CollectD

Abhimanyu Kumar
akumar24@ncsu.edu

August 4, 2017

Contents

1 Introduction

CollectD is a system statistics collection daemon which periodically reads data from a machine (virtual or bare-metal) and provides the statistics over the network. This information can be used for monitoring devices in the network, such as conforming to SLAs or finding bottleneck.

CollectD achieves this functionality with the help of Plugins.

2 Plugins

CollectD in itself is nothing. It relies on the functionality provided by the Plugins to get the statistics. Plugins can be broadly classified into two types:

1. **Read Plugins**: These types of Plugins reads statistics, such as CPU, Memory, interface stats, etc.

2. **Write Plugins**: These Plugins write the data received from the Read Plugins into a file. How the data is stored, i.e. data structure, depends on the Write Plugin enabled. Currently CollectD supports two types of data formats: Round Robin Database (RRD), Comma Separated Values (CSV).

There are many Plugins which don't fall into either of the two mentioned types. One such Plugin is Network. It is because of the network plugin that one can view system statistics of every machine in the network.

The network plugins follows client-server model. There will be one (or more) server continuously listening to clients. The CollectD daemon on the clients will be reading the data from the client system and will send the data to the server over the network using the Network Plugin.

2.1 Enabling Plugins

Plugins are enabled on per-client basis, i.e., every client machine can have different set of plugins enabled depending on the need. But in order to communicate the statistics over the network, every client must have Network Plugin enabled and pointing to the server.

The most significant option is **LoadPlugin**, which controls which plugin to enable. Wherever necessary, every Plugins will have a configuration block using which, user can change the default behaviour of the Plugin. The configuration block a plugin will look like : <**Plugin name**>, where name is the name of the plugin.

3 Configuration

We start by installing CollectD:

```
$ sudo apt-get install collectd collectd-utils
```

Note: CollectD has to be installed on every machine whose system statistics the user wants to collect, i.e. both on the server and the clients.

CollectD documentation with README containing information of each plugin, along with some examples to get started can be found by:

```
$ cd /usr/share/doc/collectd/
```

Go to the configuration file of CollectD to enable the plugins:

```
$ sudo vi /etc/collectd/collectd.conf
```

The file will have most of the lines commented. This means that most of the plugins are disabled. There will be a **#LoadPlugin name** option, uncommenting which will enable the plugin "name". This option is there for every plugin.

After the **LoadPlugin** option, there are the configuration blocks of most of the Plugins. If the user want to change the default settings, uncomment the configuration block and setting the user wish to change.

It is a good practice to always enable the Plugins: logfile and syslog, along with their configuration blocks. This ensures that all the messages are logged. The location of the log file is mentioned in the logfile's configuration block.

Note: Configurations are executed by CollectD from top to bottom. This means that the plugins are loaded in the order they appear in the configuration file. Therefore it's a good practice to place the aforementioned log plugins at the top of the file.

3.1 Configure in Server Mode

In the configuration file, search for the **LoadPlugin** option and configuration block for Network Plugin, and enable them by uncommenting.

Inside the configuration block of Network Plugin, there will be two options and configuration blocks: Server; Listen. The option enable the mode and configuration block provides hooks the user can change. For now lets just stick to the option. Uncomment the Listen option to enable the server mode and make the CollectD daemon listen to client. By default it looks like this:

```
Listen "ff18::efc0:4a42" "25826"
```

Change *ff18::efc0:4a42* to the IP address of the client. *25826* is the UDP port number on which data will be sent or received. User can change this as per the requirement. If this field is left empty, *25826* will be considered.

If there are multiple clients, add another Listen option with the IP address and the port of the client. Or if security is not of concern, replace *ff18::efc0:4a42* with *0.0.0.0*, so that any client can send data to the server.

Next, we will enable CSV and/or RRDtool plugins (by uncommenting their **LoadPlugin**) so as to store server data in a file, at the server only.
Last step is to restart CollectD daemon for the changes to take effect:

```
$ service collectd restart
```

To access the data:

```
$ cd /var/lib/collectd/
```

3.2 Configure in Client Mode

In the configuration file, search for the **LoadPlugin** option and configuration block for Network Plugin, and enable them by uncommenting.

In the same configuration file, at the client, uncomment the Server option to enable the client mode and make the CollectD daemon send data to the server. By default it looks like this:

```
Server "ff18::efc0:4a42" "25826"
```

Change *ff18::efc0:4a42* to the IP address of the servers. Just like in server, *25826* is the UDP port number on which the data will be received/sent. User can change this as per the requirement. If this field is left empty, *25826* will be considered.

If there are multiple servers, add another Server option with the IP address and the port of the server.

Next, we will enable CSV and/or RRDtool plugins (by uncommenting their **LoadPlugin**) so as to store client data in a file.
Last step is to restart CollectD daemon for the changes to take effect:

```
$ service collectd restart
```

The client data will be written in a file, both at the server and the client. To access the data:

```
$ cd /var/lib/collectd/
```

3.3 Custom Plugins

CollectD provides interfaces to CollectD's plugin system. These interfaces are in the form of Plugins themselves, some of them are:

1. **Plugin Python**: Embeds Python-interpreter into CollectD
2. **Plugin Java**: Embeds Java Virtual Machine (JVM) into CollectD
3. **Plugin Perl**: Embeds Perl-interpreter into CollectD

These make it possible to write plugins for CollectD. The Python and Perl Plugins are a lot more efficient in executing rather than executing their scripts. Also, the plugins provide a lot more functionality than the scripts.

4 Inferences

This section includes the observations made once the CollectD server-client model has been setup. These observations may be limited in nature, i.e., many more observations might be discovered in the future. Upon their discovery, they will be included in the document.

4.1 Data format of the stored data

We had mentioned that CollectD can store data in two formats, CSV and RRD. But the underlying format of both types are the same, which is:

```
epoch    data
123313   some_data
```

Epoch is the time (in seconds) elapsed since 1970-01-01 00:00:00 UTC, data is the data written by some Plugin.

4.2 CollectD runs at Application Layer

Upon taking the Wireshark captures at input interface of the server, it was found that CollectD runs at the Application Layer of the Networking stack. Currently the industry is shifting towards HTTP (v3.0) as an Application Layer protocol because of its simplicity, but since CollectD came almost a decade ago, there was no de-facto standard at Application Layer at that time.