:20001> config name=store_csv path=/root/ldmscon2023/basic/data/ buffer=0
```

name: plugin name

path: Path to the base directory for the csv file container. This **directory must exist** prior to loading this configuration or daemon will throw an error and terminate.

buffer: '0' to disable buffering. This will flush each row to the csv file and allow us to see live updates of the stored data.

man page:

```
$man Plugin_store_csv          # opens store_csv plugin man pages
```

- **Optionally** copy & paste contents located in the text file under **# load**:

```
$ cat ~/ldmscon2023/basic/scripts/e3/store_csv.txt
```

Terminal #2

CSV Store: Cont.

- Check status

```
sock:localhost:20001> status
Name       Type       Interval   Offset     Libpath
---------- ---------- ---------- ---------- ----------
csv        store      1000000          0 /opt/ovis/lib/ovis-ldms/libstore_csv.so
Name           Host           Port       Transport  State
-------------- -------------- ---------- ---------- ----------
prdcr1         node-1         10001 sock       CONNECTED
  node-1/meminfo meminfo       READY
  node-1/vmstat vmstat         READY
prdcr2         node-2         10001 sock       CONNECTED
  node-2/meminfo meminfo       READY
Name           Interval:Offset  Auto   Mode         State      Skipped counter Oversampled counter
-------------- ---------------- ------ ------------ ---------- --------------- ------------------
updtr1         1.0s:0.0ms       true   Pull         RUNNING                 0                  0
  prdcr1         node-1         10001 sock       CONNECTED
  prdcr2         node-2         10001 sock       CONNECTED
Name           Container      Schema         Plugin         flush(sec)      State
-------------- -------------- -------------- -------------- ----------- ----------
```

Terminal #2

CSV Store: 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
```

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 in “path”.
schema: metric set schema to be stored

Note: The default for schema is sampler plugin name but can be modified but it is for a more advanced tutorial.

- **Optionally** copy & paste contents located in the text file under **# config**:

```
$ cat ~/ldmscon2023/basic/scripts/e3/store_csv.txt
```

Terminal #2

CSV Store: Cont.

- Check status

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

```
Name       Type       Interval   Offset     Libpath
---------- ---------- ---------- ---------- ----------
csv        store      1000000    0          /opt/ovis/lib/ovis-ldms/libstore_csv.so
Name       Host       Port       Transport  State
---------- ---------- ---------- ---------- ----------
prdcr1     node-1     10001      sock       CONNECTED
  node-1/meminfo meminfo      READY
  node-1/vmstat vmstat       READY
prdcr2     node-2     10001      sock       CONNECTED
  node-2/meminfo meminfo      READY
Name       Interval:Offset  Auto  Mode       State      Skipped counter  Oversampled counter
---------- ---------- ------ ---------- ---------- ---------- ----------
updtr1     5.0s:0.0ms       true  Pull       RUNNING    0                0
  prdcr1     node-1     10001      sock       CONNECTED
  prdcr2     node-2     10001      sock       CONNECTED
Name       Container  Schema     Plugin     flush(sec)  State
---------- ---------- ---------- ---------- ---------- ----------
meminfo-store_csv memory_metrics  meminfo    store_csv  0.000000    STOPPED
  producers:
  metrics:
```

Terminal #2

CSV Store: Continued

- Start meminfo-store_csv policy

```
sock:localhost:20001> strgp_start name=meminfo-store_csv
```

name: storage policy tag

- **Optionally** copy & paste contents located in the text file under **# start**:

```
$ cat ~/ldmscon2023/basic/scripts/e3/store_csv.txt
```

Terminal #2

CSV Store: Cont.

OGC

- Check status

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

```
Name       Type       Interval    Offset      Libpath
---------- ---------- ----------- ----------- ----------
csv        store          1000000           0 /opt/ovis/lib/ovis-ldms/libstore_csv.so
Name           Host           Port       Transport    State
-------------- -------------- ---------- ------------ ------------
prdcr1         node-1         10001 sock       CONNECTED
   node-1/meminfo meminfo        READY
   node-1/vmstat vmstat          READY
prdcr2         node-2         10001 sock       CONNECTED
   node-2/meminfo meminfo        READY
Name           Interval:Offset  Auto   Mode          State       Skipped counter Oversampled counter
-------------- ---------------- ------ ------------- ----------- --------------- ------------------
updtr1         5.0s:0.0ms       true   Pull          RUNNING               0               0
   prdcr1         node-1         10001 sock       CONNECTED
   prdcr2         node-2         10001 sock       CONNECTED
Name           Container      Schema         Plugin         flush(sec)      State
-------------- -------------- -------------- -------------- ------------ ---------------
meminfo-store_csv memory_metrics   meminfo        store_csv      0.000000        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…..
```

Terminal #2

Examining The CSV File

- Check the CSV file

```
$head ~/ldmscon2023/basic/data/memory_metrics/meminfo
$tail -f ~/ldmscon2023/basic/data/memory_metrics/meminfo
$ls -ltr ~/ldmscon2023/basic/data/memory_metrics/
```

- Note: You do not have to do tail and head. This is only to show that we can see the data is continuously being populated.

Terminal #1

CSV Store: Start and Configure Aggregator Using a Configuration File

- View store relevant part of configuration file at: ~/ldmscon2023/basic/conf/e3/agg_store_csv.conf

```
load name=store_csv
config name=store_csv path=/root/ldmscon2023/basic/data buffer=0


strgp_add name=meminfo-store_csv plugin=store_csv container=memory_metrics
schema=meminfo
strgp_start name=meminfo-store_csv


strgp_add name=vmstat-store_csv plugin=store_csv container=memory_metrics
schema=vmstat
strgp_start name=vmstat-store_csv
```

- **Note 1:** This configuration file also stores the **vmstat metric set** in memory_metric
- **Note 2:** Can also have different containers for meminfo and vmstat

Terminal #1

CSV Store: Start and Configure Aggregator Using a Configuration File

- **Exit** out of the ldmsd_controller with “quit” or Ctrl-d **OR kill** the ldmsd_controller with Ctrl-c
- Kill **ONLY** your **aggregator** ldmsd (i.e. port 2001).

```
$ps auxw | grep ldmsd | grep -v grep
ovis_pu+  3582  0.0  0.1 401604  2204 ?        Ssl  12:51   0:00 ldmsd -x
sock:20001 -S samplerd.sock
$kill 3582
```

- Restart your aggregator using the “agg_store_csv.conf” configuration file

```
$ldmsd -x sock:20001 -l ~/ldmscon2023/basic/logs/agg_store.log -c
~/ldmscon2023/basic/conf/e3/agg_store_csv.conf
```

- Check the CSV file

```
$head ~/ldmscon2023/basic/data/memory_metrics/meminfo
$tail -f ~/ldmscon2023/basic/data/memory_metrics/meminfo
$ls -ltr ~/ldmscon2023/basic/data/memory_metrics/
```

Terminal #1

Basics End