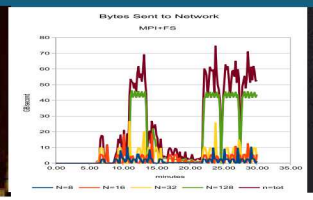
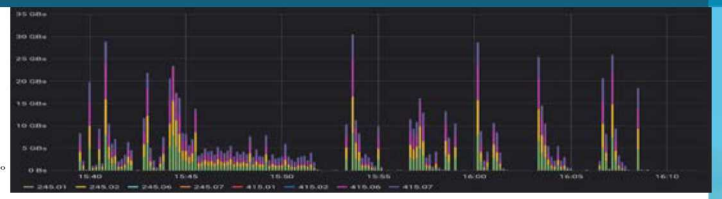
This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2020-8015C

Sandia National Laboratories

Production LDMS, genders, systemd, and the future





LDMSCON 2020

Benjamin Allan

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.

Outline

- A little background
- What it takes to run ldmsd in production
- Goals of the genders-based systemd implementation
- What is genders?
- Example of samplers configuration
- Examples of aggregators
- How to do the hard things: trap doors
- What it doesn’t do yet
- What shall we do about these?

A detailed LDMS v3 systemd/genders case

- https://github.com/baallan/distribution/blob/master/gendersTutorial.md
- Only the highlights today
- The genders configuration for LDMS has not been demonstrated in CLE6/7 yet because until recently, genders library packages were not readily available from Cray. (Kevin Stroup resolved this for Sandia machines)
- LDMSD must be built with “--enable-libgenders --enable-genderssystemd”

Running LDMSD in production

- It is a system service supporting multiple instances
  - ldmsd (sampling daemon)
  - ldmsd@agg (aggregator daemon, level one)
  - ldmsd@$CLUSTER (aggregator per cluster on level two host(s))

- Should be easy to find out how it was last started (even after dead)
  - Capture in/var/run/ldmsd files:
    - daemon configuration file.
    - daemon environment.
    - daemon command line.

- Changing the configuration to enable/disable basic features should be simple i.e. ***declarative***: add a feature to a list, defaults do the rest.

Goals for the genders-based systemd scripts

- Provide automatically:
  - Default behavior for stores, samplers, transport, aggregation.
  - All information possible for diagnosis (whether live or dead).
  - Escape hatches: BYO LDMSD configuration scripting if you do not use genders.
    - We still need ldmsd@.service to handle multiple instances.
  - DRY configuration (don't repeat yourself).
    - Defining an aggregator or storage daemon should not require repeating what was written for upstream sampler daemons (unless exceptions to defaults are needed).
  - Valgrind support in production (as root)
- For administrators:
  - minimize learning of the "ldmsd programming language".
  - minimize configuration changes across across updates.
  - Everything should "just work" in common cases.

What is genders?

- It’s a file format and a library for defining and querying two-level hash tables
  - Typically, the first level of the hash table is unqualified hostname, eg. chama3
  - The second level is attribute/value pairs with specific syntax: values cannot contain whitespace or comma.
  - The attribute syntax can be mapped to ldmsd plugin configuration language “key1=value1 key2=value2,value3,value4” (though it is not beautiful)
  - Attribute meaning is entirely up to application
- e.g.

  ch[1-1232],chgw[1-24] ldmsd,ldmsd_port=411,ldmsd_xprt=sock

  chgw[1-24] ldmsd_metric_plugins=meminfo:vmstat:lnet_stat

  chadmin7 ldmsd_dbg=DEBUG,ldmsd_log=//localdisk/ldmsd/log

Sampler configuration example

```
/etc/sysconfig/ldms.d/ldmsd.local.conf                  # environment: ldmsd
/etc/sysconfig/ldms.d/ClusterGenders/genders.local      # genders
```

```
h1 ldmsd                                                # enable ldms
h1 ldmsd_metric_plugins=meminfo:vmstat:procstat         # list wanted
h1 ldmsd_host=%n-ib0,ldmsd_xprt=rdma                    # net link
```

Defaulted:

interval (10 second), offset (0), producer name logged ($host), schema name (plugin name), instance name ($host/$schema), port(411), component_id(from $host), log, auth method.

Sampler node ldmsd.local.conf

```
LDMS_AUTH_FILE=/etc/sysconfig/ldms.d/ClusterSecrets/ldmsauth.conf
LDMS_GENDERS=/etc/sysconfig/ldms.d/ClusterGenders/genders.local
```

Defines pointers to the files for this systemd instance of ldmsd

- ldmsd.service is in code exactly as ldmsd@local.service

L1 Aggregator configuration example

/etc/sysconfig/ldms.d/ldmsd.L1.conf # environment: ldmsd@L1
/etc/sysconfig/ldms.d/ClusterGenders/genders.L1 # genders for L1

ser[1-187],sergw[1-3],serln1 ldmsd_clientof=seradmin1
ser[188-374],sergw[4-6],serln2 ldmsd_clientof=seradmin2
seradmin[1-2] ldmsaggd=CLIENTOFLIST

Defaulted:

updater policy(all), connection retry interval(2s), agg. interval (10s), agg. offset (0.2 sec), port(411), log, connection definitions, thread count, memory reservation.

Looked up in genders.local (DRY):

transport types, hostnames and ports (374 times).

Aggregator node ldmsd.L1.conf

```
LDMS_AUTH_FILE=/etc/sysconfig/ldms.d/ClusterSecrets/ldmsauth.conf
LDMS_GENDERS=/etc/sysconfig/ldms.d/ClusterGenders/genders.L1
LDMS_GENDERS_1=/etc/sysconfig/ldms.d/ClusterGenders/genders.local
```

Defines pointers to the files for this systemd instance of ldmsd@L1

- Information lookups on genders in $LDMS_GENDERS_1 tell are used to determine connection details for the producer list.

L2 Store configuration example

/etc/sysconfig/ldms.d/ldmsd.L2.conf # environment: ldmsd@L2

/etc/sysconfig/ldms.d/ClusterGenders/genders.L2 # genders for L2

```
mon1 ldmsaggd_offset_default=4200000
mon1 ldmsd_store_plugins=store_csv
mon1 ldmsd_store_csv=altheader/1:rolltype/2:rollover/0:path//scratch_A/${LDMSCLUSTER}_csv:
create_gid/1000000039:create_perm/644:typeheader/2
mon1 ldmsd_schemas_store_csv=meminfo:procstat:vmstat
```

Defaulted:

updater policy(all), connection retry interval(2s), agg. interval (10s), port(411), log, connection definitions, storage policies.

Looked up in genders.L1: transport types, hostnames, and ports.

L2 storage node ldmsd.L2.conf

```
LDMS_AUTH_FILE=/etc/sysconfig/ldms.d/ClusterSecrets/ldmsauth.conf
LDMS_GENDERS=/etc/sysconfig/ldms.d/ClusterGenders/genders.L2
LDMS_GENDERS_1=/etc/sysconfig/ldms.d/ClusterGenders/genders.L1
```

Defines pointers to the files for this systemd instance of ldmsd@L2

- Information lookups on genders in $LDMS_GENDERS_1 tell are used to determine connection details for the producer list.

Doing hard things (1): Scaling & Redundancy

- Redundancy
  - Milly (monitoring host) defines same ldmsd instances as mon1

- Scaling to a supercomputing center with many clusters
  - Mon1 (monitoring host) defines several service instances
    - ldmsd@chama, ldmsd@uno, ldmsd@skybridge, ldmsd@eclipse,... CSV feeds
    - ldmsd@eclipse_Grafana GUI feed

Doing hard things (2): no-genders trap door

- Not using genders database at all?
- ldmsd.local.conf becomes:

```
LDMS_USE_GENDERS=0
LDMSD_PLUGIN_CONFIG_FILE=/etc/ldmsd/ldmsd.local.conf
```

Doing hard things (3): avoid long attributes

- One plugin is just too complex to configure using only genders syntax?
- Use the genders trap door for the plugins:

```
mon1 ldmsd_config_text_store_csv=/etc/ldmsd/csv_config
```

The file csv_config has your ldms plugin special options code:

```
altheader=1 rolltype=2 rollover=0 \
path=/scratch_A/${LDMSCLUSTER}_csv \
create_gid=1000000039 create_perm=644 \
typeheader=2
```

which gets added to the configuration.

Doing hard things (4): complex plugin trap door

- One plugin is just too complex to configure using only genders syntax?
- Use the genders trap door for a configuration generator:

mon1 ldmsd_config_gen_mysampler=mysampler_gen.sh

Program /etc/sysconfig/ldms.d/plugins-conf/mysampler_gen.sh is called with arguments:

$plugin $producer $host $interval $offset

and the output gets added to the configuration.

Dealing with change (v4)

The genders interpreter ***does*** have support for new plugins

- New options and new store/sampler plugins are handled transparently.
- Environment expansion ${VAR} in attributes is handled transparently.
- Plugins all act as singletons, which makes attribute naming easy.

But ***does not yet*** support

- Failover definitions.
- Transport set group definitions.
- Multiple updaters.
- Automatic interval and offset management.
- Push behavior
- Passive mode connections

Dealing with change (v5)

- All plugins changing from singleton objects to potentially many similar instances of “C classes”.
  - Some might still restrict themselves to singleton behavior.
  - Some might have ‘plugins within plugins’ depending on the instance API.
  - Two-layer hash table is likely not a good match for this.

- Still developing what new ldmsd features we will get.
  - Maybe include ***hostlist expansion*** in the ldmsd configuration language to provide for compact expression of aggregation, update, failover.

What shall we do about unsupported v4 features?

- Extending genders processing is a ‘mere matter of scripting’.
  - It requires sensible answers to:

    What should the default behavior be for feature X?
  - Naming the attributes can get a bit messy, however. E.g. now have:
    - ldmsd_strgp_exclude_metrics_meminfo
- Should we refactor the gender interpreter to do less (nothing?), and lean more on a directory full of files that look like .INI/YAML/TOML?
  - This would break compatibility with v2/v3 genders support.
  - This would break the “single file for tuning ldmsd” approach.

Thoughts?

- Share them here
- Post them on the ovis issue tracker
- See also: Chris Morrone's Lively Discussion
  - https://github.com/ovis-hpc/ovis/issues/67

  and
  - https://github.com/ovis-hpc/ovis/wiki/Proposal-2