NAME

rrd_pdpcalc – PDP inner calculation logics with an example by Tianpeng Xia

DESCRIPTION

This article explains how PDP are calculated in a detailed yet easy-to-understand way, with an example.

Refreshing some basics about PDP

Fundamental knowledge

If you have not read the tutorials or man pages either on the official site or those by others, then I strongly encourage you to do so. As said in the description, this article will only explain how a PDP is calculated, but not the definition of it. So please read the following materials to get a basic understanding of PDP:

<http://rrdtool.vandenbogaerdt.nl/process.php> – By Alex van den Bogaerdt. This article explained PDP in a very detailed and clear way, however, it does not explain the ''normalization process'' in its ''Normalize interval'' section in the right way(as opposed to the official version I confirmed with `@oetiker` himself). The flaw can be easily seen in the bar charts, discussed in the ''Calculation logics'' section.

<https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html> – This one is on the official site. Actually it's the manual page for ''rrdcreate'', and it reveals what's under the hood with regard to PDP calculation in its ''The HEARTBEAT and the STEP'' section.

The text graph by Don Baarda provides a vivid explanation on how **UNKNOWN** data are produced and how heartbeat value can influence in the sampling. Unfortunately, it fails to give a clear method by which PDPs are calculated.

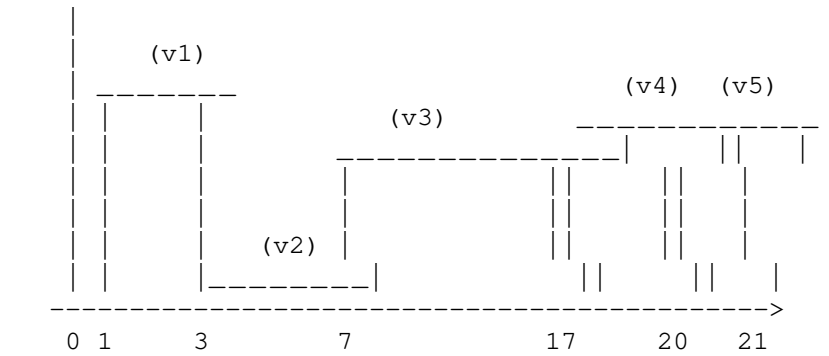
<https://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html> – Another detailed official tutorial by Alex van den Bogaerdt. Similarly, it only provides examples with data evenly and exactly distributed according to the step set.

If you don't like doing experiments or care about the inner mechanics that much, you can just stop here and give more attention to more practical topics like graph exports or command manual. But if you are the sort of people like me who just care as much about the calculation logics, please read on.

Calculation logics

Here begins the core part of this article. In the following content of this section, I would like to give two versions of calculation methods, one by Alex van den Bogaerdt and the other by `@eotiker`.

To provide an ASCII-friendly explanation, I will explain both versions with the char below instead of a real image.



The X axis means time slots(each second denotes one slot) and the Y axis means the value.

Let's make everything a little clearer:

– The step is 5

– each PDP gets updated only if a value arrives at or after the last slot of the PDP, for instance, the last slot of the PDP from 16 to 20 is 20

– The heartbeat is 20, so the samples during the entire 7–17 period is not discarded

– At second 3, the first value comes in as v1, and so on

– Second 0 is the origin, and it does not count as a sample

Bogaerdt version

As can be seen on this page: <http://rrdtool.vandenbogaerdt.nl/process.php>, after all the primary data are transformed to rates(except for GAUGE, of course), they have to go through a **normalization process** if they are not distributed exactly according to the step or on well-defined boundaries in time, in the words of the author.

What does that mean? Basically, if all the **known** (as opposed to an **unknown** value) data make up at least 50% of all slots during a period, then a PDP is calculated from them.

This version seems to go well until we reach the bar chart part.

According to the ASCII bar chart, we have the following results:

From second 1 on, the PDP of each period(1−5,6−10, ...) is computed by averaging all the values within it.

So: – the PDP from 1 to 5 is (v1*3+v2*2)/5

– the PDP from 6 to 10 is (v2*2+v3*3)/5

– the PDP from 11 to 15 is (v3*5)/5, since all the values in slots 11, 12, 13, 14 and 15 are the same, which is v3

– ...

The official version(also `@oetiker` version):

Using the same chart, this version suggests the following:

– the PDP from 1 to 5 is (v1*3+v2*2)/5

– the PDPs from 6 to 10 and 11 to 15 are the **SAME**, which is (v2*2+v3*8)

– ...

A Comparison and some explanation

So we have seen the above two versions and their PDPs from 6 to 10 and 11 to 15 do not comply with each other.

Why is that?

Because the difference between the official version and Bogaerdt version stems from the way they do the calculation for PDP(6−10) and PDP(11−15).

Let's discuss this in more detail using the above bar chart.

Bogaerdt's version,

PDPs are **always computed individually** no matter how values arrive.

For example, the value at slot 17 comes after the last slot of PDP(11−15). Also, the immediate previous value before slot 17 is at 7. All the slots from 7 to 17 are assigned v3. Since each PDP is computed individually, PDP(6−10) is (v2*2+v3*3)/5 while the PDP(11−15) is (v3*5)/5.

The official version

PDPs are **always computed in terms of the steps which the next update spans**, be it 1 step, 2 steps or n steps; in other words, PDPs may be computed **together**.

For example, the update at slot 17 spans PDP(6−10) and PDP(11−15) because the **immediate** previous value is at 7 and 7 is within 6 and 10 , and 17 is after 15. PDP(1−5) and PDP(16−20) are not included since the update at slot 7 has already triggered the calculation for PDP(1−5) and the update at slot 17 comes before the last slot of PDP(16−20) which is 20.

That's the reason why PDP(6−10) and PDP(11−15) have the same value, (v2*2+v3*8).

An example

If you are still confused, don't worry, an example is here to help you.

Let's get our hands dirty with some commands

```
rrdtool create target.rrd --start 1000000000  --step 5 DS:mem:GAUGE:20:0:100 RRA
rrdtool update target.rrd 1000000003:8 1000000006:1 1000000017:6 \
1000000020:7 1000000021:7 1000000022:4 \
1000000023:3 1000000036:1 1000000037:2 \
1000000038:3 1000000039:3 1000000042:5
rrdtool fetch target.rrd AVERAGE --start 1000000000 --end 1000000045
```

Basically, the above codes contain 3 commands: create, update and fetch. First create a new rrd file, and then we feed in some data and last we fetch all the PDPs from the rrd.

Focus on single steps

In order to provide a detailed explanation, each the calculation process of each PDP is provided.

Below is the output of the commands above:

```
1000000005: 5.2000000000e+00
1000000010: 5.5000000000e+00
1000000015: 5.5000000000e+00
1000000020: 6.6000000000e+00
1000000025: 1.7333333333e+00
1000000030: 1.7333333333e+00
1000000035: 1.7333333333e+00
1000000040: 2.8000000000e+00
1000000045: nan
1000000050: nan
```

NOTE: 1000000005 means the PDP from 1000000001 to 1000000005, and so on. For concision and readability, we use only the last two digits, so 05 denotes 1000000005. We choose the type of the data source as gauge because original values will be treated as rates, no additional transformation is needed, see this article <http://rrdtool.vandenbogaerdt.nl/process.php> for detail.

05: 5.2 = (8*3+1*2)/5

10: 5.5 = (1*1+6*9)/10

15: the same as the previous one

20: 6.6 = (6*2+7*3)/5

25: 1.73333 = (7+4+3+1*12)/15

...

45: nan, as the last value is at 42,which does not trigger the calculation for PDP(41−45)

50: nan, why this unknown PDP is shown is explained in this article <https://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html>

SUMMARY

All that said, I hope you get a clear understanding of the inner calculation ‘‘magic’’ for PDPs.

Other References

- A great PowerShell shell script for generating ASCII bar charts: <https://gallery.technet.microsoft.com/scriptcenter/Sample−Script−to−Generate−59c80d4c>
- <https://stackoverflow.com/questions/18924450/rrd−wrong−values>