LDMS Version 3 Tutorial
https://github.com/ovis-hpc/ovis

Jim Brandt, Tom Tucker, Ann Gentile, Nichamon Nasksinehaboon, Narate Taerat
Open Grid Computing, Inc.
Sandia National Laboratories
04/2017
About this document

• This is a sub-selection of materials from an LDMS tutorial. The full tutorial includes VM’s with an LDMS installation. The VM is not here, however the run scripts from the exercises are included.

• If you install LDMS on your system, you can then use these scripts as models and work through the exercises.

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
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

- Publications:
  - https://ovis.ca.sandia.gov
Tutorial Format

Overview of the Lightweight Distributed Metric Service (LDMS)
• Introduction to HPC monitoring
• 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 labs Instructor walk through and facilitated student exploration
• Lab 1: Samplers
  • Sampler startup and local and remote verification
• Lab 2: Aggregators
  • Aggregation startup and verification using sampler
  • Aggregation of all other attendees’ samplers
• Lab 3: Dynamic configurations and resilience
• Lab 4: Storing data in CSV stores
• Lab 5: Calculating derived data and saving to a CSV store
• Lab 6: Storing the data in an SOS database
• Lab 7: Exploring data in an SOS database
• Lab 8: Data analysis and Visualization from an SOS database

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
Introduction to HPC Monitoring

• Canonical Monitoring Goal: Real-time troubleshooting (e.g., nodes down, out of memory, resource congestion)

• HPC monitoring concerns:
  • Impact on running applications
  • How to aggregate data from different sources for analysis.
    • Network, filesystem, CPU utilization, memory utilization
  • What analyses would be meaningful.
    • e.g., What raw and derived data would indicate performance-impacting network congestion.
  • How to process large amounts of data in real-time

• As a result, canonical system monitoring:
  • Typically performed at intervals of minutes
  • Analyses largely consists of detecting monitoring values exceeding pre-defined thresholds
  • Data is unsuitable for gaining significant insights into application performance problems
Monitoring Can Enable Resource-Aware Computing

Lightweight high-frequency continuous run-time monitoring, analysis, and feedback could enable:

- Faster problem detection, including component-specific issues based on a particular component’s known behaviors and environment (e.g., thermal variations)
- Insight into a large-scale application’s use of resources under production conditions, including contention from other applications
- Dynamic application-to-resource mapping based on application needs and system state
- Co-scheduling of applications based on contention for shared resources
- Dynamic system operations based on a data center’s power demands, temperature etc.
LDMS Overview

• What is the Lightweight Distributed Metric System (LDMS)?
  • Collect numeric data
  • Move and aggregate data
  • Store data
  • Analyze data
    • Troubleshooting
    • Optimization
    • Inform future designs

• Typical use case descriptions

• Supported technologies
  • Linux on all but IBM Blue Gene platforms

• Sources of code, information, and support
Lightweight Distributed Metric Service (LDMS) High Level Overview

* Only the current data is retained on-node
LDMS Plugin Architecture

Memory
- Metric Set
- Metric Set
- Metric Set
- Metric Set

Sampler Plug-in Interface
- Memory Sampler
- HSN Sampler

Transport Driver Interface
- RDMA Transport
- Socket Transport

LDMS API (libldms)

Storage Plug-in Interface
- CSV Store
- Other Store

LDMS API (libldms)

Storage
- CSV
- MySQL
- SOS
Data Flow

Idmsd (sampler)

Idmsd sampler api

{3} sample()
{1} config()

sampler plugin

{4} set_metric()
{2} create_set()

ldms_set_metric()
ldms_create_set()

Metric Set
nr_pages
nr_free_pages...

(on aggregator update)
(on aggregator lookup)

Idms_xprt

Idms (aggregator)

metric update thread

{a} {e}
{id} store()

{d} lookup_complete cb()
{h}

ldms_lookup()
ldms_update()

ldms_lookup()
ldms_update()

create set on
lookup success

Metric Set
nr_pages
nr_free_pages...

(on update complete)

Idms_xprt

Idms store api

update_complete cb()
Supported platforms and networks

• Platforms
  • Rhel 6 and 7
  • SLES 11 & 12
  • Ubuntu
  • Cray XE6, XK and XC

• Transports
  • Socket
  • Cray ugni
    • Aries
    • Gemini
  • RDMA
    • Infiniband
    • iWarp
Build dependencies

• Typical compute node environment
  • Autoconf >=2.3, automake, autotool
  • Libevent2-devel >=2.0.31
  • OpenSSH-devel

• End use hosts (monitor cluster, special aggregation hosts, etc.)
  • Python
    • 2.6 with the argparse module
    • 2.7
  • Swig
  • Doxygen for documentation
LDMS Installation methods

• Manually install using autoconf and automake
• Deployment using RPM

Note: For this demo, LDMS is pre-installed on student VMs in /opt/ovis.

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
Getting started: Log in and set up your environment

```
ssh -Y ovis_public@XXXXXXX
$ ovis_public@XXXXXX's password: ******

ovis_public@ovis-demo-login ~  [sshd:]
$ ssh -Y ovis_public@ovis-demo-01
```

Note: “/home/ovis_public/demo/ldmsd/env/ldms-env.sh” is used to set up LDMS environment

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball.
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:
  - source-code/
    - ldms/ source code of LDMS latest release version
    - util/ utility codes for use in the examples
  - data/ Pre-collected numeric data and log message data
    - ldms-data/ Released numeric data from NCSA BlueWaters
      - csv A subset of Blue Waters data in the CSV format
  - demo/
    - ldmsd/
      - conf/ Configuration files used in the LDMS demo
      - data/ Place holders for the to-be-stored LDMS data
      - env/ Scripts to setup environment variables
      - scripts/ Helper scripts to deploying LDMS daemons

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball.
Getting started: Set up and verify your Environment

- **System env. var.**
  - \( \text{PATH} = \{\text{OVIS_HOME}\}/\text{bin}:/\{\text{OVIS_HOME}\}/\text{sbin}:/\{\text{PATH}\} \)
  - \( \text{LD_LIBRARY_PATH} = \{\text{OVIS_HOME}\}/\text{lib}:/\{\text{LD_LIBRARY_PATH}\} \)
  - \( \text{PYTHONPATH} = \{\text{OVIS_HOME}\}/\text{lib}:/\text{python2.7}/\text{site-packages}:/\{\text{PYTHONPATH}\} \)

- **LDMS env. var.**
  - \( \text{ZAP_LIBPATH} = \{\text{OVIS_HOME}\}/\text{lib}:/\text{ovis-lib} \)
  - \( \text{LDMSD_PLUGIN_LIBPATH} = \{\text{OVIS_HOME}\}/\text{lib}:/\text{ovis-ldms} \)

- **LDMS authentication**
  - \( \text{LDMS_AUTH_FILE} = \langle\text{path to file with your shared secret}\rangle \)
    - Permissions 600
    - Format: secretword=<8 or more characters> (e.g. secretword=mylittlesecret)

**NOTE:** \( \{\text{OVIS_HOME}\} = /\text{opt}/\text{ovis} \) in this example

*Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball*
Test code: memeater.c

- Memeater code which repeatedly allocs mem. Run with LDMS to see changes in memory utilization values reported in /proc/meminfo.
- Located at /home/ovis_public/source-code/util/memeater.c. Compile with cc.

Periodically increase memory allocated

Sleep between alloc. Change this wrt sampling frequency.

Sleep before releasing memory
Lab Exercises

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
LAB 1: Samplers

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
Start and configure a LDMS daemon

Lab Goals:

• Basic LDMS daemon startup and configuration flags/args
  • Manual and run-time configuration options
  • Output options
    • Log files and log levels
    • Debug information
  • man pages
    • man /opt/ovis/share/man/man8/ldmsd.8 – opens ldmsd man pages
    • man /opt/ovis/share/man/man8/ldmsd_controller.8 – opens “ldmsd_controller” man pages

• Use of ldms_ls utility as a diagnostic tool
  • man pages
    • man /opt/ovis/share/man/man8/ldms_ls.8 – opens ldms_ls man pages
LDMS Plugin Architecture
Start a LDMS daemon

• Start ldmsd

```
ldmsd -x sock:10001 -l sampled.log -S sampled.sock -r sampled.pid -p 20001
```

- **-x:** Transport: listening port
- **-l:** Specify the log file path and name
- **-S:** Specify the Unix domain socket for communication with ldmsctl or ldmsd_controller
- **-r:** Specify where to write the pid file
- **-p:** Specify the listener port for remote configuration

Note: The log and Unix domain socket names are just strings. We use “samplerd” here to denote those being used by a ldmsd that will be running “samplers” as opposed to performing aggregation.
Check to see if ldmsd is running

- 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 -S samplerd.sock” 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
- Returns: blank line if the ldmsd specified exists but has no metric sets configured
Exercise: Run ldmsd

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
Manually load and configure a sampler plugin

Lab Goals:

• Basic sampler plugin operation
  • Manual dynamic configuration using the “ldmsd_controller” utility
  • Static configuration using a configuration file
  • man pages
    • man /opt/ovis/share/man/man7/Plugin_meminfo.7 – opens meminfo plugin man pages
    • man /opt/ovis/share/man/man7/Plugin_vmstat.7 – opens vmstat plugin man pages

• Use of ldms_ls utility as a diagnostic tool
  • man pages
    • man /opt/ovis/share/man/man8/ldms_ls.8 – opens ldms_ls man pages
Configure LDMS daemon Sampler Plugin(s)

- 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
Connect ldmsd_controller to an ldmsd

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

```
$ldmsd_controller --host localhost --port 20001
--auth_file ~/.ldmsauth.conf
```

Welcome to the LDMSD control processor
localhost:20001>
Exercise: Connect to ldmsd with ldmsd_controller

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
LDMS Plugin Architecture

- Memory Sampler
- HSN Sampler
- Memory Set
- RDMA Transport
- Socket Transport
- CSV Store
- Other Store
- CSV
- MySQL
- SOS

Diagram shows the architecture with layers for memory, sampler plug-in interface, transport driver interface, storage plug-in interface, and storage.
Interactive Configuration using the ldmsd_controller

• Load the “meminfo” sampler

    localhost:20001> load name=meminfo

• Configure the “meminfo” sampler

    localhost:20001> config name=meminfo
      producer=<$HOSTNAME>
      instance=<$HOSTNAME>/meminfo
      component_id=<host 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

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

```plaintext
ovis-demo-01/meminfo: inconsistent, last update: Wed Dec 31 18:00:00 1969 [0us]
METADATA --------
  Producer Name : ovis-demo-01
  Instance Name : ovis-demo-01/meminfo
  Schema Name : meminfo
  Size : 1904
  Metric Count : 45
  GN : 2
DATA ------------
  Timestamp : Wed Dec 31 18:00:00 1969 [0us]
  Duration : [0.000000s]
  Consistent : FALSE
  Size : 400
  GN : 1
```
Query current metric values before starting the “meminfo” sampler

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

ovis-demo-01/meminfo: inconsistent, last update: Wed Dec 31 18:00:00 1969 [0us]

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_id</td>
<td>1</td>
</tr>
<tr>
<td>job_id</td>
<td>0</td>
</tr>
<tr>
<td>MemTotal</td>
<td>0</td>
</tr>
<tr>
<td>MemFree</td>
<td>0</td>
</tr>
<tr>
<td>MemAvailable</td>
<td>0</td>
</tr>
<tr>
<td>Buffers</td>
<td>0</td>
</tr>
<tr>
<td>Cached</td>
<td>0</td>
</tr>
<tr>
<td>SwapCached</td>
<td>0</td>
</tr>
<tr>
<td>Active</td>
<td>0</td>
</tr>
<tr>
<td>Inactive</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Start the “meminfo” sampler

• Start the “meminfo” sampler

    localhost:20001> start name=meminfo interval=1000000 offset=0

• This starts the sampler updating the metric values every 1 second
Get the set information

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


METADATA --------
  Producer Name : ovis-demo-01
  Instance Name : ovis-demo-01/meminfo
  Schema Name : meminfo
  Size : 1904
  Metric Count : 45
  GN : 2

DATA -----------
  Timestamp : Fri Feb 10 12:46:55 2017 [3486us]
  Duration : [0.000068s]
  Consistent : TRUE
  Size : 400
  GN : 259

--------------
Query current metric values

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

ovis-demo-01/meminfo: consistent, last update: Fri Feb 10 12:50:25 2017

| M u64  | component_id | 1 |
| D u64  | job_id       | 0 |
| D u64  | MemTotal     | 1884188 |
| D u64  | MemFree      | 828244  |
| D u64  | MemAvailable | 1639232 |
| D u64  | Buffers      | 948     |
| D u64  | Cached       | 915992  |
| D u64  | SwapCached   | 0       |
| D u64  | Active       | 84336   |
| D u64  | Inactive     | 891196  |

...
Check source for reference

$ cat /proc/meminfo
MemTotal:  1884188 kB
MemFree:   828420 kB
MemAvailable:  1639912 kB
Buffers:   948 kB
Cached:    916396 kB
SwapCached: 0 kB
Active:    85144 kB
Inactive:  890212 kB
Active(anon):  58272 kB
Inactive(anon):  8372 kB
Active(file):  26872 kB
Inactive(file):  881840 kB
Exercise: Manual sampler configuration

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
• Kill all of your ldmsd in preparation for the next section
  
  ```bash
  $ pkill ldmsd
  ```

• Kill a particular ldmsd
  
  ```bash
  • ps auxw | grep ldmsd | grep -v grep
  ovisPu+ 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
  
  ```bash
  $ ps auxw | grep ldmsd | grep -v grep
  ```
Start ldmsd and sampler plugin using a configuration file

• 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

• Sample configuration file for meminfo example:

```bash
$cat /home/ovis_public/demo/ldmsd/conf/simple_sampler.conf
load name=meminfo
config name=meminfo producer=<$HOSTNAME> instance=<$HOSTNAME>/meminfo
component_id=<host number>
start name=meminfo interval=1000000
```

• Run ldmsd using this configuration file

```bash
$ldmsd -x sock:10001 -l samplerd.log -S samplerd.sock -c
/home/ovis_public/demo/ldmsd/conf/simple_sampler.conf
```
Query current metric values

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

M u64   component_id       1
D u64   job_id             0
D u64   MemTotal           1884188
D u64   MemFree            828244
D u64   MemAvailable       1639232
D u64   Buffers            948
D u64   Cached             915992
D u64   SwapCached         0
D u64   Active             84336
D u64   Inactive           891196
...
```
Exercise: Static sampler configuration

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
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

Static configuration (local)
- Configuration file – loaded at ldmsd run time
Configuration option and tool.

• CMD line configuration –c
• ldmsctl
  • C interface to configure LDMSD.
  • Only for sampler daemon
• ldmsd_controller
  • Python interface to configure LDMSD.
  • Connect to an LDMSD using UNIX domain socket (local) or socket (remote).
  • Auto-completion
  • Command help

• More details can be found at https://www.opengridcomputing.com/wordpress/index.php/ovis-3-3-user-guide/#ldmsd-config
Start ldmsd_controller

• Connect with UNIX domain socket
  ```
  ldmsd_controller --sockname samplerd.sock
  ```

• Connect with socket
  ```
  ldmsd_controller --host localhost --port 20001 --auth_file ~/.ldmsauth.conf
  ```
ldmsd_controller: Get command list

```
samplerd.sock> help

Documented commands (type help <topic>):
----------------------------------------
EOF prdcr_del stop udata version
add prdcr_start store udata_regex
config prdcr_start_regex strgp_add updtr_add
env prdcr_stop strgp_del updtr_del
help prdcr_stop_regex strgp_metric_add updtr_match_add
include quit strgp_metric_del updtr_match_del
info say strgp_prdcr_add updtr_prdcr_add
load shell strgp_prdcr_del updtr_prdcr_del
loglevel source strgp_start updtr_start
logrotate standby strgp_stop updtr_stop
prdcr_add start term usage

Definitely use for samplerd
Definitely use for aggregators
Use to load and config plugin
Get help and daemon status
```
ldmsd_controller: command help

samplerd.sock> 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)
LAB 2: Aggregators

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
LDMS Plugin Architecture

- Memory Sampler Plug-in Interface
  - Memory Sampler
  - HSN Sampler
  - LDMS API (libldms)
  - Transport Driver Interface
    - RDMA Transport
    - Socket Transport
  - LDMS Daemon
- Storage Plug-in Interface
  - CSV Store
  - Other Store
  - LDMS API (libldms)

- Storage
  - CSV
  - MySQL
  - SOS
Configure a LDMS daemon (ldmsd) to Aggregate metric set(s)

Goals:
• Add list of connections to sampler ldmsd’s
• Start the connections
• Create an Update policy
  • How often to get a metric set’s update
  • From which sampler ldmsd’s to aggregate
• Start the Update policy
Start an ldmsd that will be used for aggregation

- Start LDMSD

```
ldmsd -x sock:10002 -m 10M -l aggd.log -S aggd.sock -p 20002
```

- `-x` : transport : listener port
- `-m` : Allocate set memory for aggregated metric sets (default: 512K)
- `-l` : Specify the log file path
- `-S` : Specify “Unix Domain Socket” name used for local configuration
- `-p` : Specify the listener port for remote configuration
Interactive aggregator configuration

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

```
$ ldmsd_controller --host localhost --port 20002 --auth_file ~/.ldmsauth.conf
```

Welcome to the LDMSD control processor
localhost:20002>
Simple Aggregator Configuration

- Configure the aggregator to aggregate the “meminfo” set from the sampler daemon

```
localhost:20002> prdcr_add name=bar host=$HOSTNAME port=10001 xprt=sock
type=active interval=20000000
localhost:2002> prdcr_start name=bar
```

- name: policy tag
- host: hostname of the sampler daemon
- port: Listener port of the sampler daemon
- xprt: Transport the sampler daemon listens on
- type: Always “active”
- interval: Re-connect interval
Plugin status  (on agg after started prdcr but before updtr)

```
localhost:20002> status
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Host</th>
<th>Port</th>
<th>Transport</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>localhost</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-demo-i03/meminfo</td>
<td>meminfo</td>
<td></td>
<td></td>
<td>READY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Interval</th>
<th>Offset</th>
<th>State</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Container</th>
<th>Schema</th>
<th>Plugin</th>
<th>State</th>
</tr>
</thead>
</table>
Query current metric values on the aggregator

```bash
$ ldms_ls -h localhost -x sock -p 10002 -l
```

ovis-demo-01/meminfo: inconsistent, last update: Wed Dec 31 18:00:00 1969 [0us]

- M u64 component_id: 1
- D u64 job_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

...
Simple Aggregator Configuration

- Configure the aggregator to **update** the “meminfo” set

```bash
localhost:20002> updtr_add name=foo interval=1000000 offset=200000
localhost:20002> updtr_prdcr_add name=foo regex=.*
localhost:20002> updtr_start name=foo
```

- **name:** policy tag
- **interval:** update interval (in usec)
  - Example: `interval=1000000` means aggregate 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)
Plugin status  (on aggregator after started prdcr and updtr)

<table>
<thead>
<tr>
<th>Name</th>
<th>Host</th>
<th>Port</th>
<th>Transport</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>localhost</td>
<td>localhost</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-demo-i03/meminfo</td>
<td>meminfo</td>
<td></td>
<td>sock</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Interval</th>
<th>Offset</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>1000000</td>
<td>200000</td>
<td>RUNNING</td>
</tr>
<tr>
<td>localhost</td>
<td>localhost</td>
<td>10001</td>
<td>sock</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Container</th>
<th>Schema</th>
<th>Plugin</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Query current metric values on the aggregator

```bash
$ldms_ls -h localhost -x sock -p 10002 -l ovis-demo-01/meminfo
```

```bash
M u64   component_id               1
D u64   job_id                     0
D u64   MemTotal                  1884188
D u64   MemFree                   828244
D u64   MemAvailable              1639232
D u64   Buffers                   948
D u64   Cached                    915992
D u64   SwapCached                0
D u64   Active                    84336
D u64   Inactive                  891196
...
```
Exercise: Validate manual configuration and aggregation from sampler

Note: VM's not in the release materials. Additional configuration scripts in the associated tarball
Start ldmsd and aggregation using a configuration file

• 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

• Sample configuration file for meminfo example:

```bash
$cat /home/ovis_public/demo/ldmsd/conf/simple_aggregator.conf
prdcr_add name=localhost host=$HOSTNAME port=10001 xprt=sock type=active interval=20000000
prdcr_start name=localhost
updtr_add name=foo interval=1000000 offset=200000
updtr_prdcr_add name=foo regex=.*
updtr_start name=foo
```

• Run ldmsd using this configuration file

```bash
$ldmsd -x sock:10002 -l aggd.log -s aggd.sock -c /home/ovis_public/demo/ldmsd/conf/simple_aggregator.conf
```
Query current metric values

```sh
$ldms_ls -x sock -p 10002 -l ovis-demo-01/meminfo
```


```
M u64 component_id 1
D u64 job_id 0
D u64 MemTotal 1884188
D u64 MemFree 828244
D u64 MemAvailable 1639232
D u64 Buffers 948
D u64 Cached 915992
D u64 SwapCached 0
D u64 Active 84336
D u64 Inactive 891196
...
```
Exercise: Validate static aggregator configuration and aggregation from sampler

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
Aggregate from student VMs

• Kill aggregator ldmsd
• Restart ldmsd using “-c students_all_aggregator.conf”
• Kill aggregator ldmsd
• Restart ldmsd using “-c students_subset_aggregator.conf”
## Plugin status

(on aggregator from all students)

```plaintext
localhost:20002> status

<table>
<thead>
<tr>
<th>Name</th>
<th>Host</th>
<th>Port</th>
<th>Transport</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovis-demo-01</td>
<td>ovis-demo-01</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-demo-01/meminfo</td>
<td>meminfo</td>
<td></td>
<td>ready</td>
<td></td>
</tr>
<tr>
<td>ovis-demo-02</td>
<td>ovis-demo-02</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-demo-02/meminfo</td>
<td>meminfo</td>
<td></td>
<td>ready</td>
<td></td>
</tr>
<tr>
<td>ovis-demo-02/vmstat</td>
<td>vmstat</td>
<td></td>
<td>ready</td>
<td></td>
</tr>
<tr>
<td>ovis-demo-03</td>
<td>ovis-demo-03</td>
<td>10001</td>
<td>sock</td>
<td>DISCONNECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ovis-instructor-02</td>
<td>ovis-demo-i02</td>
<td>10001</td>
<td>sock</td>
<td>DISCONNECTED</td>
</tr>
<tr>
<td>ovis-instructor-03</td>
<td>ovis-demo-i03</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-demo-i03/meminfo</td>
<td>meminfo</td>
<td></td>
<td>ready</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Interval</th>
<th>Offset</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>100000</td>
<td>200000</td>
<td>Running</td>
</tr>
<tr>
<td>ovis-instructor-03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ovis-demo-i03</td>
<td>10001</td>
<td>sock</td>
<td>CONNECTED</td>
</tr>
<tr>
<td>ovis-instructor-02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ovis-demo-i02</td>
<td>10001</td>
<td>sock</td>
<td>DISCONNECTED</td>
</tr>
<tr>
<td>ovis-instructor-01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ovis-demo-i01</td>
<td>10001</td>
<td>sock</td>
<td>DISCONNECTED</td>
</tr>
<tr>
<td>ovis-demo-16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ovis-demo-16</td>
<td>10001</td>
<td>sock</td>
<td>DISCONNECTED</td>
</tr>
</tbody>
</table>
```
Exercise: Validate static aggregator configuration and aggregation from sampler

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
LAB 3: Dynamic Changes and Resilience

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
Dynamic Configuration Changes

- Dynamic configuration
  - Sampler daemons
    - stop sampler plugins
    - start with different intervals
  - Aggregator daemons
    - stop prdcr/updtr/strgp
    - remove prdcr/updtr/strgp
    - change interval
Dynamic Changes and Robustness

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

- Samplers and Aggregators can be started in any order
- LDMS collection and transport topologies are robust to Samplers and Aggregators being killed and restarted
  - **Exercise** – one student will kill his ldmsd sampler. All other students will see in ldms_ls timestamp output that that student’s metric set ceases to update
  - **Exercise** – the same student will restart his ldmsd sampler. All other students will see in ldms_ls timestamp output that that student’s metric set resumes updating.
LAB 4: Storing data in CSV stores

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
LDMS Plugin Architecture
Storing data to csv file(s)

 Goals:
• Configure a csv store with ldmsd_controller
• Configure a csv store with configuration file
• Store options

 Example output:

```
#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,Dirt y,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,ovis-demo-09,9,0,1884188,571028,6188632,0,1212004,6108,104536,1122496,8276,8580,96260,1113916,0,0,839676,793956,420,0,10552,24812,1796,52124,40104,12020,1792,3280,0,0,0,1781768,387984,34359738367,7216,34359728128,0,2048,0,0,0,2048,47040,2050048
1487105963.002583,2583,ovis-demo-02,2,0,1884188,1665280,1671132,948,107512,1212496,8276,8580,96260,1113916,0,0,839676,793956,420,0,10552,24812,1796,52124,40104,12020,1792,3280,0,0,0,1781768,296444,34359738367,7216,34359728128,0,6144,0,0,0,0,2048,34752,2062336
1487105963.001964,1964,ovis-demo-08,8,0,1884188,1623168,1644996,948,129700,1212496,8276,8580,96260,1113916,0,0,839676,839676,0,0,60620,23912,8500,36456,24608,11848,1872,4364,0,0,0,1781768,403252,34359738367,7216,34359728128,0,16384,0,0,0,0,2048,34752,2062336
```
Aggregator Configuration to store metric set data using CSV store

- Configure the aggregator to store the “meminfo” set to a csv file using ldmsd_controller
  - Load the store_csv plugin
  - Configure the plugin

```
$ldmsd_controller --host localhost --port 20002 --auth_file ~/.ldmsauth.conf
localhost:20002> load name=store_csv
localhost:20002> config name=store_csv path=/home/ovis_public/demo/ldmsd/data
action=init buffer=0
```

- name: plugin name
- path: Path to the base directory for the csv file container. This directory must pre-exist.
- action: ‘init’ to initialize the plugin (*other actions will not be described in this tutorial*)
- buffer: ‘0’ to disable buffering
- man page:
  - man /opt/ovis/share/man/man7/Plugin_store_csv.7 -- opens store_csv plugin man pages
Aggregator Configuration to store metric set data using **CSV store**

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

```bash
localhost:20002> strgp_add name=meminfo_store_csv plugin=store_csv container=csv schema=meminfo
localhost:20002> strgp_start name=meminfo_store_csv
```

- **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
## Plugin Status

(store info only)

```plaintext
localhost:20002> status

<table>
<thead>
<tr>
<th>Name</th>
<th>Container</th>
<th>Schema</th>
<th>Plugin</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>meminfo_store_csv</td>
<td>csv</td>
<td>meminfo</td>
<td>store_csv</td>
<td>RUNNING</td>
</tr>
</tbody>
</table>

producers:

metrics: 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
```
Examining the CSV file

• The data is saved in:
  /home/ovis_public/demo/ldmsd/data/csv/meminfo

1. Checking the csv file

$ tail –f /home/ovis_public/demo/ldmsd/data/csv/meminfo

• If aggregating from others’ vm’s, see multiple hosts in the output

2. Data changes:

• Run the memeater executable

$ ./a.out

• Compare the live memeater output with the tail –f values
Exercise: Store CSV

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
Start csv store with a configuration file with advanced configuration options

- Aggregator configuration file at:
  /home/ovis_public/demo/ldmsd/conf/agg.conf
  
  ```
  load name=store_csv
  config name=store_csv path=/home/ovis_public/demo/ldmsd/data action=init buffer=0
  rollover=120 rolltype=1 altheader=1
  strgp_add name=meminfo_store_csv schema=meminfo plugin=store_csv container=csv
  strgp_start name=meminfo_store_csv
  ```

- New configuration options:
  - Rollover by time or size:
    - `rollover=120 rolltype=1` – rolls over every 120 sec. Output file is postpended with epoch timestamp (meminfo.12345)
  - Header in a separate file:
    - `altheader=1`
Start csv store with a configuration file with advanced configuration options

• Uncomment the lines for store_csv only (not store_function_csv)
• Kill current aggregator (not the sampler) and Restart aggregator:

```
ldmsd -x sock:10002 -l agg.log -p 20002
    -c /home/ovis_public/demo/ldmsd/conf/agg.conf
```
• Note the file rollover and alternate header
Exercise: CSV store with a configuration file and advanced configuration options

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
LAB 5: Calculating derived data and saving to a CSV store

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
Storing data to store function csv file(s)

Goals:
• Configure a function csv store with ldmsd_controller
• Configure a function csv store with a configuration file
• Function options

Example output:

```
#Time,Time_usec,DT,DT_usec,ProducerName,component_id,job_id,RAW_ACTIVE,RAW_ACTIVE.Flag,RAW_MEMTOTAL,RAW_MEMTOTAL.Flag,RATIO100,RATIO100.Flag,TimeFlag
1487107627.002486,2486,0.999712,999712,ovis-demo-i03,103,0,828068,0,1884188,0,43,0,0
1487107628.002425,2425,0.999939,999939,ovis-demo-i03,103,0,975536,0,1884188,0,51,0,0
1487107629.002402,2402,0.999977,999977,ovis-demo-i03,103,0,975528,0,1884188,0,51,0,0
1487107630.018970,18970,1.016568,16568,ovis-demo-i03,103,0,980228,0,1884188,0,52,0,0
1487107631.002405,2405,0.983435,983435,ovis-demo-i03,103,0,1122996,0,1884188,0,59,0,0
```

Active/Memtotal ratio increasing while memeater runs
Store_function_csv configuration file

Configuration File at /home/ovic_public/demo/ldmsd/conf/fct.conf

# SCHEMA NEW_METRICNAME FUNCTION N_MET <METS_CSV> SCALE|THRESH WRITEOUT

meminfo RAW_ACTIVE RAW 1 Active 1
meminfo RAW_MEMTOTAL RAW 1 MemTotal 1
meminfo RATIO100 DIV_AB 2 RAW_ACTIVE,RAW_MEMTOTAL 100 1

• Functions: RAW (raw value), Scalar and Vector add/subtract/multiply/divide, threshold checks, min/max
  • man page
    • man /opt/ovic/share/man/man7/Plugin_store_function_csv.7 – opens store_function_csv plugin man pages

• Chain variables for a complex computation

• V3 Limitations (addressed in future versions):
  • u64 cast at all steps. Can use scale to keep precision.
  • Functions are only per instance of a metric set (e.g., cannot combine data from meminfo and vmstat, cannot combine info from different components)

• Output flags: Flag for invalid for every computation and for ageusec
Aggregator Configuration to store metric set data using `store_function_csv`

- Configure the aggregator to store derived data from the “meminfo” set to a csv file.

```
$ ldmsd_controller --host localhost --port 20002 --auth_file ~/.ldmsauth.conf
localhost:20002> load name=store_function_csv
localhost:20002> config name=store_function_csv
path=/home/ovis_public/demo/ldmsd/data
buffer=0 ageusec=2000000
derivedconf=/home/ovis_public/demo/ldmsd/conf/fct.conf
```

- **action**: ‘init’ to initialize the plugin
- **derived_conf**: derived configuration file (can take multiples: csv)
- **ageusec**: flag when the DT between data points is greater than this value
Aggregator Configuration to store metric set data using store_function_csv

• Configure the aggregator to **store** derived data from the “meminfo” set to a csv file.

```bash
localhost:20002> strgp_add name=mem_f
plugin=store_function_csv container=csv_fct
schema=meminfo
localhost:20002> strgp_start name=mem_f
```
Plugin Status  (store info only shown)

```
localhost:20002> status

<table>
<thead>
<tr>
<th>Name</th>
<th>Container</th>
<th>Schema</th>
<th>Plugin</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>mem_f</td>
<td>csv_fct</td>
<td>meminfo</td>
<td>store_function_csv</td>
<td>RUNNING</td>
</tr>
</tbody>
</table>

producers:

metrics: component_id job_id MemTotal MemFree MemAvailable Buffers Cached Swap Cached Active Inactive (anon) Inactive (anon) Active (file) Inactive (file) Unevictable Mlocked Swap Total Swap Free Dirty Writeback Anon Pages Mapped Shmem Slab SReclaimable SUnreclaim Kernel Stack Page Tables NFS Unstable Bounce WritebackTmp Commit Limit Committed_AS Vmalloc Total Vmalloc Used Vmalloc Chunk Hardware Corrupted Anon Huge Pages Huge Pages_Total Huge Pages_Free Huge Pages_Rsvd Huge Pages_Surp Huge Pagesize Direct Map 4k Direct Map 2M

meminfo_store_csv csv

producers:

metrics: component_id job_id MemTotal MemFree MemAvailable Buffers Cached Swap Cached Active Inactive (anon) Inactive (anon) Active (file) Inactive (file) Unevictable Mlocked Swap Total Swap Free Dirty Writeback Anon Pages Mapped Shmem Slab SReclaimable SUnreclaim Kernel Stack Page Tables NFS Unstable Bounce WritebackTmp Commit Limit Committed_AS Vmalloc Total Vmalloc Used Vmalloc Chunk Hardware Corrupted Anon Huge Pages Huge Pages_Total Huge Pages_Free Huge Pages_Rsvd Huge Pages_Surp Huge Pagesize Direct Map 4k Direct Map 2M
```
Storing derived data to a function store CSV file

• The data is saved at
  /home/ovis_public/demo/ldmsd/data/csv_fct/meminfo

• Checking the csv_fct file:

  `tail -f /home/ovis_public/demo/ldmsd/data/csv_fct/meminfo`
Exercise: Store_function_csv

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
Storing derived data to a function store CSV file using the ldmsd configuration file

- Uncomment the lines for `store_function_csv` (*store_csv lines are still uncommented*)

- Kill current aggregator (not the sampler) and Restart aggregator:
  ```
  ldmsd -x sock:10002 -l agg.log -p 20002
  -c /home/ovis_public/demo/ldmsd/conf/agg.conf
  ```

- Checking the csv_fct file
  ```
  tail -f /home/ovis_public/demo/ldmsd/data/csv_fct/meminfo
  ```

- Run the memeater code at same time as storing data:
  ```
  ./a.out # the memeater executable
  ```
  compare the live memeater output with the `tail -f` values
Exercise: Store_function_csv with configuration file and memeater

Note: VM’s not in the release materials.
Additional configuration scripts in the associated tarball
LAB 6: Storing the data in an SOS database

Note: VM's not in the release materials.
Additional configuration scripts in the associated tarball
LDMS Plugin Architecture

- **Memory**
  - Metric Set
  - Metric Set
  - Metric Set
  - Metric Set

- **Sampler Plug-in Interface**
  - Memory Sampler
  - HSN Sampler

- **LDMS API (libldms)**

- **Transport Driver Interface**
  - RDMA Transport
  - Socket Transport

- **Storage Plug-in Interface**
  - CSV Store
  - Other Store

- **Storage**
  - CSV
  - MySQL
  - SOS
Configure the aggregator’s **SOS store** plugin

- **Steps:**
  - Load the store_sos plugin
  - Configure the plugin

```bash
localhost:20002> load name=store_sos
localhost:20002> config name=store_sos
path=/home/ovis_public/demo/ldmsd/data/sos
```

- **name**: plugin name
- **path**: Path to the directory to contain the SOS database
Add a storage policy to save the meminfo data to the SOS store

• Configure the aggregator to store the “meminfo” set to a SOS database.

localhost:20002> strgp_add name=meminfo_sos plugin=store_sos container=meminfo schema=meminfo
localhost:20002> strgp_start name=meminfo_sos

• name: storage policy tag
• plugin: store plugin used for storing metric set data
• container: the storage backend container name
• schema: metric set schema to be stored
Use a configuration file to configure the storage back-end

- Edit the configuration file at `~/demo/ldmsd/conf/agg.conf`
  - Uncomment the store_sos configuration lines
- Kill current aggregator (not the sampler)
- Restart the aggregator

```
ldmsd -x sock:10002 -l agg.log -p 20003 \ 
   -c ~/demo/ldmsd/conf/agg.conf
```
LAB 7: Exploring data in an SOS database

Note: VM’s not in the release materials. Additional configuration scripts in the associated tarball
Exercise: Use the SOS tools to explore the database

• sos_cmd
  • Create containers
  • Create and query schema
  • Import and query data

• lmq
  • Plot data stored in the SOS database

• Data visualization on Grafana
Query available schemas in your database

```bash
$ sos_cmd -C /home/ovis_public/demo/ldmsd/data/sos/meminfo/ -l
```

```
schema:
  name : meminfo
  schema_sz : 4504
  obj_sz : 408
  id : 129
  -attribute : timestamp
    type : TIMESTAMP
    idx : 0
    indexed : 1
    offset : 8
  -attribute : MemTotal
    type : UINT64
    idx : 5
    indexed : 0
    offset : 48
  -attribute : MemFree
    type : UINT64
    idx : 6
    indexed : 0
    offset : 56
```
### Query data in the SOS database

**sos_cmd** `-C/home/ovis_public/demo/ldmsd/data/sos/meminfo` \ 
-q -S meminfo -X comp_time -V timestamp –V component_id -V MemFree -V Active | less

<table>
<thead>
<tr>
<th>timestamp</th>
<th>component_id</th>
<th>MemFree</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>1487100290.607418</td>
<td>0</td>
<td>1636160</td>
<td>80120</td>
</tr>
<tr>
<td>1487100300.609416</td>
<td>0</td>
<td>1636160</td>
<td>80120</td>
</tr>
<tr>
<td>1487100310.611474</td>
<td>0</td>
<td>1642688</td>
<td></td>
</tr>
<tr>
<td>1487114607.002163</td>
<td>103</td>
<td>1628516</td>
<td>90320</td>
</tr>
<tr>
<td>1487114608.002077</td>
<td>103</td>
<td>1628516</td>
<td>90320</td>
</tr>
</tbody>
</table>

Records 887636/887636.

- **-q** Query the database
- **-S** Schema name
- **-X** index used to order data
- **-V** once for column in the output
Output the data as a CSV file

```
sos_cmd -C /home/ovis_public/demo/ldmsd/data/sos/meminfo \
    -q -S meminfo -X comp_time-V timestamp –V component_id -V MemFree -V Active -f csv| less
```

# timestamp,component_id,MemFree,Active
1487100290.607418,0,1636160,80120
1487100300.609416,0,1636160,80120
1487100310.611474,0,1642688,76016

... 
1487114606.002196,103,1628548,90320
1487114607.002163,103,1628516,90320
1487114608.002077,103,1628516,90320

# Records 889483/889483.

Records 887636/887636.

-q Query the database
-S Schema name
-X index used to order data
-V once for column in the output
-f csv format the output as CSV
Output the data as a JSON file

```bash
sos_cmd -C /home/ovis_public/demo/ldmsd/data/sos/meminfo \
   -q -S meminfo -X comp_time-V timestamp –V component_id -V MemFree -V Active -f json | less
```

```json
{
   "data": [
      {
         "timestamp": "1487100290.607418", "component_id": "0", "MemFree": "1636160", "Active": "80120"},
      {
         "timestamp": "1487100300.609416", "component_id": "0", "MemFree": "1636160", "Active": "80120"},
      {
         "timestamp": "1487100310.611474", "component_id": "0", "MemFree": "1642688", "Active": "76016"},
      {
         "timestamp": "1487100320.613736", "component_id": "0", "MemFree": "1641272", "Active": "77292"},
      . . .
      {
         "timestamp": "1487114606.002196", "component_id": "103", "MemFree": "1628548", "Active": "90320"},
      {
         "timestamp": "1487114607.002163", "component_id": "103", "MemFree": "1628516", "Active": "90320"},
      {
         "timestamp": "1487114608.002077", "component_id": "103", "MemFree": "1628516", "Active": "90320"},
   ],
   "totalRecords": 890414,
   "recordCount": 890414
}
```

-q Query the database
-S Schema name
-X index used to order data
-V once for column in the output
-f csv format the output as JSON
LAB 8: Data Analysis and Visualization from an SOS database

Note: VM’s not in the release materials. 
Additional configuration scripts in the associated tarball
lmq
LDMS tool to plot time-series graphs
Query range of dates available in the database

```
Imq --path /home/ovis_public/demo/data/sos/meminfo \  
--query dates --schema meminfo
```

There are data available from 02/13/17 14:47:44 (1487018864.002345) through 02/15/17 21:12:21 (1487214741.002282)

--path     The path to the container
--query    What is being queried
--schema   The schema to query
Exercise: Plot time-series graph of a metric

```
$lmq --path ~/demo/ldmsd/data/sos/meminfo --query data --schema meminfo \ 
   --metric_name MemFree --component_id 2
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--path</td>
<td>The path to the container</td>
</tr>
<tr>
<td>--query</td>
<td>What is being queried</td>
</tr>
<tr>
<td>--schema</td>
<td>The schema to query</td>
</tr>
<tr>
<td>--metric_name</td>
<td>The metric data to plot</td>
</tr>
<tr>
<td>--component_id</td>
<td>The component data to plot</td>
</tr>
</tbody>
</table>
lmq plot of MemFee of component 2
Exercise: Plot a graph showing windowed average, and running windowed variance

```
lmq --path ~/demo/ldmsd/data/sos/meminfo --query data --schema meminfo \
    --metric_name current_freemem --component_id 2 --bollinger
```

--path          The path to the container
--query         What is being queried
--schema        The schema to query
--metric_name   The metric data to plot
--component_id  The component data to plot
--bollinger     Plot Bollinger bands and outliers
lmq plot of MemFree of component 2

current.freemem – Bollinger Band (window of 60 time units and bands of 2 x sd)