Lustre Statistics Collection

Table of Contents

- Overview
- Telegraf
- Graphite
- Grafana
- Selinux
- Firewall Rules
- Where to Find Stuff

Overview

Lustre statistics are collected with the Telegraf package, which runs on all of the Lustre servers. Telegraf is configured to send Lustre metrics every 30 seconds to a server running Grafana and Graphite. This document describes the relevant configuration of each of these components.

Telegraf

As distributed from InfluxData, telegraf already includes an agent that will collect Lustre server metrics. However, that agent does not collect jobstats metrics in a particularly friendly format, so I've made some local changes which are necessary if you want to be able to correlate Lustre activity to particular users or HPC jobs.

The changes to telegraf are located here: https://gitlab.umd.edu/acigs-tools/telegraf. The only changes made are to the telegraf input plugin.

The configuration file for telegraf on each Lustre server appears as follows (comments stripped out for brevity). The relevant sections of interest are *outputs* and *lustre2.*

/etc/telegraf/telegraf.conf

```
[global_tags]
[agent]
interval = "10s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "10s"
flush_jitter = "0s"
precision = ""
debug = false
quiet = false
logfile = ""
hostname = ""
omit_hostname = false
[[outputs.graphite]]
servers = ["stats-1.deepthought2.umd.edu:2003"]
template = "host.measurement.tags.field"
timeout = 10
[[inputs.cpu]]
percpu = true
totalcpu = true
collect_cpu_time = false
report_active = false
[[inputs.disk]]
ignore_fs = ["tmpfs", "devtmpfs", "devfs", "overlay", "aufs", "squashfs"]
[[inputs.diskio]]
[[inputs.kernel]]
[[inputs.mem]]
[[inputs.processes]]
[[inputs.swap]]
[[inputs.system]]
[[inputs.exec]]
commands = [
"/etc/telegraf/scripts/collect_ib"
]
timeout = "1s"
name_suffix = "_collect_ib"
data_format = "json"
tag_keys = [
"device",
"port"
]
[[inputs.ipmi_sensor]]
[[inputs.lustre2]]
interval = "30s"
[[inputs.net]]
```

Graphite

Graphite receives and stores the metrics collected by telegraf. It is highly recommended that Graphite (and Grafana) run on a separate server, and that the filesystem where metrics are stored reside on a flash drive. Depending on how many Lustre servers you have, Lustre generates a LOT of metrics and the backend server must be capable of storing them all as they arrive. We are currently running the metric collection on a node with two E5-2670 processors (16 cores total), with the metric storage on an Intel 1TB SSD. Collecting metrics for 12 OSSes and 1 MDS, and a handful of non-Lustre hosts is using about 320GB of storage. The SSD is mounted as */opt/graphite/storage*.

Graphite can be configured to run a few different processes, *carbon-cache* is the one we're using, and it's responsible for collecting the metrics and writing them to disk. Graphite also comes with *carbon-relay* and *carbon-aggregator.* The relay receives metrics and can redistribute them to multiple aggregators or caches. The aggregator can combine multiple input metrics to generate summary metrics. Those two tools as provided with Graphite are both written in python, and in practice I've found those too slow to keep up with the volume of metrics necessary. Instead, we're using a third tool called *carbon-c-relay* which is an integrated relay and aggregation engine, that's written in C for much improved performance.

On the collector box, we're running four carbon-cache processes, each listening on its own port. Carbon-c-relay receives metrics from the clients, does some aggregation, and hands those metrics off to the *carbon-cache* processes which then write the metrics to disk.

Lustre generates per-user (or per-job) statistics for every single OST. In order to generate graphs more efficiently, I've added aggregation rules to generate summary metrics across the OSTs. In addition, I've added an aggregation rule to generate a summary metric for all MDS operations.

This is the configuration file for the carbon caches- note that only the carbon-cache service needs to be enabled, as carbon-relay and carbon-aggregator are not used.

/opt/graphite/conf/carbon.conf

```
[cache]
DATABASE = whisper
ENABLE_LOGROTATION = True
USER =
MAX_CACHE_SIZE = inf
MAX_UPDATES_PER_SECOND = 100000
MAX_UPDATES_PER_SECOND_ON_SHUTDOWN = 100000
MAX_CREATES_PER_MINUTE = 50000
MIN_TIMESTAMP_RESOLUTION = 10
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2100
ENABLE_UDP_LISTENER = False
UDP_RECEIVER_INTERFACE = 0.0.0.0
UDP_RECEIVER_PORT = 2023
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2101
USE_INSECURE_UNPICKLER = False
CACHE_QUERY_INTERFACE = 127.0.0.1
CACHE_QUERY_PORT = 2102
USE_FLOW_CONTROL = True
LOG_UPDATES = False
LOG_CREATES = False
LOG_CACHE_HITS = False
LOG_CACHE_QUEUE_SORTS = False
CACHE_WRITE_STRATEGY = sorted
WHISPER_AUTOFLUSH = False
WHISPER_FALLOCATE_CREATE = True
[cache:b]
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2103
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2104
CACHE_QUERY_INTERFACE = 127.0.0.1
CACHE_QUERY_PORT = 2105
[cache:c]
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2106
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2107
CACHE_QUERY_INTERFACE = 127.0.0.1
CACHE_QUERY_PORT = 2108
[cache:d]
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2109
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2110
CACHE_QUERY_INTERFACE = 127.0.0.1
CACHE_QUERY_PORT = 2111
[relay]
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2013
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2014
RELAY_METHOD = rules
REPLICATION_FACTOR = 1
DESTINATIONS = 127.0.0.1:2004
MAX_QUEUE_SIZE = 3000000
MAX_DATAPOINTS_PER_MESSAGE = 5000
QUEUE_LOW_WATERMARK_PCT = 0.8
TIME_TO_DEFER_SENDING = 0.0001
USE_FLOW_CONTROL = True
USE_RATIO_RESET=False
MIN_RESET_STAT_FLOW=1000
MIN_RESET_RATIO=0.9
MIN_RESET_INTERVAL=121
```

```
[aggregator]
LINE_RECEIVER_INTERFACE = 127.0.0.1
LINE_RECEIVER_PORT = 2003
PICKLE_RECEIVER_INTERFACE = 127.0.0.1
PICKLE_RECEIVER_PORT = 2004
FORWARD_ALL = True
REPLICATION_FACTOR = 1
MAX_QUEUE_SIZE = 3000000
USE_FLOW_CONTROL = True
MAX_DATAPOINTS_PER_MESSAGE = 500
MAX_AGGREGATION_INTERVALS = 5
LOG_AGGREGATOR_MISSES = False
```

This determines the size of the individual whisper databases and how they store and aggregate the metrics over time:

/opt/graphite/conf/storage-schemas.conf

```
[carbon]
pattern = ^carbon\.
retentions = 60:90d
[lustre_space]
pattern = ^lustre\.space\.
retentions = 60s:7d,300s:90d
[lustre_files]
pattern = ^lustre\.files\.
retentions = 60s:7d,300s:90d
[jobstats_agg]
pattern = ^lustre\.usage\.
retentions = 30s:2d,300s:2w
[jobstats_raw]
pattern = .*lustre2.*
retentions = 30s:2d,300s:2w
[default_10sec_for_1day]
pattern = .*
retentions = 10s:1d,60s:1w,300s:90d
```

And these are the configuration files for carbon-c-relay:

/etc/carbon-c-relay.conf

```
cluster graphite
  fnv1a_ch
    127.0.0.1:2100
    127.0.0.1:2103
    127.0.0.1:2106
    127.0.0.1:2109
  ;
aggregate
        ^(oss|mds)[^.]+\.lustre2\.([^.]+)\.[^.]+\.(jobstats_[^.]+)$
  every 30 seconds
  expire after 45 seconds
  compute sum write to
    lustre.usage.\2.\3
  send to graphite
  ;
aggregate
        ^mds[^.]+\.lustre2\.([^.]+)\.[^.]+\.jobstats_[^.]+$
  every 30 seconds
  expire after 45 seconds
  compute sum write to
    lustre.usage.\1.mds_aggr
  send to graphite
  ;
match *
  send to graphite
  stop
  ;
```

This increases the size of the metric batch sizes and receive queues

/etc/sysconfig/carbon-c-relay

```
ARGS="-p 2003 -w 4 -b 25000 -q 250000"
```

Grafana

Grafana is responsible for actually graphing the metrics. It is a wsgi process that sits behind a web server (Apache). It needs to have access to the metric storage and therefore must run on the same host as Graphite. By default Grafana has its own authentication mechanisms, but it's somewhat limited. I've chosen to disable the built-in authentication and instead use the much richer Apache authentication mechanism so that we can use our Kerberos authentication. In our configuration, the server that hosts Grafana and Graphite is on a private network. The apache server runs on a publicly accessible server, and acts as a reverse proxy to connect user requests to Grafana.

Here are relevant configuration files:

/etc/grafana/grafana.ini

```
[paths]
[server]
domain = deepthought2.umd.edu
root_url = https://deepthought2.umd.edu/graphs
[database]
[remote_cache]
[dataproxy]
[analytics]
[security]
[snapshots]
[dashboards]
[users]
allow_sign_up = false
[auth]
disable_login_form = true
disable_signout_menu = true
[auth.anonymous]
enabled = true
org_name = University of Maryland
[auth.github]
[auth.google]
[auth.generic_oauth]
[auth.grafana_com]
[auth.proxy]
enabled = true
[auth.basic]
enabled = false
[auth.ldap]
[smtp]
[emails]
[log]
[log.console]
[log.file]
[log.syslog]
[alerting]
[explore]
[metrics]
[metrics.graphite]
[tracing.jaeger]
[grafana_com]
[external_image_storage]
[external_image_storage.s3]
[external_image_storage.webdav]
[external_image_storage.gcs]
[external_image_storage.azure_blob]
[external_image_storage.local]
[rendering]
[enterprise]
[panels]
[plugins]
```

/etc/httpd/conf.d/graphite.conf (on the Grafana/graphite server)

```
LoadModule wsgi_module modules/mod_wsgi.so

WSGISocketPrefix /var/run/wsgi

Listen 8013
<VirtualHost *:8013>

    WSGIDaemonProcess graphite-api processes=5 threads=5 display-name='%{GROUP}' inactivity-timeout=120 python-
home=/opt/graphite
    WSGIProcessGroup graphite-api
    WSGIApplicationGroup %{GLOBAL}
    WSGIImportScript /var/www/wsgi-scripts/graphite-api.wsgi process-group=graphite-api application-group=%
{GLOBAL}

    WSGIScriptAlias / /var/www/wsgi-scripts/graphite-api.wsgi

    <Directory /var/www/wsgi-scripts/>
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

And the httpd proxy on the host deepthought2.umd.edu:
/etc/httpd/conf.d/graphite-proxy.conf

```
<Proxy *>
    SSLRequireSSL
    AuthType Kerberos
    KrbMethodNegotiate on
    KrbAuthoritative on
    Krb5Keytab /etc/krb5.keytab.httpd
    KrbSaveCredentials off
    KrbLocalUserMapping On
    Require valid-user

    RewriteEngine On
    RewriteRule .* - [E=PROXY_USER:%{LA-U:REMOTE_USER},NS]
    RequestHeader set X-WEBAUTH-USER "%{PROXY_USER}e"
</Proxy>

RequestHeader unset Authorization

ProxyRequests Off
ProxyPass /graphs http://stats-1.deepthought2.umd.edu:3000
ProxyPassReverse /graphs http://stats-1.deepthought2.umd.edu:3000
```

If you are using selinux on the proxy server you'll need to allow httpd to make outbound connections:

```
/usr/sbin/setsebool -P httpd_can_network_connect 1
```

Selinux

The server running Graphite/Grafana works fine with selinux in enforcing mode, with no changes necessary.

The proxy server works fine with selinux in enforcing mode as well, with the one change mentioned above.

Firewall Rules

On the Graphite/Grafana server you need to have port 2003/tcp open to your OSSes, MDSes, and any other machines you want to receive metrics from. You also need to have port 3000 open to your proxy server.

On the proxy server you need to have https (443/tcp) open to whatever machines you have users sitting in front of with their web browsers.

Hosts sending metrics do not need any inbound ports open, as they are pushing metrics.

Where to Find Stuff

- Telegraf - https://docs.influxdata.com/telegraf
- Graphite - https://graphite.readthedocs.io/en/latest/index.html
- carbon-c-relay - https://github.com/grobian/carbon-c-relay
- Grafana - https://grafana.com/