RRD Data Resampling

Use case

The customer wants to monitor temperature from a device in NMIS, the device returns temperatures as integer numbers using SNMP. The problem the customer faces is that even when the data is only integers, NMIS still display and graphs data with decimal values.

RRDTool Data Resampling

The effect that occurs here is due to data resampling, this is an important feature of RRDtool, and it is used since it is practically impossible to collect data and enter it in RRDtool at exact intervals. Therefore, RRDtool interpolates the data, so it is stored at exact intervals.

Suppose a counter increases by exactly one for every second. You want to measure it in 300 seconds intervals. You should retrieve values that are exactly 300 apart. However, due to various circumstances you are a few seconds late and the interval is 303. The delta will also be 303 in that case. Obviously, RRDtool should not put 303 in the database and make you believe that the counter increased by 303 in 300 seconds. This is where RRDtool interpolates: it alters the 303 value as if it would have been stored earlier and it will be 300 in 300 seconds. Next time you are at exactly the right time. This means that the current interval is 297 seconds and also the counter increased by 297. Again, RRDtool interpolates and stores 300 as it should be.

```
      in the RRD                 in reality

 time+000:   0 delta="U"   time+000:    0 delta="U"
 time+300: 300 delta=300   time+300:  300 delta=300
 time+600: 600 delta=300   time+603:  603 delta=303
 time+900: 900 delta=300   time+900:  900 delta=297
```

To carry out this document, real data was obtained from a server, here it is detailed how the data was obtained.

1) Polling.log

We use the "polling.log" to be able to review the exact values that NMIS is sending to the RRDTool for storage in the database (RRD).

It should be taken into consideration that this log can consume a large amount of space quickly due to the amount of data obtained during the collection or polling process. This option is not active by default, so in order to activate the log you must change the item: "polling_log" in the configuration file "Config. nmis", specifying the path where the log will be saved, it is recommended to use: '<nmis_logs> /polling.log' looking like this:

```
'polling_log' => '<nmis_logs>/polling.log'
```

NMIS will not create the polling.log file by default, even if the option is activated, the file must be created manually for NMIS to start writing over it. It is recommended to just “touch” the file:

```
$ touch /usr/local/nmis8/logs/polling.log
```

2) nmis.log

Default NMIS log, located at `/usr/local/nmis8/logs/`

3) RRD Dump

RRD are usually stored at `/usr/local/nmis8/database/nodes/`

The RRD data has to be exported using the following command:

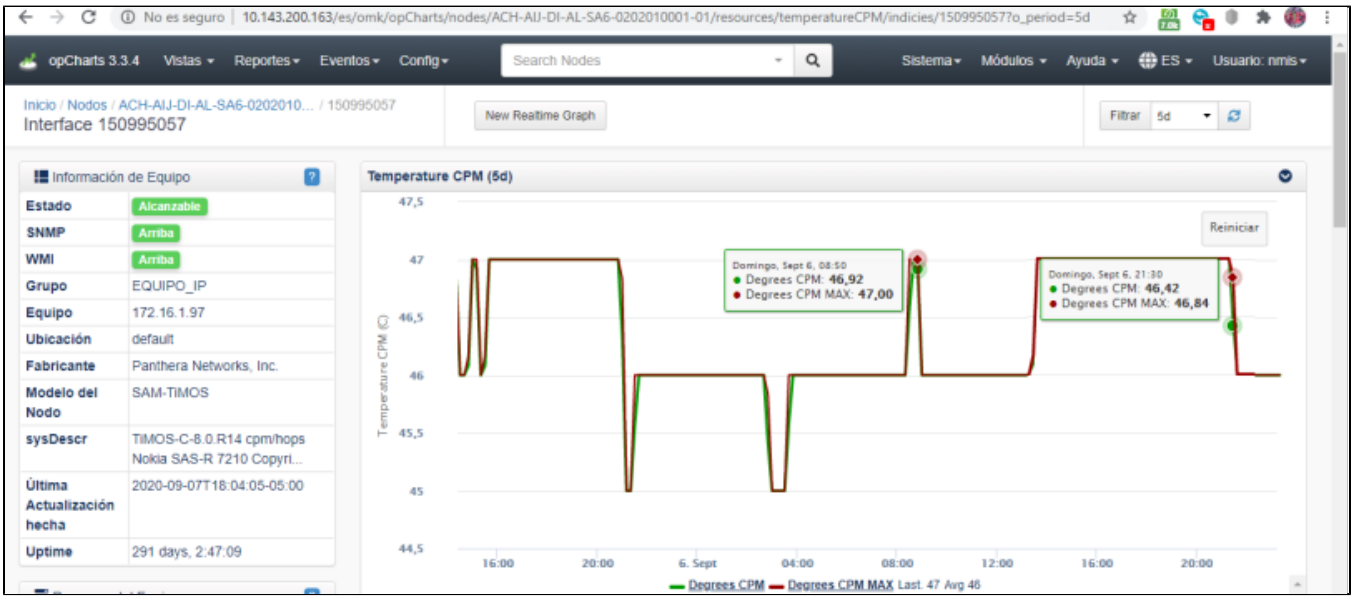
```
$ rrdtool dump temperatureCPM-150995057.rrd  /tmp/temperature.xml
```

4) Graph

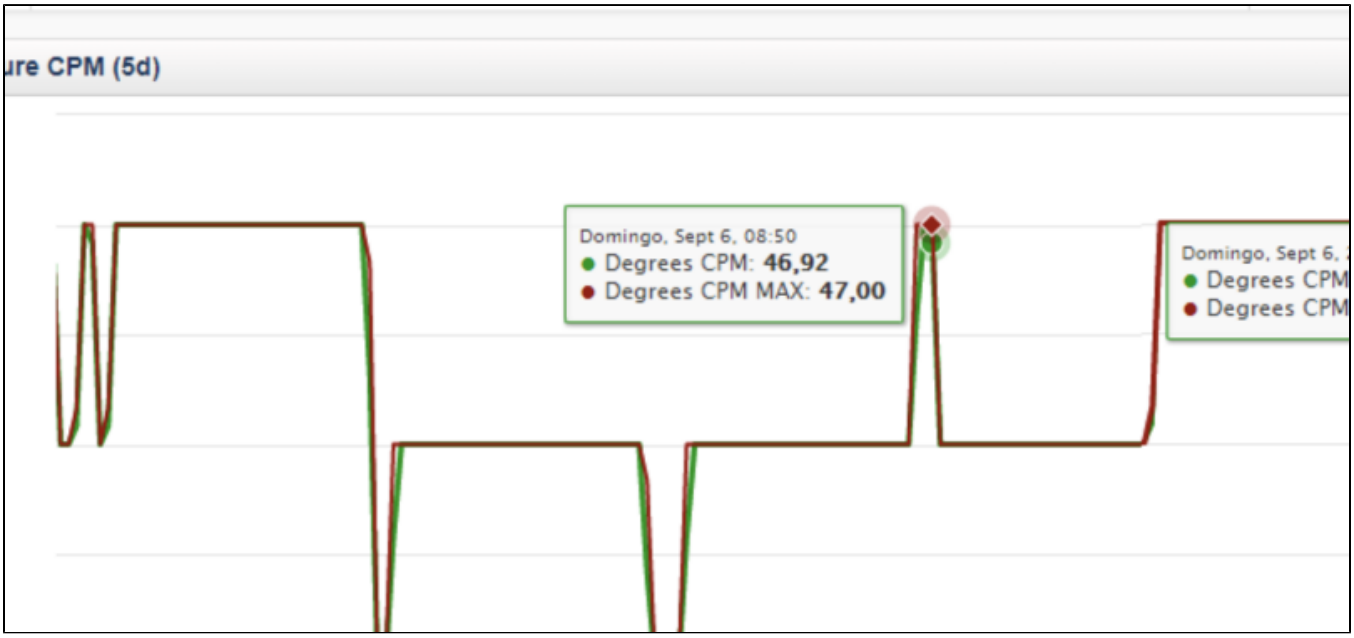
We need to obtain the graph of the device on NMIS or opCharts.

Demonstration

For this demonstration, we will use the custom graph called: "temperatureCPM" that is part of the model : "Model-SAM-TiMOS.nmis".



Let's focus on the value obtained on Sep. 06 2020 at 8:50am, the value displayed is 46.62. We have to consider that this graph shows a period of 5 days and data summarisation may have been already applied and the value may be slightly different, which doesn't affect the final result.



First we must export the rrd data to an XML file to be able to review its content.

Once exported, We will review the frequency of updating the RRD.

```
temperatureCPM-150995057.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE rrd SYSTEM "http://oss.oetiker.ch/rrdtool/rrdtool.dtd">
3  <!-- Round Robin Database Dump -->
4  <rrd>
5      <version>0003</version>
6      <step>300</step> <!-- Seconds -->
7      <lastupdate>1599519851</lastupdate> <!-- 2020-09-07 18:04:11 -05 -->
8
9      <ds>
10         <name> temperatureCPM </name>
11         <type> GAUGE </type>
12         <minimal_heartbeat>900</minimal_heartbeat>
13         <min>NaN</min>
```

In this particular case, this RRD is updated on:

2020-09-07 **18:04:11** -05

Now let's review the last data stored in the RRD.

```
2336            <!-- 2020-09-07 17:30:00 -05 / 1599517800 --> <row><v>4.7000000000e+01</v></row>
2337            <!-- 2020-09-07 17:35:00 -05 / 1599518100 --> <row><v>4.7000000000e+01</v></row>
2338            <!-- 2020-09-07 17:40:00 -05 / 1599518400 --> <row><v>4.7000000000e+01</v></row>
2339            <!-- 2020-09-07 17:45:00 -05 / 1599518700 --> <row><v>4.7000000000e+01</v></row>
2340            <!-- 2020-09-07 17:50:00 -05 / 1599519000 --> <row><v>4.7000000000e+01</v></row>
2341            <!-- 2020-09-07 17:55:00 -05 / 1599519300 --> <row><v>4.7000000000e+01</v></row>
2342            <!-- 2020-09-07 18:00:00 -05 / 1599519600 --> <row><v>4.7000000000e+01</v></row>
2343        </database>
2344    </rra>
```

Here we can see that the lastest data was:

```
<!-- 2020-09-07 18:00:00 -05 / 1599519600 --> <row><v>4.7000000000e+01</v></row>
```

We can also observe that the data was stored exactly every 5 minutes.

With this we notice that the RRD was updated at 18:04:11 but the latest stored data as for the period was on 18:00:00. This is because the RRD always writes the data from the previous period. With this in mind, we are going to review the period that we are interested in demonstrating.

Here we can see that on 06/09/2020 08:50:00 the value: 4.6835242743e+01 was saved into the RRD.

```
1940            <!-- 2020-09-06 08:30:00 -05 / 1599399000 --> <row><v>4.7000000000e+01</v></row>
1941            <!-- 2020-09-06 08:35:00 -05 / 1599399300 --> <row><v>4.7000000000e+01</v></row>
1942            <!-- 2020-09-06 08:40:00 -05 / 1599399600 --> <row><v>4.7000000000e+01</v></row>
1943            <!-- 2020-09-06 08:45:00 -05 / 1599399900 --> <row><v>4.7000000000e+01</v></row>
1944            <!-- 2020-09-06 08:50:00 -05 / 1599400200 --> <row><v>4.6835242743e+01</v></row>
1945            <!-- 2020-09-06 08:55:00 -05 / 1599400500 --> <row><v>4.6000000000e+01</v></row>
1946            <!-- 2020-09-06 09:00:00 -05 / 1599400800 --> <row><v>4.6000000000e+01</v></row>
1947            <!-- 2020-09-06 09:05:00 -05 / 1599401100 --> <row><v>4.6000000000e+01</v></row>
1948            <!-- 2020-09-06 09:10:00 -05 / 1599401400 --> <row><v>4.6000000000e+01</v></row>
```

Let's check the polling.log to see the exact time that NMIS sent the value to the RRD and what that value was:

```
353067:302603731:06-Sep-2020 07:59:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
353522:303006035:06-Sep-2020 08:04:11,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
353977:303368949:06-Sep-2020 08:09:08,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
354432:303801217:06-Sep-2020 08:14:09,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
354887:304198293:06-Sep-2020 08:19:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
355342:304573734:06-Sep-2020 08:24:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
355797:304978074:06-Sep-2020 08:29:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:47
356252:305387660:06-Sep-2020 08:34:11,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:47
356707:305777630:06-Sep-2020 08:39:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:47
357162:306183287:06-Sep-2020 08:44:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:47
357617:306574099:06-Sep-2020 08:49:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:47
358072:306952799:06-Sep-2020 08:54:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
358527:307395607:06-Sep-2020 08:59:08,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
358982:307779408:06-Sep-2020 09:04:12,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
359437:308143608:06-Sep-2020 09:09:08,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
359892:308521623:06-Sep-2020 09:14:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
360347:308948793:06-Sep-2020 09:19:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
360802:309282849:06-Sep-2020 09:24:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
361257:309689974:06-Sep-2020 09:29:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
361712:310093869:06-Sep-2020 09:34:11,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
362167:310505619:06-Sep-2020 09:39:10,temperatureCPM,ACH-AIJ-DI-AL-SA6-0202010001-01,150995057,,temperatureCPM,N:46
```

```
06-Sep-2020 08:49:10  ...   temperatureCPM,N:47
06-Sep-2020 08:54:10  ...   temperatureCPM,N:46
```

With this we can say that NMIS sends the data to the RRD at 08:49:10 and the RRD processes the data and writes the data corresponding to the previous period.

NMIS sent data to RRD	Received Value	Period written	Value Written
08:44:10	47	08:40:00	4.7000000000e+01
08:49:10	47	08:45:00	4.7000000000e+01
08:54:10	**46**	08:50:00	**4.6835242743e+01**
08:59:08	46	08:55:00	4.6000000000e+01

We can see that the received value was 46 but 46,835 was written to RRD, this is where the "data resampling" was applied, actually it is been applied to all the values received by the RRD but it is only noticed when the value change occurs, when the value is constant, this effect is not appreciated.

Let's see how resampling is calculated by the RRD tool.

First we must obtain the delta or difference between the value to be examined and its previous value.

NMIS sent data to RRD	Received Value	Period written	Value Written
08:49:10	47	08:45:00	4.7000000000e+01
08:54:10	**46**	08:50:00	**4.6835242743e+01**

The difference between 46 and 47 is 1. We have to divide this value by the interval in seconds of every period. Here the period was 5 minutes or 300 second:

1/300 = 0.00333333333 (rate)

Now we obtain the difference between the time in which the RRD received the data and the period in which it writes that data.

Received: 08:54:10 and Period Written: 08:50:00. The difference between them is 250 seconds.

Now we have to multiply the previously obtained seconds (delta) (250) by the rate (0.00333333333333), the result of this operation is: **0.83333333 (resampling value)**

In this case, the value is descending from 47 to 46, for these reason the result must be added to the current received value (46), the final result is: **46.83333 333**

This result is slightly different to the one displayed: , as we are not considering the millisecond when the data was stored on the RRD.

On the other hand, if the value increments, the same operation must be done, however the resampling value has to be taken from the current received value.

For example:

NMIS sent data to RRD	Received Value	Period written	Value Written
08:24:10	46	08:20:00	4.600000000e+01
08:29:10	**47**	08:25:00	**4.6166107933e+01**

The delta is 1 (47 and 46), interval in seconds between data: 300 Seconds.

1/300 = 0.00333333333 (rate)

The difference between the data received and the period written: again, 250 seconds.

The resampling value is: **0.83333333333**

The current received value is: 47, because the value is increasing from 46 to 47, we have to take the reampling value (**0.83333333333**) from current received value.

47-0.83333333333 = **46.1666666667**

```
<!-- 2020-09-06 08:15:00 -05 / 1599398100 --> <row><v>4.6000000000e+01</v></row>
<!-- 2020-09-06 08:20:00 -05 / 1599398400 --> <row><v>4.6000000000e+01</v></row>
<!-- 2020-09-06 08:25:00 -05 / 1599398700 --> <row><v>4.6166107933e+01</v></row>
<!-- 2020-09-06 08:30:00 -05 / 1599399000 --> <row><v>4.7000000000e+01</v></row>
<!-- 2020-09-06 08:35:00 -05 / 1599399300 --> <row><v>4.7000000000e+01</v></row>
```