opCharts 4 TopN Tables including new metrics

Version

This relates to opCharts version 4

Introduction

Each TopN table is really just a component in a dashboard, components are defined in json documents. opCharts ships the Default components in omk/lib /json/opCharts/components.d/. User defined components are loaded from omk/conf/components.d, and are loaded last so default components are overridden, if desired, by creating new versions in the User defined folder.

opCharts 4 takes several steps to generate TopN data and display it (these are different and simpler than with opCharts3). Understanding these steps will make the process of adding a new TopN table make more sense.

1. NMIS collects statistics during polling and stores the data as timeseries in RRD files.
2. NMIS9 also adds the latest values for certain metrics into the MongoDB nodes latest_data collection.
    a. Which metrics are stored and where they are stored is defined in Common-stats.nmis.
    b. Common-stats.nmis definitions controls what metrics are stored in "data" section of the Node from the DEF statements.
    c. Common-stats.nmis can also calculate new values from other data the results are found in the "dervied_data" section of the node's MongoDB database entry. This derived_data is the Results of the PRINT statements - so the line which is printed by the statement.
    a. The data and derived data from Common-stats.nmis is how we "normalise" polled data into metrics which are common to all device types. For example "cpuLoad" is normalised to a value between 0 and 100, this represents how much CPU has been used in each poll cycle. This might have come from one device as "5 minute Avg Busy percentage" and need no conversion but might have been calculated from "Idle seconds" on another using the formula cpuLoad is (($r / 300) * 100)
3. opCharts reads the latest values from MongoDB's relevant "data" and "derived_data" for each node based on what is defined in the TopN components JSON document.
4. opCharts finds the Top N entries for that normalised metric and displays them
5. When a user views the TopN page, opCharts loads all components tagged with topn, and puts them into a dashboard which is dynamically created and presented
    a. If a TopN component is added to a user defined dashboard the specific component is searched for and included in the dashboard

Understanding the TopN component document

Below is the definition for CPU Load TopN. Note this is looking for a Statistic named cpuLoad and it looks for it in the derived_data section of the node's MongoDB database entry. This means to have this TopN work for your node there must be a PRINT line in Common-stats.nmis which returns "cpuLoad" calculated / normalised from that nodes CPU statistics. See next section for more information on calculating.

/usr/local/omk/lib/json/opCharts/components.d/topn_cpu_load.json

```
// VERSION=2.46.0
{
  "name": "TopN CPU Load",
  "tags": [ "topn" ],
  "ep_template_file": "charts/topn/topn_table_component",
  "ep_configuration_file" : "charts/topn/topn_table_component_configuration",
  "options": {
    "titleText": "TopN CPU Load",
    "limit" : 10,
    "show_element": 0,
    "show_sparkline": 0,
    "show_value": 1
  },
  "order": 1,
  "parameters": {
    "topn_key": "cpuLoad",
    "data_section": "derived_data",
  },
  "type": "ep_template"
}
```

Have device Metrics appear in current TopN

If you have written a new device model or a current model does not show in one of the current TopN categories you may need to update Common-stats.nmis to include the metrics you collect as a Normalised metric. Also, if you wanted to create a new TopN Metric you would need to make sure this new metric type appears in Common-stats.nmis

First find the name of the normalised metric used by the TopN component, as described above. If this is a new TopN then select a new name for the new TopN metric type to be used in the component definition and in Common-stats.nmis.

The document we saw in the previous section uses the topn_key "cpuLoad". Some examples from Common-stats.nmis of how the topn_key "cpuLoad" is calculated for different Models with different CPU statistics are shown here:

Net-SNMP (hr resources MIB)

'hrsmpcpu' => [
'DEF:hrCpuLoad=$database:hrCpuLoad:MAX',
'PRINT:hrCpuLoad:AVERAGE:hrCpuLoad=%1.2lf',
'PRINT:hrCpuLoad:AVERAGE:**cpuLoad**=%1.2lf'
],

Some Types of Cisco Router

'cpuUtil' => [
'DEF:cpuUtil=$database:cpuUtil:AVERAGE',
'PRINT:cpuUtil:AVERAGE:cpuUtil=%1.2lf',
'PRINT:cpuUtil:AVERAGE:**cpuLoad**=%1.2lf',
],

or with a calculation before hand to convert some other metric style to a normalised value. This shows converting Idle CPU seconds into a percentage busy over 5 minutes and then using that provide cpuLoad.

'CDEF:avgBusy5= 300,idleSecs,-,300/,100,*',

'PRINT:avgBusy5:AVERAGE:cpuLoad=%1.2lf',

Note in the above that the topn_key is the last part of the PRINT statement so the label that is printed.

Default shipping TopN table "topn_key" names

CPU "cpuLoad",
Memory "memoryUtil"
Interface Output "outputUtil"
Interface Input "inputUtil"
Inteface Errors In "ifInErrors"
Interface Discards Out "ifOutDiscards"
TCP connections "tcpCurrEstab",

Add a new Table for a new type of metric

The easiest way to get started is to copy an existing topn component file from omk/lib/json/opCharts/components.d/topn_* into omk/conf/components.d (and rename the file).

Next open the new file and modify it to suite your new TopN data, the changes required are fairly straight forward and are outlined below:

```
{
  "name": "TopN NEW THING", # set a new name here
  "tags": [ "topn" ], # leave this
  "ep_template_file": "charts/topn/topn_table_component", #leave this
  "ep_configuration_file" : "charts/topn/topn_table_component_configuration", # leave this
  "options": {
    "titleText": "TopN NEW THING", # set the component title here
    "limit" : 10, # set how many to show here
    "show_element": 0,
    "show_sparkline": 1 , # 0 for no sparkline, 1 for sparkline
    "show_value": 1, # 0 to not show the value, 1 to show the value
  },
  "order": 1, # order the new TopN table will appear in the TopN list, see the others to get a total ordering
  "parameters": {
    "topn_key": "someNewThing", # The variable or Print statement return from Common-stats.nmis
    "data_section": "derived_data",  # derived_data if the key comes from PRINT or data if the key comes from
DEF or CDEF.
  },
  "type": "ep_template" # leave this
}
```

The new TopN table should now appear in opCharts TopN page and also as an option when adding a component to a dashboard under the opCharts data source.

Editing TopN Table

If you are having issues with overflow on your values you can edit /usr/local/omk/templates/charts/topn/topn_table.html.ep and remove class="wide-20" from the first portion of this file. ***Please note that upgrades will overwrite this!***

Before:

<table class="table table-condensed table-bordered topn-table">

 <col span="1" class="wide-20">

 <col span="1">

 <col span="1" class="wide-45">

 <col span="1">

 </col>

After:

<table class="table table-condensed table-bordered topn-table">

<col span="1">

<col span="1">

<col span="1">

<col span="1">

</col>