What is the mysql_multicheck

Mysql_multicheck is a Perl application that allows to track MySQL historical information. Mysql_multicheck is not a simile Perl script and it requires all the modules included in the directory to works correctly.

Mysql_multicheck is not suppose to replace the Cacti monitoring tool, but it can/should be use when in the Cacti is not present at the customer.

It is also still “work” in progress and a lot of work is still require to make it more stable and flexible.

Mysql_multicheck is suppose to cover up to MySQL 5.5, and 5.6 but it is still not supporting all the new metrics coming with MySQL 5.6 and the Performance schema.

Mysql_multicheck takes the information from:

* SHOW GLOBAL STATUS;
* SHOW SLAVE/MASTER STATUS;
* SHOW PROCESSLIST;
* SHOW ENGINE INNODB STATUS;
* SYSSTATS

Finally it is an open source project under the “GNU General Public License” so any contribution is more then welcome.

Architecture

Mysql_multicheck is base of :

- **Collector**(s) these are the different collectors that grab information from MySQL and the Machine itself.
- **Mapper** that associate the collected values to internal collections
- **MySQLIndicator container**, which manage the historical series of Indicators

- **MySQL Indicator** that represent the abstraction of the collected metrics value
- **Reporter** the block which generate the reports, including the Live status, the Headers reporter and the Gnuplot generator.

Output

The scope of Mysql_multicheck is to provide Historical information regarding the MySQL Server. It generates a file (or set of file if the SysStat collection is enable) comma separated containing the all the values collected.

It is possible to include such file in the LogRotate script in order to have a daily version of it. Mysql_multicheck has also a tool in its set that allow to split the single file collected by day, if this will be require.

As additional help Mysql_multicheck can also provide information regarding the current activity of the MySQL server, directly at console output.

Which values are collected?

The list of the collected values, change a lot in respect of the MySQL version, and in relation to which set of metrics the monitor will be set to collect.

Because this, there is a special report that can be generate to have the full list of Metrics and their relative position in the comma-separated file(s).

NOTE
I am currently considering the additional output for RRD.

How to Use the collected data?

From the comma-separated file(s), is possible to generate graphs using different tools, like Excel, Chart GnuPlot and so on.

But given that the final file could be significantly large, and contains MILLIONS of values, it will be very difficult to manage.

Another option of the report will generate information on how to SPLIT

the single file in many “ad hoc” thematic files, and then process them separately for the full period or split them by day, using the split tool.

NOTE
GnuPlot is for now the recommended tool for the graphics, and gnuplot script are generated by Mysql_multicheck (again by the reporter) .

MySQL Permission

Given Mysql_multicheck needs to connect to MySQL it needs to have the right privileges, and given the password must be in the script or in the command line or in the configuration file, it will be wise to limit the access as much as possible.

Required and sufficient permission should be SELECT; Replication Client; PROCESS; SHOW DATABASES.

How to use it.

Mysql_multicheck is a Perl application that can run locally on the MySQL server, or it can run on your desktop, so there is no need to perform additional installation of Perl if customer is not willing to do it. Unless you need to collect SysStat metrics, in that case you MUST install it on the target machine.

Said that it requires the basic Perl installation AND the DBI + DBD::mysql library to connect.

All other needed packages are embedded.

Once Perl is installed and up and running, then it will be possible to test if the Mysql_multicheck can correctly connect running:

```
Perl <path to the
installation>/mysql_multicheck/mysqllocalcheck_multiple.pl
-u=monitor  -p=mysql -H=10.11.1.29  -P=3306  --headers=1
```

If the output will report the full list of the headers then this means all is working well. If not check the errors reported and fix the problem.

After that all will become easy and you just need to run Mysql_multicheck, taking few decisions in advance like:

1. How often I want to collect the metrics (suggested 60 seconds)
2. How long I want to keep it running
3. What kind of metrics I want to run

4. Should I run also the console reporter?

Point 3 is require a little more understanding of the tool and I advice you to do not change anything at the begging, given you can do some serious mistakes.

My suggestion is do a short run first to see if you get what you want or not.

So do a run like the following:

```
Perl <path to the
installation>/mysql_multicheck/mysqllocalcheck_multiple.pl
-u=monitor  -p=mysql -H=10.11.1.29  -P=3306 -l=0 -w=1 -i=60 -x=10
--innodb=3 --healtonscreen=1 -o=/tmp/stats.csv -C=1 --sysstats=0
```

Explanation:

```
-u=monitor; User connecting
-p=mysql; Password

-H=10.11.1.29 -P=3306 ; Target ip and port

-l=0; Output format don’t change for now

-w=1; Write file mode Append/Overwrite (1 Overwrite)

-i=60; Interval time in seconds(60 seconds)

-x=10; Number of loops to execute then exit (10 means in this
context 10 minutes)

--innodb=3; Method to collect Innodb statistics (3 is use full set)

--healtonscreen=1; write informative output to console

-o=/tmp/stats.csv; Is the output file for the metrics values

-C=1; Collect also process information

--sysstats=0; Collect SysStats information (0 is disable)
```

A common use for the Mysql_multicheck is in combination of the “screen” tool or with **nohup** like

```
Nohup ./mysqllocalcheck_multiple.pl  -u=monitor  -p=mysql
-H=10.11.1.29  -P=3306 -l=0 -w=1 -i=60 -x=10 --innodb=3
--healtonscreen=1 -o=/tmp/stats.csv -C=1 --sysstats=0 &
```

How to read the csv value file.

From the report using the header option you will get the full list of used metrics, but two fields will always be there no matter what:

1:date
2:time

That's it all entries will have the time split in to field date and time, you will have to re-join the two IF you need to analyze them together.

The rest of the header file will give you information about the other metrics following the same format <id/position> <Metric name>

How to use the reporter to generate the SPLIT file and GNUPLOT script

To have the full set of information, you must run the Mysql_multicheck with the options **–headers=1 –creategnuplot=1**

That will generate TWO set of information:

1. The AWK commands to split the main files into thematic ones
2. The GnuPlot scripts

The AWK section and the gnuplot_graph.ini

The thematic files are generated following the indication and distribution define in the gnuplot_graph.ini file (which MUST be present).

The file follow the standard convention for any *.ini as such:

```
[Bytes]
      yaxis=Mbytes          <-- Label for Y axis
      xaxis=time            <-- Label for X axis
      bytes_received=lines  <-- metric and graph stile
      bytes_sent=lines      <-- metric and graph stile
```

Define the file and graph for the incoming/outgoing traffic, using the bytes_received, bytes_sent metrics.

Reading the gnuplot_graph.ini will tells you what graphs come by default, edit it will allow you to rearrange the metrics as you like.

When you will run the Mysql_multicheck with the headers AND creategnuplot option enable, you will generate information about the statistics fields the tool will generate, the best way to manage it is to

redirect the standard output to a file.

The file will contain 3 sections.

In the first section the list of the headers;

The second section the AWK commands like:

```
awk -F , '{print $1,$2,$176,$177,$178,$179}' stats_all.csv >>
Binlog_cache.csv
awk -F , '{print $1,$2,$180,$181}' stats_all.csv >> Bytes.csv
```

The third section containing the gnuplot scripts:

```
#------------Bytes-----------------------
reset
set title "MBytes sent and received"
set xlabel "time"
set ylabel "Mbytes"
set grid
set border 1
set datafile separator " "
set xdata time
set timefmt "%Y-%m-%d %H:%M:%S"
set key autotitle columnhead
#set logscale # turn on double logarithmic plotting
#set logscale y # for y-axis only
#set logscale x
#set xtics 3800
set xtics 30000
set mxtics 10
#font "arial:name 10:size"
set ytics
set mytics 5
set terminal x11 size 1149,861
# set terminal jpeg size 1149,861  font "arial:name 6:size"
#set timefmt "%H:%M:%S"
set format x "%m-%d %H:%M"
#set output "Bytes.jpg"
plot "Bytes.csv" u 1:(($3/1024)/1024)  w l , "Bytes.csv" u 1:
(($4/1024)/1024)  w l
```

Note that while in the awk commands you will have the list of fields coming from their position in the main CSV value file. In the gnuplot script the position will be relative to the thematic file generated, also the separator will be a space (default for gnuplot) and not a comma, so the extension csv is inappropriate and will be changed soon.

I.e. the byte.csv file will be:

```
date time bytes_received bytes_sent
2012-8-3 9:53:43 0 0
```

```
2012-8-3 9:54:43 1632630 30294989
2012-8-3 9:55:43 2682573 31882554
2012-8-3 9:56:43 3229821 33860130
2012-8-3 9:57:43 873616 19583565
```

Finally in the directory there is an utility named **split_stats.sh** that can be run on the generate output metric file.

Running like split_stats.sh <FILE_name> <PATH>

where PATH is the full path of the file

Will generate in the <PATH> a number of N files split by day.

Generate gnuplot graphs

Please refer to http://www.gnuplot.info/ for installation information.

Once the gnuplot is installed on your PC you only need to run the command:

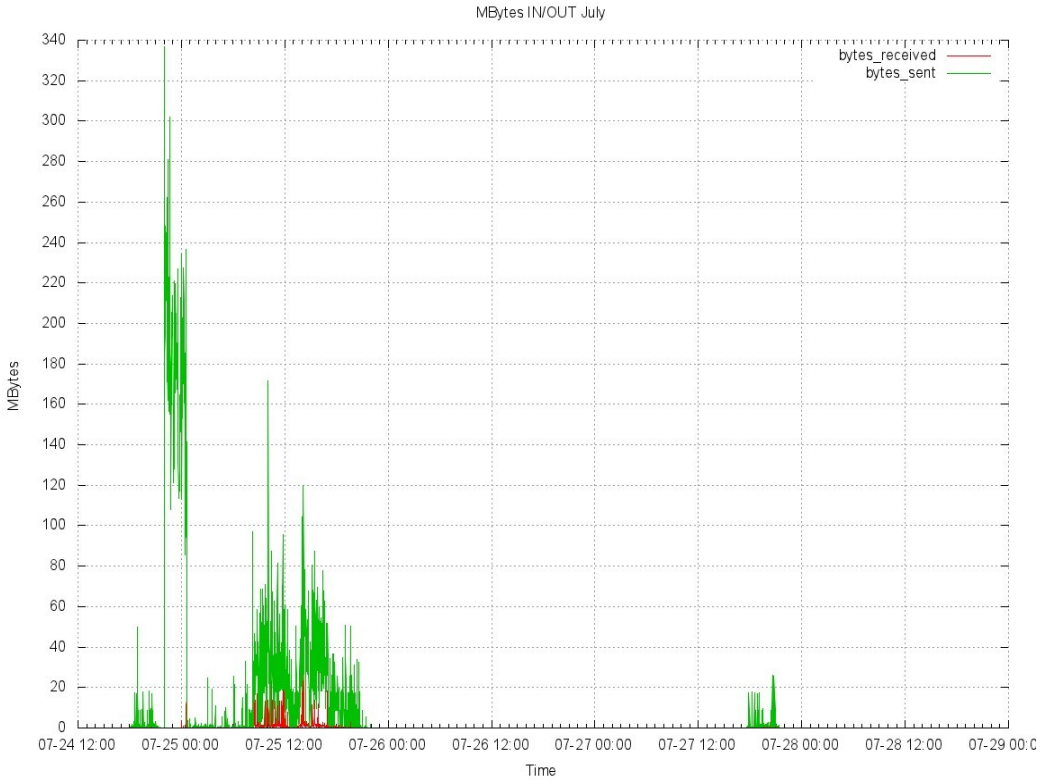
```
tusamac$ gnuplot

      G N U P L O T
      Version 4.4 patchlevel 4
      last modified November 2011
      System: Darwin 10.8.0

      Copyright (C) 1986-1993, 1998, 2004, 2007-2011
      Thomas Williams, Colin Kelley and many others

      gnuplot home:     http://www.gnuplot.info
      faq, bugs, etc:   type "help seeking-assistance"
      immediate help:   type "help"
      plot window:      hit 'h'

Terminal type set to 'x11'
gnuplot>
```

And then either run the scripts as they are, or adapt it to what you need or like.

Running the script will generate a temporary graph, which you can review BEFORE generating the final JPG file.

To Have the JPG generated just do :

```
set terminal x11 size 1149,861  <-- Comment this line
# set terminal jpeg size 1149,861  font "arial:name 6:size" <--
Uncomment this line
#set output "Bytes.jpg"  <-- uncomment this line
```



NOTE
Once you have defined the right set of configuration for a customer, you can keep them for any future reuse, so this one time per customer exercise.

Recommendations

The tool is still in development and some bugs are expected. I really ask all to collect them and to document the steps that generate issues.

Finally this is a tool that could helps in understanding the server behavior but it cannot replace a more stable and consolidate tool as CACTI.

Because this always tries to push customers to use Historical tool and use Mysql_multicheck only as last option.