

OGC | Open Grid Computing, Austin, TX



# LDMS Version 4.3+ Basics Tutorial

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

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

Open Grid Computing, Inc.: Tom Tucker

# 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:

### *Configuring and deploying a distributed monitoring system with storage*

- **Exercise 1:** Configuring and Running Samplers (37 slides – ~20 min)
  - Sampler startup and local and remote verification
  - Intro to ldmsd\_controller and ldms\_ls
- **Exercise 2:** Configure Aggregators (13 slides – ~20 min)
  - Aggregation startup and verification using local samplers
  - Aggregation of all other attendees' (remote) samplers
- **Exercise 3:** Aggregating From Remote Hosts: Building a Distributed Monitoring System (4 slides – if time permits)
- **Exercise 4:** Dynamic Configurations and Resilience (4 slides - if time permits)
- **Exercise 5:** Storing Data In CSV Format (11 slides - ~20 min)

## What is the Lightweight Distributed Metric Service (LDMS)?

- Daemon based data collection
  - Plugin architecture
  - Sample and transport numeric data
  - Transmit information published to publish/subscribe API
- Transport and aggregate data
- Store data

## Typical use cases for information “stored” by LDMS

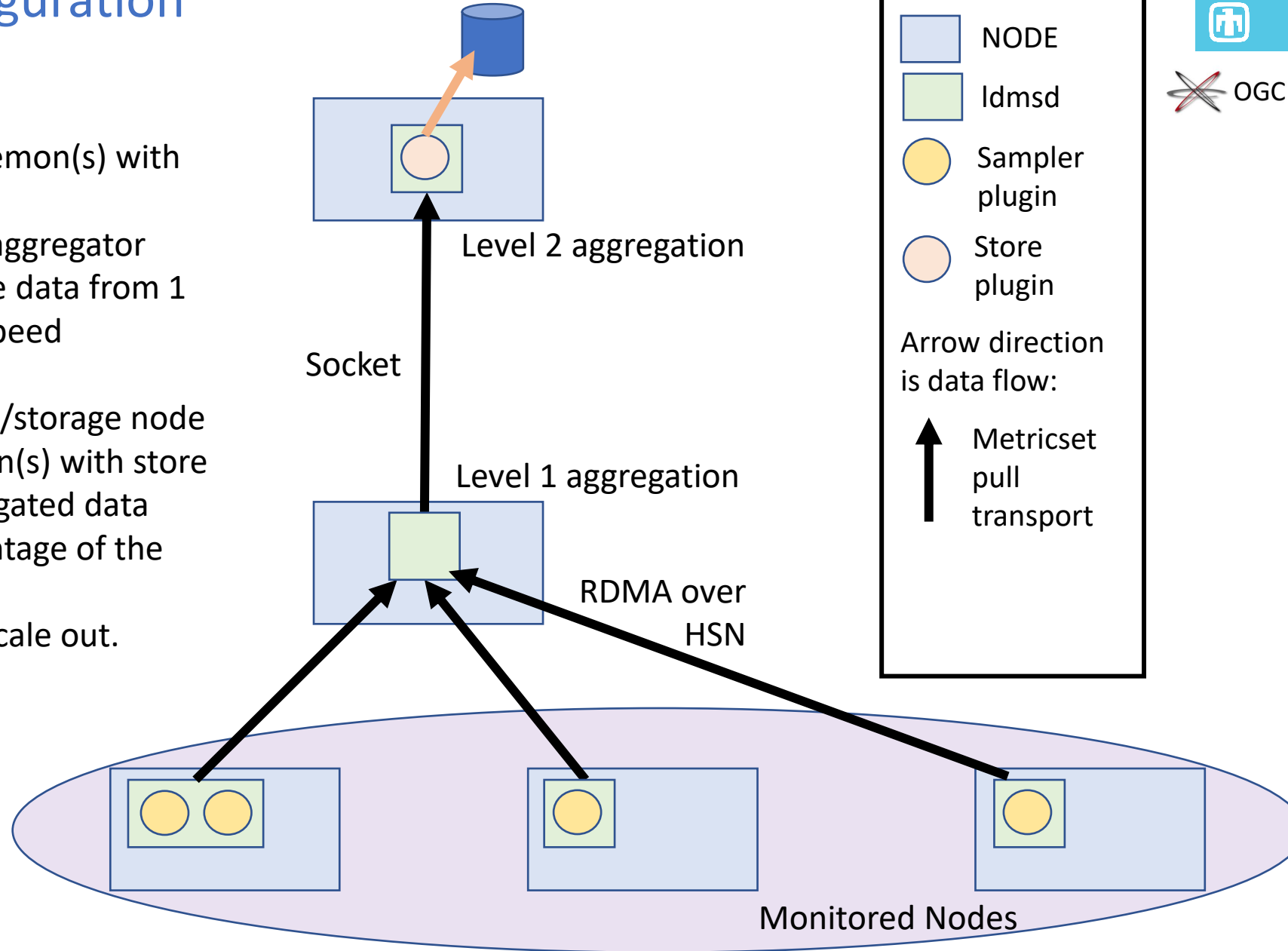
- Identify application execution behaviors
- Identify applications memory (and other resource) utilization behaviors
- Identify network congestion
- Determine over-provisioned resources
- Identify heavy Lustre users

# Common Simple Configuration

- Setup:
  - Compute Nodes have Idms daemon(s) with one or more sampler plugins
  - Cluster service nodes have L1 aggregator Idms daemon(s). Aggregate the data from 1 or more nodes over the high speed transport. No plugins.
  - Off cluster monitoring/analysis/storage node has L2 aggregator Idms daemon(s) with store plugin(s). Writes out the aggregated data
  - Mixed transports to take advantage of the HSN
- This topology can be replicated to scale out.
 

Examples:

  - Replicate per “Scalable Unit”
  - Note that fan-in of ~thousands to one are possible

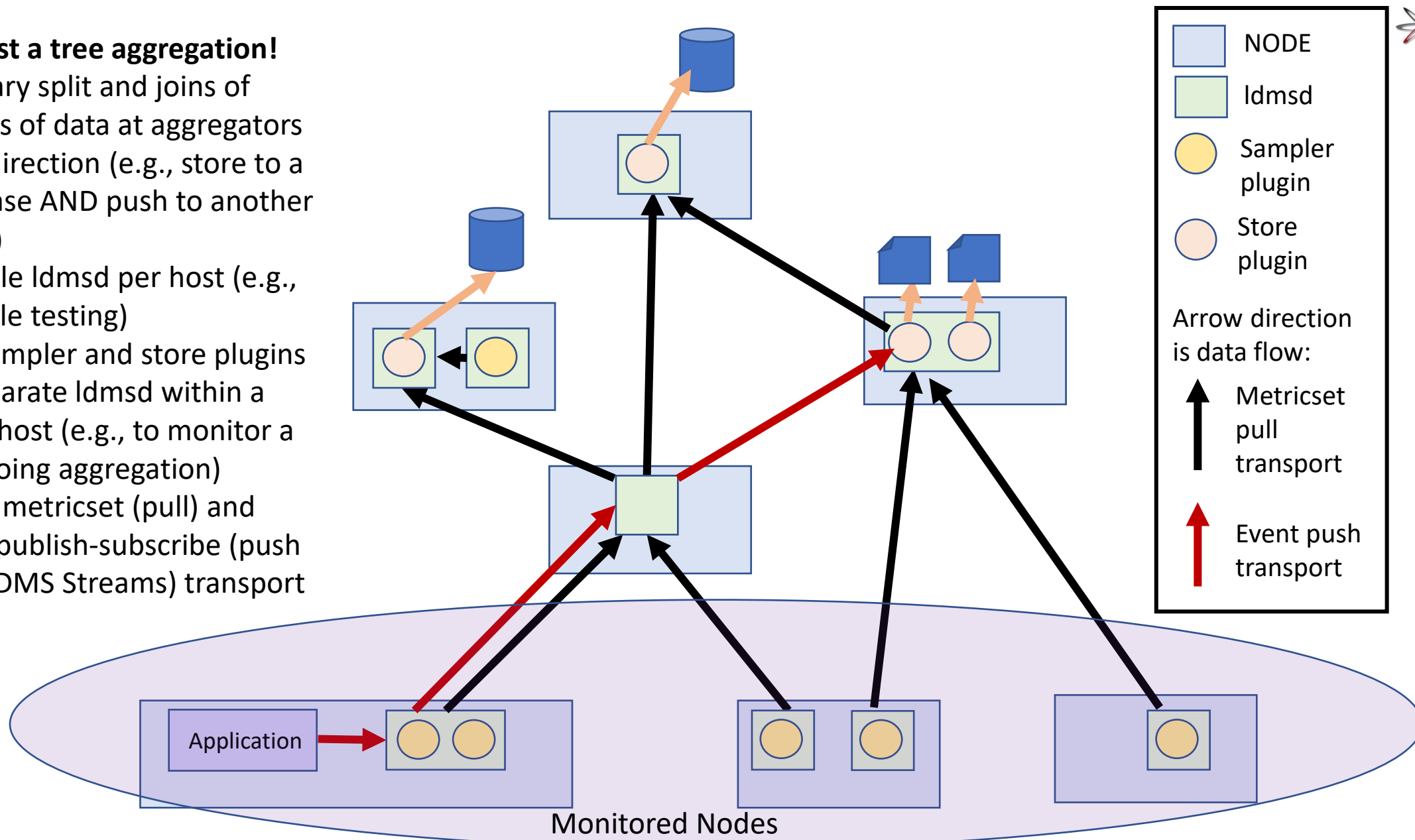


# Anything is possible!

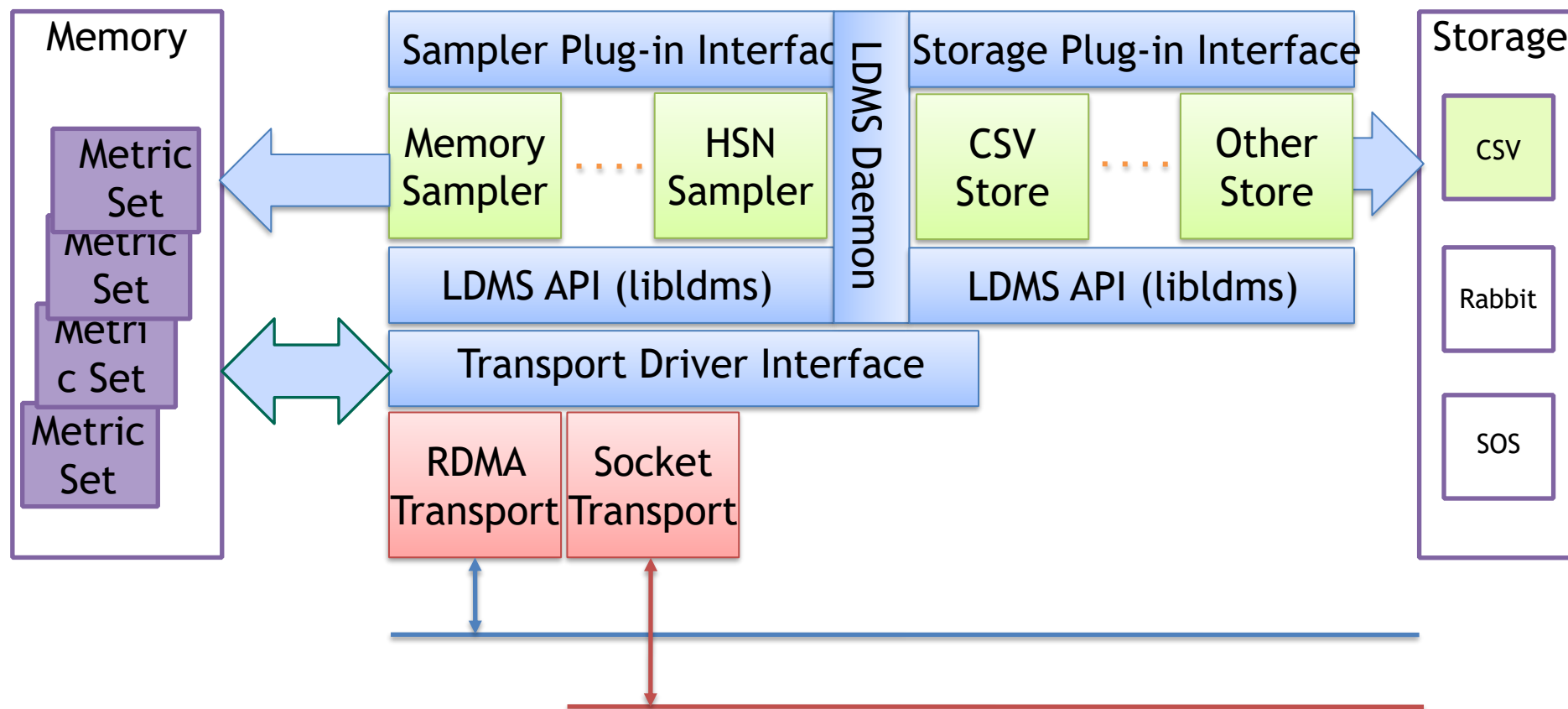


OGC

- **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 Idmsd)
- Multiple Idmsd per host (e.g., for scale testing)
- Run sampler and store plugins on separate Idmsd 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

7



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

## Metric Meta Data

### • Generation Number

#### Metric Descriptor

- Name
- Component ID
- Type
- Offset

#### Metric Descriptor

- Name
- Component ID
- Type
- Offset

#### Metric Descriptor

- Name
- Component ID
- Type
- Offset



## Metric Data

- Meta Data Generation Number
- Data Generation Number
- Consistent Status

Value

Value

Value



## Documentation (Building, Using)

- <https://github.com/ovis-hpc/ovis-wiki/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](mailto:tom@ogc.us)
- **Support services:** contact [tom@ogc.us](mailto:tom@ogc.us), [ldms@sandia.gov](mailto:ldms@sandia.gov)



# Supported platforms and networks



## Linux support

- RHEL 6 - 8
- SLES 11 - 15
- 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  $\geq 2.63$ , 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
- Doxygen for documentation
- Cython needed for SOS
  - Get from pip
- libcurl & libcurl-devel if using influx\_store
- flex and bison including devel versions
- etcd if using Maestro

# LDMS Installation methods



Manually build and install using autoconf and automake

Build and install RPMs

**Note1:** For this tutorial, LDMS is pre-installed on student containers 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



We will be using containers hosted at Open Grid Computing

Place the key files (e.g. user<#>\_id\_rsa and user<#>\_id\_rsa.pub) into the ~/.ssh/ directory.

Add the following entries to ~/.ssh/config

```
Host Idmscon2021
```

```
    Hostname ogc.us
```

```
    Port 65422
```

```
    User user64 # <----- Change to the assigned username
```

```
    IdentityFile ~/.ssh/user64_id_rsa # <---- Change to the assigned key file
```

```
    IdentitiesOnly yes
```

```
$ ssh Idmscon2021
```

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

# Directory structure



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

Directory structure:

<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/</code>	<code># Location of exercise related directories</code>
<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/conf/E*/</code>	<code># Exercise configuration files</code>
<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/ldms/data/</code>	<code># LDMS data</code>
<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/ldms/ env/</code>	<code># Scripts to configure environment variables</code>
<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/ldms/scripts/E*/</code>	<code># Helper scripts for deploying LDMS daemons</code>
<code>/home/&lt;user&gt;/ldmscon2021/basic/exercises/ldms/logs/</code>	<code># Place to write log files</code>

# Getting started: Set up and verify your environment



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

```
$ source 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=$TOP/lib64/ovis-ldms
export ZAP_LIBPATH=$TOP/lib64/ovis-ldms
export PYTHONPATH=$TOP/lib/python3.6/site-packages/:$PYTHONPATH
export PATH=$TOP/sbin:$TOP/bin:$PATH
```

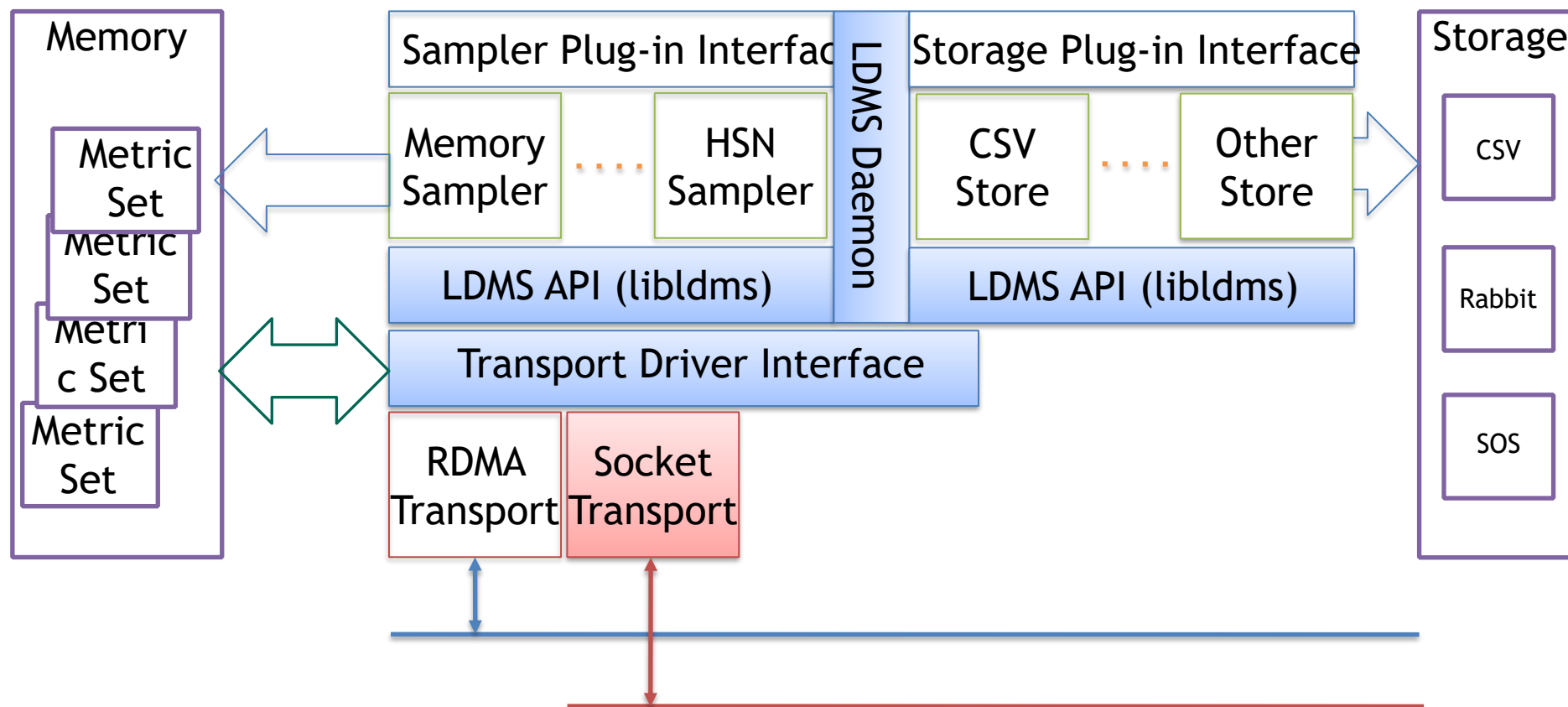
\*A live example of these commands can be found here:

[Verify Environment Variables](#)

# Exercise 1: Configuring and Running Samplers



# LDMS Plugin Architecture



## Exercise Goals:

Basic LDMS daemon startup and configuration flags/args

- Manual and run-time configuration options
- Output options
  - Log files and verbosity levels
- 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>/ldmscon2021/basic/exercises/ldms/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

# 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 ?          ss1 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

- Also check network port for listener

```
$netstat -an | grep 10001 OR $ss -ln | grep 10001
```

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

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

```
$ cat /home/<user>/ldmscon2021/basic/exercises/ldms/logs/sampler1.log
```

## EXAMPLE: Start and Check LDMS Daemon

Please see the [Start and Check an LDMS daemon](#) to view a video example of Exercise 1 (slides 19-20).

# Manually Configure a Sampler Plugin



## Exercise Goals:

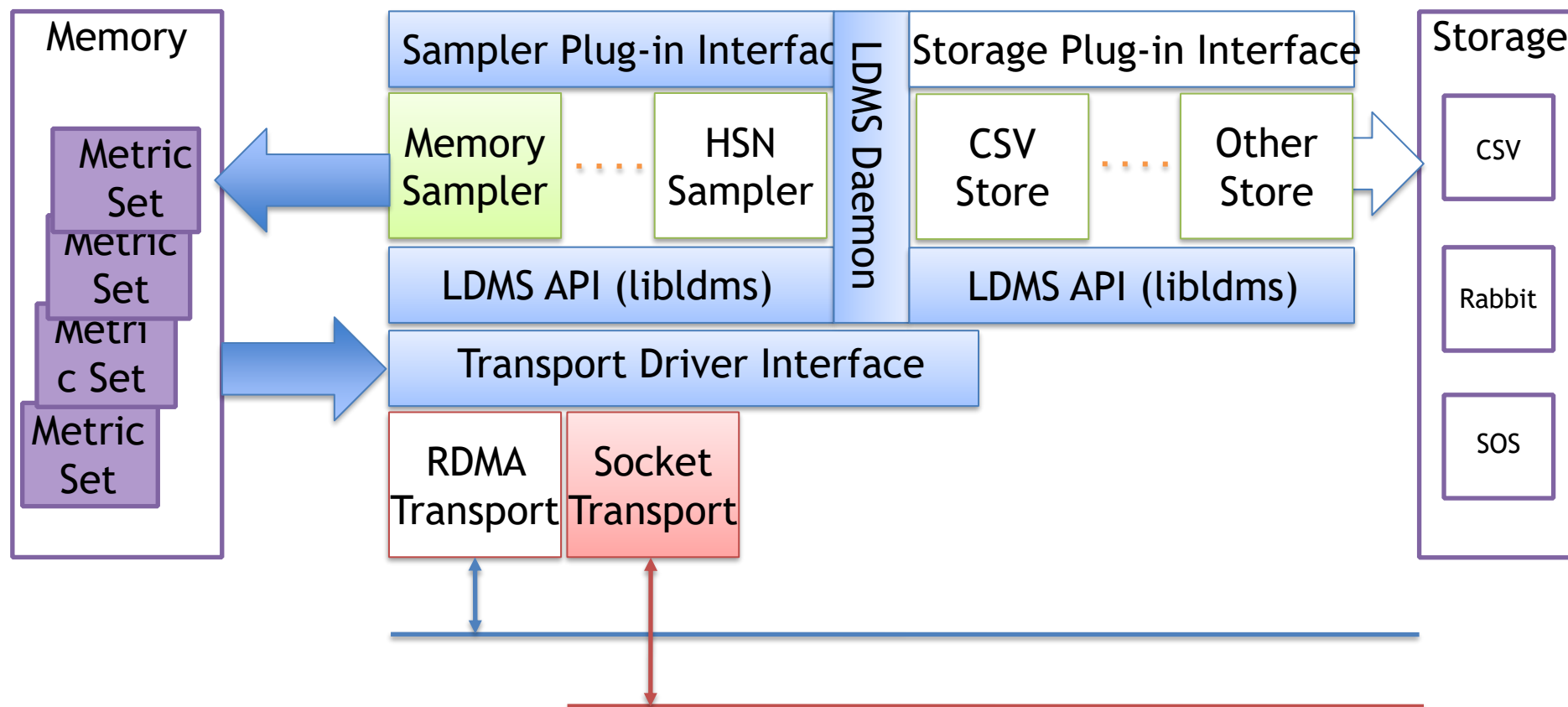
### Basic sampler plugin operation

- 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



# 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** and **offset**



# Interactive Configuration Using The `ldmsd_controller`



Connect to your `ldmsd` using the “`ldmsd_controller`” utility

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

```
welcome to the LDMSD control processor
```

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

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`

\*An example of running these commands can be found here: [LDMSD Controller Interface Video](#)

# 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=<host number>
```

**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<sub>64</sub>). Another unique identifier for faster comparison (e.g. number)

# Query Current Sets Using “ldms\_ls”



- Use ldms\_ls to query the sets currently available on an LDMS daemon

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

node-1/meminfo

# Get The Set Meta-Data Before Starting The “meminfo” Sampler Plugin Using -v Flag



```
$ldms_ls -h localhost -x sock -p 10001 -v node-1/meminfo
```

Schema Perm	Instance	Update	Duration	Flags Info	Msize	Dsize	UID	GID
-----	-----	-----	-----	-----	-----	-----	-----	-----
meminfo rwxrwxrwx	node-1/meminfo 0.000000	0.000000	L 1952	416 "updt_hint_us"="1000000:0"	596 742	-		
-----	-----	-----	-----	-----	-----	-----	-----	-----

Total Sets: 1, Meta Data (kB): 1.95, Data (kB) 0.42, Memory (kB): 2.37

**NOTE:** The “node-1/meminfo” is optional. Leaving it off will display the meta-data for all metric sets resident on this LDMS daemon.

## EXAMPLE: Interactive Configuration Using The ldmsd\_controller

Please see [Manual Meminfo Configuration](#) to view a video example of Exercise 1 (slides 25-28).

# Query Current Metric Values Before Starting The “meminfo” Sampler Plugin using -l flag



```
$ldms_ls -x sock -p 10001 -l node-1/meminfo
```

node-1/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*

**NOTE:** The “node-1/meminfo” is optional. Leaving it off will display the data for all metric sets resident on this LDMS daemon.

# Start The “meminfo” Sampler Plugin Using the ldmsd\_controller



- Start the “meminfo” sampler with a 1 second (1,000,000us) 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:** “interval” defines the number of micro-seconds between successive samples

**Note 2:** “offset” defines micro-seconds after the second

# Query Current Metric Values After Starting The “meminfo” Sampler Plugin



```
$ ldms_ls -x sock -p 10001 -l node-1/meminfo
```

node-1/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 are being collected



# 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: 184380 kB

Inactive: 393064 kB

Active(anon): 125324 kB

Inactive(anon): 284680 kB

Active(file): 59128 kB



## EXAMPLE: “meminfo” Sampler Plugin



Please see [Meminfo Sampler Daemon](#) to view a video example of Exercise 1 (slides 30-33).

# 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)

## EXAMPLE: Change Sample Interval

Please see [Change Sample Interval for Meminfo](#) to view a video example of Exercise 1 (slide 37).

# 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 ?  
sock:10001 -S samplerd.sock
```

```
$kill 3582
```

- Check to make sure it is dead

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

```
ss1 12:51 0:00 ldmsd -x
```

# 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 ~/ldmscon2021/basic/.../conf/E1/simple_sampler.conf
```

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

```
load name=meminfo  
config name=meminfo producer=<$HOSTNAME> instance=<$HOSTNAME>/meminfo  
component_id=<host number>  
start name=meminfo interval=1000000 offset=0
```

- 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>/.../logs/sampler1.log  
-c /home/<user>/.../conf/E1/simple_sampler.conf
```

# Query The Metric Values: The “meminfo” Sampler Is Configured And Running



```
$ ldms_ls -x sock -p 10001 -l node-1/meminfo
```

node-1/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 are being collected*

# Multiple Sampler Plugins Running on a Single ldmsd



- Edit and uncomment the lines for the vmstat plugin in simple\_sampler.conf then kill and restart your ldmsd using -c.
- Alternatively modify your previously created file with the additional contents:

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

- Query the ldmsd using ldms\_ls:

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

node-1/vmstat

node-1/meminfo



## EXAMPLE: Multiple Sampler Plugins

Please see [Multiple Plugin Sampler Daemon](#) to view a video example of Exercise 1 (slides 37-40).

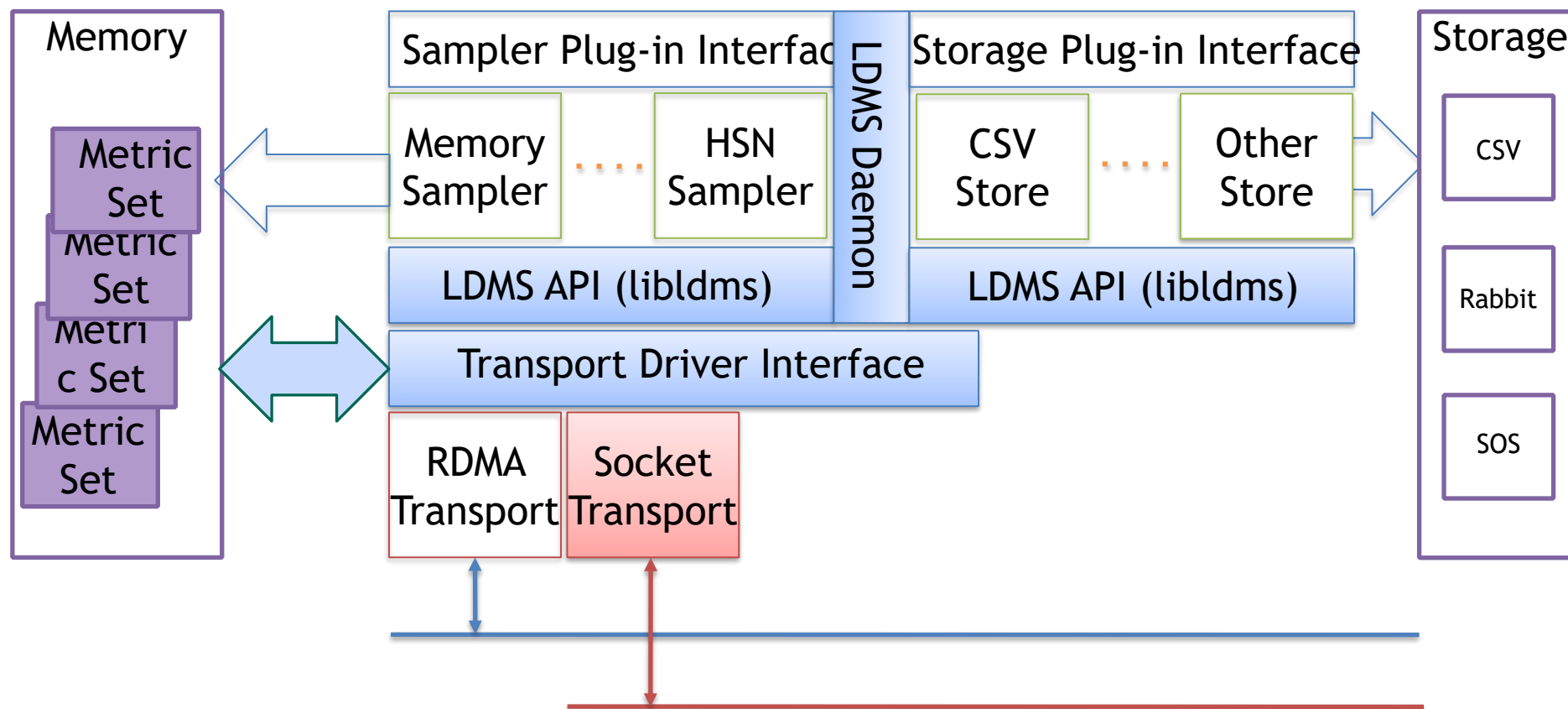
# Configuration Methods Summary



- Dynamic/manual configuration using `ldmsd_controller`
  - `ldmsd_controller` is a Python script that can connect to a `ldmsd` via a configured network socket (supports command completion)
- Static configuration via configuration file
  - Configuration file – loaded at `ldmsd` run time

## Exercise 2: Configure LDMS 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 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



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

```
$ldmsd -x sock:20001 -l ~/ldmscon2021/basic/exercises/ldms/logs/agg1.log
```

-x: Transport : listening port

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

**NOTE:** We are using **10001** for our sampler and port **20001** for our aggregator.

# Interactive Aggregator Configuration Using the ldmsd\_controller



- 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 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 of the sampler daemon that this aggregator is connecting to (e.g., node-1)

**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 update 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	node-1	10001	sock	CONNECTED
--------	--------	-------	------	-----------

node-1/meminfo meminfo

node-1/vmstat vmstat

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 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.

## EXAMPLE: Simple Aggregator Producer Configuration



Please see the simple [Aggregator Producer Configuration](#) to view a video example of Exercise 2 (slides 46-50).

# Configure and Start Aggregator Updater Policy



- 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 associated prdcr every 1 seconds

**offset:** Target (in us) from <epoc sec>.000000

- Example: offset=200000 means aggregate every <interval> seconds at 200ms into the second.
  - To prevent pulling when sampler is updating.

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

- prdcr1 in this case (see slide 50)

# Check Aggregator Status

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



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

Name	Type	Interval	Offset	Libpath
prdcr1	node-1	10001	sock	CONNECTED
	node-1/meminfo	meminfo	READY	
	node-1/vmstat	vmstat	READY	

Name	Interval	Offset	Mode	State
updtr1	1000000	0	Pull	RUNNING
prdcr1	node-1	10001	sock	CONNECTED

Name	Container	Schema	Plugin	State
------	-----------	--------	--------	-------

# Query Current Metric Values On The Aggregator



```
$ldms_ls -h localhost -x sock -p 20001 -l node-1/meminfo
```

node-1/meminfo: consistent, last update: Wed Oct 09 18:30:49 2021 -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

## EXAMPLE: Simple Aggregator Updater Configuration

Please see [Aggregator Updater Configuration](#) to view a video example of Exercise 2 (slides 52-54).

# 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 ~/ldmscon2021/basic/exercises/ldms/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=$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 ldmsd and restart it using this configuration file

```
$ldmsd -x sock:20001 -l ~/ldmscon2021/basic/exercises/ldms/logs/aggd.log
-c ~/ldmscon2021/basic/exercises/ldms/conf/E2/simple_agg.conf
```



# Query Current Metric Values On The Aggregator



```
$ldms_ls -x sock -p 20001 -l node-1/meminfo
```

node-1/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 video example of Exercise 2 (slides 56-57).

# **Exercise 3: Aggregating From Remote Hosts: Building a Distributed Monitoring System**



- ```
$cat /home/<user>/exercises/ldms/conf/E3/agg.conf
```

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

|                                                        |                                              |
|--------------------------------------------------------|----------------------------------------------|
| updtr_add name=updtr1 interval=1000000 offset=200000   | # <i>UPDATE AT <b>1 SECOND</b> INTERVALS</i> |
| 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 with POLICY “updtr1”         |

|                                                       |                                       |
|-------------------------------------------------------|---------------------------------------|
| updtr_add name=updtr2 interval=2000000 offset=200000  | # UPDATE AT <b>2 SECOND</b> 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 with POLICY "updtr2"  |

# Aggregate From Other Student Samplers (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/E3/agg.conf
```

**Alternatively** create this file by copying 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 | 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 |
|----------------|---------|--|--|-------|

|               |        |  |  |       |
|---------------|--------|--|--|-------|
| node-2/vmstat | vmstat |  |  | READY |
|---------------|--------|--|--|-------|

|        |        |       |      |              |
|--------|--------|-------|------|--------------|
| prdcr3 | node-3 | 10001 | sock | DISCONNECTED |
|--------|--------|-------|------|--------------|

| Name | Interval | Offset | Mode | State |
|------|----------|--------|------|-------|
|------|----------|--------|------|-------|

|        |         |        |      |         |
|--------|---------|--------|------|---------|
| updtr1 | 1000000 | 200000 | Pull | RUNNING |
|--------|---------|--------|------|---------|

|        |        |       |      |           |
|--------|--------|-------|------|-----------|
| prdcr1 | node-1 | 10001 | sock | CONNECTED |
|--------|--------|-------|------|-----------|

|        |        |       |      |           |
|--------|--------|-------|------|-----------|
| prdcr2 | node-2 | 10001 | sock | CONNECTED |
|--------|--------|-------|------|-----------|

|        |        |       |      |              |
|--------|--------|-------|------|--------------|
| prdcr3 | node-3 | 10001 | sock | DISCONNECTED |
|--------|--------|-------|------|--------------|

| Name | Container | Schema | Plugin | State |
|------|-----------|--------|--------|-------|
|------|-----------|--------|--------|-------|

## EXAMPLE: Aggregate From Other Student Samplers

Please see [Aggregate From Multiple Containers](#) to view a similar video example of Exercise 3.

# Exercise 4: Basic Dynamic Configurations and Resilience



# Basic Dynamic Configuration Changes



## Exercise Goals:

- Explore dynamic configuration options
  - Sampler daemons (from Exercise 1 - slide 35)
    - Stopping sampler plugins
    - Starting sampler plugins with different intervals
  - Aggregator daemons
    - Automatic detection of new metric sets on connected sampler Idmsd
    - Stopping producer (prdcr) and updater (updtr) policies
    - Changing updater intervals

# Dynamically Changing a Sampler Plugin's Interval Parameters (Exercise 1 - slide 35)



- 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 changes in samplers will be discovered by the aggregating Idmsd

- **Exercise** – one student will add the vmstat sampler, using Idmsd\_controller, to their running sampler Idmsd.
  - All others will see it, using Idms\_ls, appear in their aggregators which are pulling from that sampler.
- **Exercise** – one student will first stop their meminfo sampler, using Idmsd\_controller, on their running sampler Idmsd and then remove (term) their meminfo sampler
  - All others will see, using Idms\_ls, that the timestamp in that student's metric set ceases to update
  - Upon removal all other students will see that metric set disappear from their list of metric sets
- **Exercise** – the same student will restart their meminfo sampler, using Idmsd\_controller, on their running Idmsd.
  - All others will see, using Idms\_ls, that the timestamp in that student's metric set resumes updating.

Note: updtr policies may preclude updating new metric sets e.g., match=schema regex=vmstat would not match a new schema "foo"

# Dynamic Changes and Aggregator Robustness (cont'd)



Samplers and Aggregators can be started in any order

- **Exercise** – Use your modified configuration files to start the aggregator Idmsd before starting the sampler Idmsd
  - Use Idms\_Is to convince yourself that, whether a sampler Idmsd is started before or after an aggregator Idmsd, you are able to see the data generated at the sampler Idmsd on the aggregator Idmsd when both are running

LDMS collection and transport are robust to Samplers and Aggregators being killed and restarted

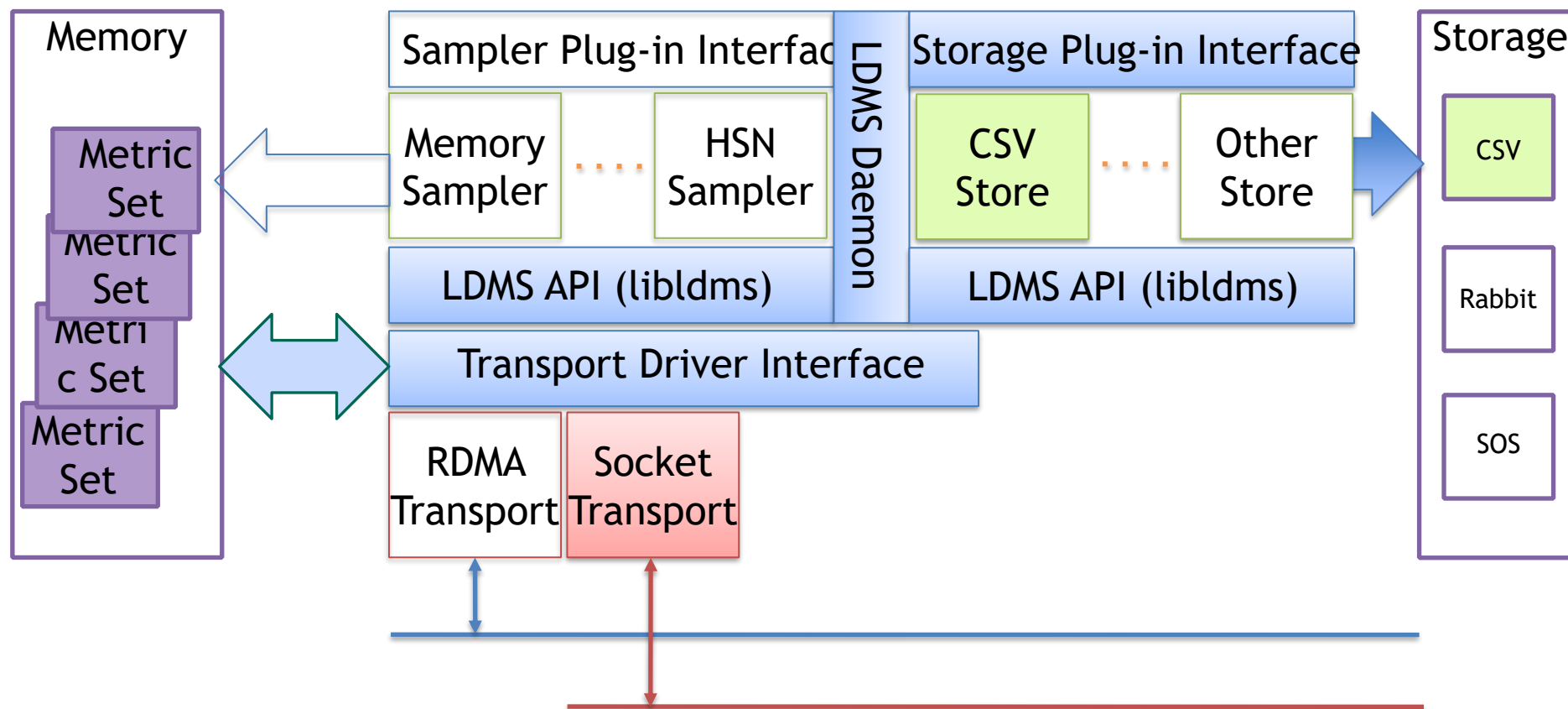
- **Exercise** – one student will kill their sampler Idmsd. All other students will see from Idms\_Is timestamp that the student's metric set is removed from the list.
- **Exercise** – the same student will restart their sampler Idmsd. All other students will see from Idms\_Is timestamp that the metric 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 Idmsd and verify, using Idms\_Is, that they are able to see appropriate data.

## EXAMPLE: Dynamic Changes and Aggregator Robustness

Please see [Dynamic Changes and Aggregator Robustness](#) to view a video example of Exercise 4.

# Exercise 5: Storing Data In CSV Format

# LDMS Plugin Architecture



# Storing Data: CSV Store Plugin



## Goals:

- Configure an aggregator Idmsd with a CSV store plugin using Idmsd\_controller
- Configure an aggregator Idmsd with a CSV store plugin using a configuration file
- Minimal store options (don't buffer data)
- **Note:** The scripts/E1 – E5 directories contain scripts to start Idmsd 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-3,3,0,1884188,571028,1688632,0,1212004,6108,104536,1122496,8276,8580,96260,1113916,0,0,839676,793956,420,0,10552,24812,179652124,40104,12020,1792,3280,0,0,0,1781768,387984,34359738367,7216,34359728128,0,2048,0,0,0,0,2048,47040,2050048
```

```
1487105963.002583,2583,node-10,10,0,1884188,1665280,1671132,948,107512,0,71540,80920,44128,8308,27412,72612,0,0,839676,839676,0,0,44000,22264,8436,35680,24304,11376,1600,2940,0,0,0,1781768,296444,34359738367,7216,34359728128,0,6144,0,0,0,0,2048,34752,2062336
```

```
1487105963.001964,1964,node-12,12,0,1884188,1623168,1644996,948,129700,0,89312,101956,60788,8332,28524,93624,0,0,839676,839676,0,0,60620,23912,8500,36456,24608,11848,1872,4364,0,0,0,1781768,403252,34359738367,7216,34359728128,0,16384,0,0,0,0,2048,44992,2052096
```



# CSV Store: Manual Aggregator Configuration



- 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

```
$ldmsd_controller --host localhost --xprt sock --port 20001
```

```
sock:localhost:20001> load name=store_csv
```

```
sock:localhost:20001> config name=store_csv \  
path=/home/user1/ldmscon2021/basic/exercises/ldms/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 **# USE WITH CAUTION as this will limit performance as scale increases!**

**man page:**

```
$man Plugin_store_csv
```

# opens store\_csv plugin man pages

# CSV Store: Cont.



- Check status

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

| Name  | Type  | Interval | Offset | Libpath                                   |
|-------|-------|----------|--------|-------------------------------------------|
| ----- |       |          |        |                                           |
| csv   | store | 1000000  | 0      | /opt/ovis/lib64/ovis-ldms/libstore_csv.so |

| Name  | Host | Port | Transport | State |
|-------|------|------|-----------|-------|
| ----- |      |      |           |       |

|       |        |       |      |           |
|-------|--------|-------|------|-----------|
| prdc1 | node-1 | 10001 | sock | CONNECTED |
|-------|--------|-------|------|-----------|

|                |         |       |
|----------------|---------|-------|
| node-1/meminfo | meminfo | READY |
|----------------|---------|-------|

|               |        |       |
|---------------|--------|-------|
| node-1/vmstat | vmstat | READY |
|---------------|--------|-------|

| Name  | Interval:Offset | Auto | Mode | State |
|-------|-----------------|------|------|-------|
| ----- |                 |      |      |       |

|        |              |       |      |         |
|--------|--------------|-------|------|---------|
| updtr1 | 1.0s:200.0ms | false | Pull | RUNNING |
|--------|--------------|-------|------|---------|

|       |        |       |      |           |
|-------|--------|-------|------|-----------|
| prdc1 | node-1 | 10001 | sock | CONNECTED |
|-------|--------|-------|------|-----------|

| Name  | Container | Schema | Plugin | flush(sec) | State |
|-------|-----------|--------|--------|------------|-------|
| ----- |           |        |        |            |       |

# 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.

**schema:** metric set schema to be stored

# CSV Store: Cont.



- Check status

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

| Name | Type  | Interval | Offset | Libpath                                   |
|------|-------|----------|--------|-------------------------------------------|
| csv  | store | 1000000  | 0      | /opt/ovis/lib64/ovis-ldms/libstore_csv.so |

| Name           | Host    | Port  | Transport | State     |
|----------------|---------|-------|-----------|-----------|
| prdcrl         | node-1  | 10001 | sock      | CONNECTED |
| node-1/meminfo | meminfo |       | READY     |           |
| node-1/vmstat  | vmstat  |       | READY     |           |

| Name   | Interval:Offset | Auto  | Mode | State     |
|--------|-----------------|-------|------|-----------|
| updtrl | 1.0s:200.0ms    | false | Pull | RUNNING   |
| prdcrl | node-1          | 10001 | sock | CONNECTED |

| Name              | Container      | Schema  | Plugin    | flush(sec) | State   |
|-------------------|----------------|---------|-----------|------------|---------|
| meminfo-store_csv | memory_metrics | meminfo | store_csv | 0.000000   | STOPPED |
| producers:        |                |         |           |            |         |
| metrics:          |                |         |           |            |         |

# CSV Store: Continued



- Start meminfo-store\_csv policy

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

**name:** storage policy tag

# CSV Store: Cont.



- Check status

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

| Name | Type  | Interval | Offset | Libpath                                   |
|------|-------|----------|--------|-------------------------------------------|
| csv  | store | 1000000  | 0      | /opt/ovis/lib64/ovis-ldms/libstore_csv.so |

| Name           | Host    | Port  | Transport | State     |
|----------------|---------|-------|-----------|-----------|
| prdcrl         | node-1  | 10001 | sock      | CONNECTED |
| node-1/meminfo | meminfo |       | READY     |           |
| node-1/vmstat  | vmstat  |       | READY     |           |

| Name | Interval:Offset | Auto | Mode | State |
|------|-----------------|------|------|-------|
|------|-----------------|------|------|-------|

|        |              |       |      |           |
|--------|--------------|-------|------|-----------|
| updtr1 | 1.0s:200.0ms | false | Pull | RUNNING   |
| prdcrl | node-1       | 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 ...

# Examining The CSV File



- **Exercise:** Check the CSV file

```
$head /home/user1/ldmscon2021/basic/exercises/ldms/data/memory_metrics/meminfo  
$tail -f /home/user1/ldmscon2021/basic/exercises/ldms/data/memory_metrics/meminfo
```

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

## EXAMPLE: CSV Store - Manual Aggregator Configuration

Please see [Manual CSV Store](#) to view a video example of Exercise 5 (slides 73-79).





# CSV Store: Start and Configure Aggregator Using a Configuration File



- View store relevant part of configuration file at:  
/home/<user>/ldmscon2021/basic/exercises/ldms/conf/E5/agg\_store\_csv.conf

```
load name=store_csv
config name=store_csv path=/home/user1/ldmscon2021/basic/exercises/ldms/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** that this configuration file also stores the **vmstat metric set** in memory\_metrics
- Restart your aggregator using:  
/home/<user>/ldmscon2021/basic/exercises/ldms/conf/E5/agg\_store\_csv.conf
- “tail -f” each of meminfo and vmstat to see metrics being stored

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

Please see [CSV Store Using Configuration File](#) to view a video example of Exercise 5 (slide 81).





Basics End