IPERF3 DATA GRABBER

INSTRUCTIONS

Heiko Eggers

V0-15

This guide will walk you through the installation of the iperf3 data grabber, a tool that captures output from the well-known iperf3 traffic generator test tool. It is enriched with ping test data and uploaded to a mysql database. From there, detailed statistics are displayed through a Grafana dashboard.

I hope you enjoy this guide, please provide feedback to me at h_eggers@yahoo.com.

Note that the grabber is still evolving and hasn't been tested thoroughly. I need help and suggestions to make it better. Please try and play with it and report any bugs, please also check out the github site https://github.com/bloxix/iperf3-data-grabber .

I will use a few formatting conventions in this guide which are briefly described hereafter:

Normal text is used for descriptions.

```
Text in Consolas font with a grey background represents a command, e.g.
ubuntu@ip-172-31-5-177:~$ ./module1b.sh
```

```
Text in Consolas font with a yellow background represents a file content
```

Text in an italic font is a comment to a command or response

```
White text on a black background is console output, e.g
groupadd: group 'docker' already exists
```

TABLE OF CONTENTS

1 BACKGROUND

The use of simple, open source monitoring tools, is still at the center of key networking activities, be it to verify network quality or to perform troubleshooting activities.

Iperf3 is one of these tools, it is based on a client server architecture and allows for the generation of different traffic types and patterns. Iperf captures quite a lot of network data, as it is a command-line driven tool, quick extrapolation and analysis of this data is not simple.

This is where the data grabber comes in, it captures Iperf's JSON based outputs, processes them and populates a mysql database with each single data point. The grabber also ships with Grafana, the dashboard visualization tool, to show essential traffic data in realtime.

2 GRABBER PRINCIPLES

The grabber principles are simple and shown in the picture below.

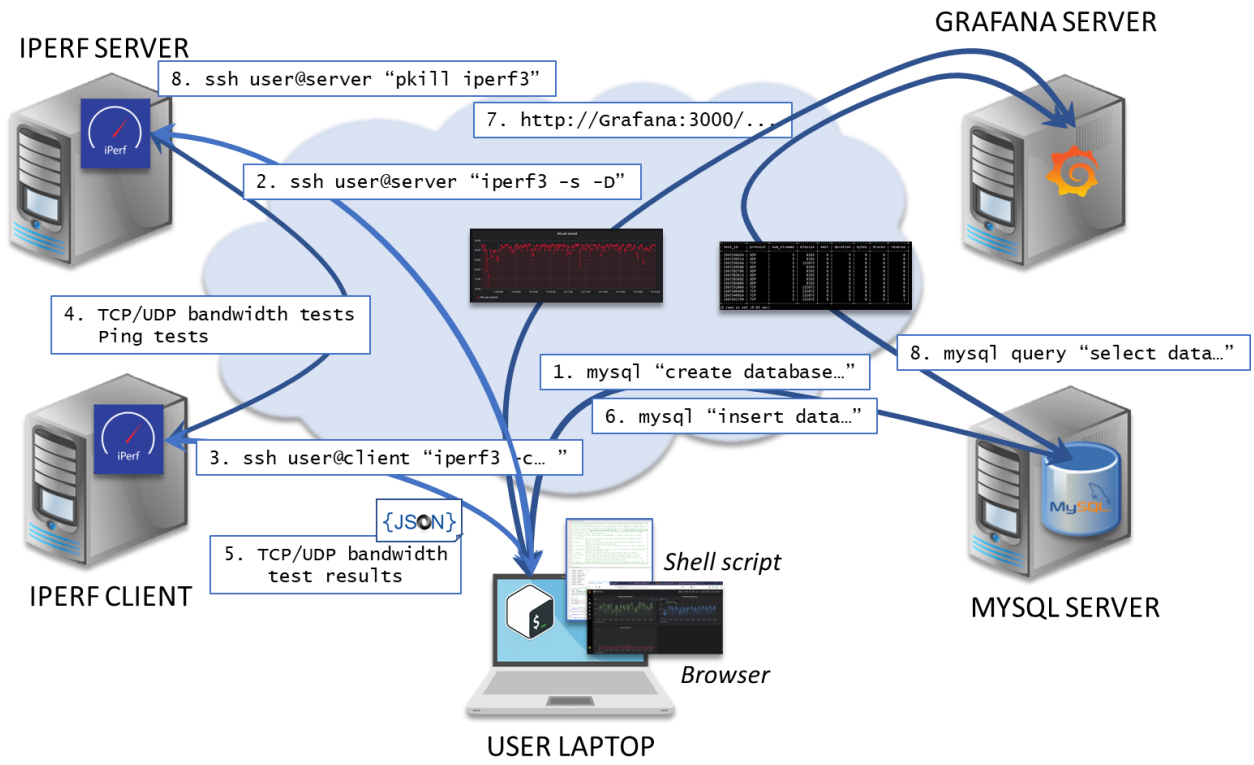


Figure 1 - Grabber principles

The grabber comes with a pre- packaged MySql database, Grafana and OpenSsh server, embedded into a ubuntu base container. A Grafana grabber dashboard and database are also pre- integrated.

Using the grabber is very simple, it comes a s a packaged docker container and requires very few steps to launch from scratch.

The grabber test script is launched through a command line script which is quite similar to iperf itself.

Once that happens

- The grabber connects to the iperf server via ssh and launches the server process.
- It then connects to the iperf client via ssh and launches the iperf client process.
- In parallel, ping tests are started between client and server, and the ping data stored in the mysql database.
- Once the Iperf process is terminated, it returns the test results in JSON format and the MySql database is updated. This process is repeated over the specified number of test runs.
- When the grabber is launched, the Grafana server listens on port 3000 and a grabber dashboard is accessible through a web browser. It manages all queries to the MySql datasource and visualizes traffic data in realtime.
- All data is maintained inside the database for further post-processing.

3 DEPLOY THE GRABBER

3.1 PREREQUISITES

The Grabber is available as a pre- packaged docker image on the docker hub.

All you need, is

- A host or Virtual Machine that runs Docker.
- A client iperf3 host/vm with installed iperf3 SW[1].
- A server iperf3 host/vm with installed iperf3 SW.
- Internet connectivity

3.2 CONTAINER DEPLOYMENT

3.2.1 Deploy container

Deploying the Grabber as a container is easy. At the prompt of your docker host, type:

```
docker run -it -p3000:3000 --hostname grabber --user iperf3 bloxix/iperf3-data-grabber:v0151 /bin/bash
```

The first time you run it you should see this screen…

[1] https://iperf.fr/iperf-download.php , https://github.com/esnet/iperf

```
iperf3@ubuntu:~$ docker run -it -p3000:3000 --hostname grabber --user iperf3 bloxix/iperf3-data-grabber:v015 /bin/bash
Unable to find image 'bloxix/iperf3-data-grabber:v015' locally
v015: Pulling from bloxix/iperf3-data-grabber
5667fdb72017: Pull complete
d83811f270d5: Pull complete
ee671aafb583: Pull complete
7fc152dfb3a6: Pull complete
ad04dae92804: Pull complete
29d4ad55dac8: Pull complete
64acb68982b5: Pull complete
8c2e8905fa15: Pull complete
1628aa3cb0f9: Pull complete
ff4bb8c5e5d7: Pull complete
Digest: sha256:84782af1713963fffe893d5d032aaab651c59b2b5252edeca71cbf00fced5544
Status: Downloaded newer image for bloxix/iperf3-data-grabber:v015
iperf3@grabber:/$
```

Finally it will open up an interactive shell inside the container on host grabber as user iperf3. Grafana's listen port 3000 is mapped to its host.

3.2.2 Check container connectivity

Make sure you reach the internet from inside the container.

```
iperf3@grabber:~/iperf3$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=21.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=22.5 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 21.217/21.864/22.512/0.664 ms
```

Make sure you can reach both iperf server and client.

In the example below, the server and client have IP addresses 192.168.0.81 and 192.168.0.22, respectively, and the user names are iperf3.

```
iperf3@grabber:~/iperf3$ ping 192.168.0.81
PING 192.168.0.81 (192.168.0.81) 56(84) bytes of data.
64 bytes from 192.168.0.81: icmp_seq=1 ttl=63 time=22.7 ms
^C
--- 192.168.0.81 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.791/22.791/22.791/0.000 ms
iperf3@grabber:~/iperf3$ ping 192.168.0.22
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_seq=1 ttl=63 time=0.705 ms
64 bytes from 192.168.0.22: icmp_seq=2 ttl=63 time=0.334 ms
64 bytes from 192.168.0.22: icmp_seq=3 ttl=63 time=0.393 ms
^C
--- 192.168.0.22 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.334/0.477/0.705/0.163 ms
```

3.2.3 Generate and upload public SSL key

Given that the grabber will possibly connect to the iperf client and server multiple times, we want to make sure this process is automated and therefore will want to generate a public SSL key and upload it to the iperf server and client public key files.

Type

```
iperf3@grabber:~/iperf3$ ssh-keygen
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/iperf3/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/iperf3/.ssh/id_rsa.
Your public key has been saved in /home/iperf3/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:IF+pzt91kxyhO/wLFRx5Ibr5Bj1s7OAo7Au1VsR3kr8 iperf3@grabber
The key's randomart image is:
+---[RSA 2048]----+
|            ..o.|
|        ..  o.o..|
|    . . oo = .+. |
|     o +. . X. o |
|      o.S. =.Bo  |
|     oo o o.*+oo |
|     .o= . .*E=  |
|      +... ..= . |
|       oo .   o. |
+----[SHA256]-----+
```

Accept all defaults by pressing <ENTER> when prompted.

Copy your public key to the two hosts:

```
iperf3@grabber:~/iperf3$ ssh-copy-id -i /home/iperf3/.ssh/id_rsa.pub 192.168.0.22
```

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/iperf3/.ssh/id_rsa.pub"
The authenticity of host '192.168.0.22 (192.168.0.22)' can't be established.
ECDSA key fingerprint is SHA256:7k6Fh7tufHc4kTCMitkXC1+du2TIwTMAJygL77Dx/DA.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
iperf3@192.168.0.22's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.0.22'"
and check to make sure that only the key(s) you wanted were added.
```

```
iperf3@grabber:~/iperf3$ ssh-copy-id -i /home/iperf3/.ssh/id_rsa.pub 192.168.0.81
```

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/iperf3/.ssh/id_rsa.pub"
```

```
The authenticity of host '192.168.0.81 (192.168.0.81)' can't be established.
ECDSA key fingerprint is SHA256:JKb2AXCS26qBbyqYR7TrJKSqinSDxkEIqr8sidWxBco.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
iperf3@192.168.0.81's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.0.81'"
and check to make sure that only the key(s) you wanted were added.
```

3.2.4 Update the grabber configuration:

```
iperf3@grabber:~/iperf3$ cd ~/iperf3

iperf3@grabber:~/iperf3$ vi t15.sh
```

Change your iperf3 client and server user names and ip addresses to match your iperf setup. Scroll down to the configuration section.

```
# [-t time]      specifies the time of each test run in seconds, same as in ipe
# [-P Parallel]  streams, specifies the number of concurrent streams, same as i
# [-R ]          will run tests in reverse mode (server transmits), same as in
# [-o other]     allows you to include other iperf switches, that will be passe
#                command line. Note this option has not been tested.
# [-r runs]      specifies the number of test runs the tool will carry out. Def
################################################################################

################################################################################
#Variables
#Define Default values used by IPERF and other script variables
################################################################################

IPERF3_LOCAL_RUN=1                    # 0 = run the iperf client from the loca
IPERF3_CLIENT='192.168.0.22'             # IPERF3_CLIENT ip address
IPERF3_CLIENT_USER='iperf3'           # used to login to IPERF3_CLIENT
IPERF3_SERVER='192.168.0.81'             # IPERF3_SERVER ip address
IPERF3_SERVER_USER='iperf3'           # used to login to IPERF3_SERVER
IPERF3_PORT=5201                      # default port
IPERF3_PROTOCOL='tcp'                 # default 'tcp' or 'udp'
IPERF3_BANDWIDTH=0                    # default 0 - max bandwidth
IPERF3_TIME=10                        # default 10 seconds per test run
```

Figure 2 – Grabber script configuration parameters

3.2.5 Start grabber processes

We need to start the mysql, openssh and Grafana servers. The simple commands are listed in the start_grabber script we must now launch…

```
iperf3@grabber:~/iperf3$ /home/iperf3/start_grabber.sh
```

```
 * Starting Grafana Server                                                    [ OK ]
 * Starting OpenBSD Secure Shell server sshd                                  [ OK ]
 * Starting MySQL database server mysqld
No directory, logging in with HOME=/
                                                                             [ OK ]
```

3.2.6 Now run a first test…

```
iperf3@grabber:~/iperf3$ ./t15.sh -r 10 -P 3 -t 5
```

You should see something like this…

```
Option -t "5" specified
Option -P "3" specified
Option -r "10" specified
will execute the command : iperf3 -c 192.168.0.81 -p 5201 -b 0 -t 5 -P 3 -J
over 10 test runs...
starting iperf server on 192.168.0.81
will execute ssh iperf3@192.168.0.81 'iperf3 -s -p 5201 -D'
will execute the following sql query:
INSERT INTO iperf3_general_info
(test_id,test_time,test_name,test_user,iperf3_client,iperf3_server,test_runs,args,test_complet
e) VALUES ("1570980032","2019-10-13 15:20:32","IPERF3_TEST
1570980032","iperf3","192.168.0.22","192.168.0.81","10","-c 192.168.0.81 -p 5201 -b 0 -t 5 -P
3 -J ",false);
(…)
```

3.2.7 Check Grafana…

Let’s check our dashboard now:

Open a browser on port 3000 of your docker host.

Figure 3 – Grafana login screen

Login with admin/admin.

Skip password update.

Select dashboards->manage.

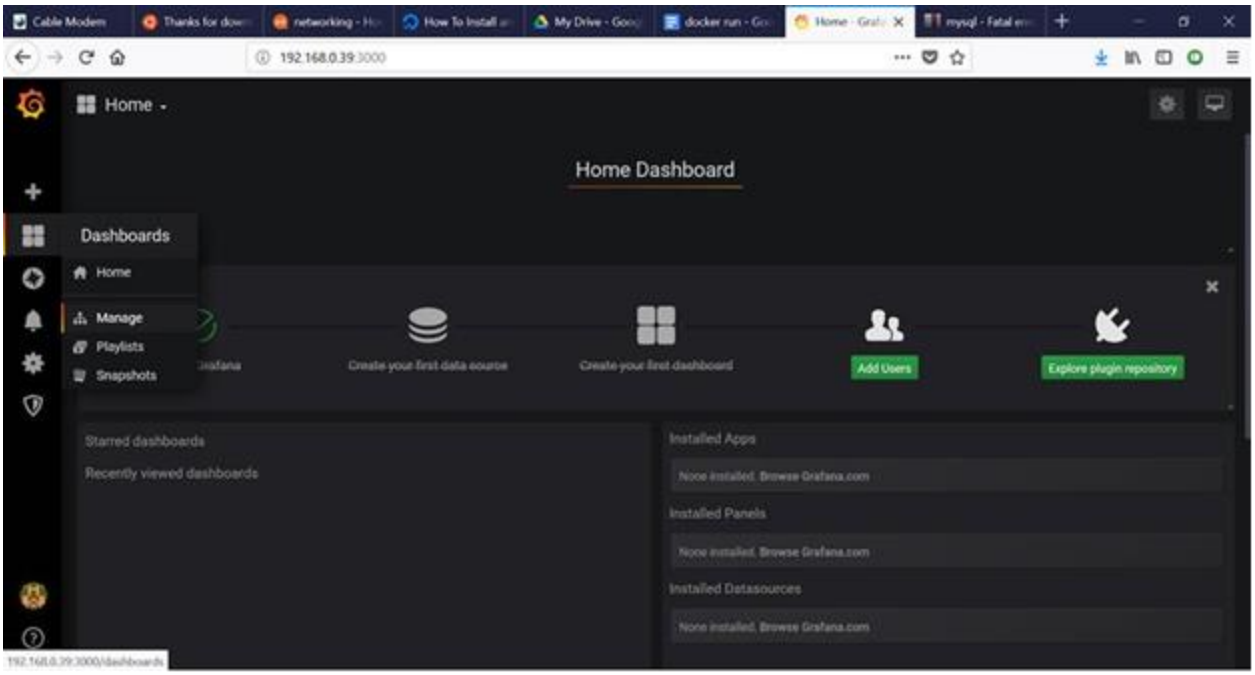
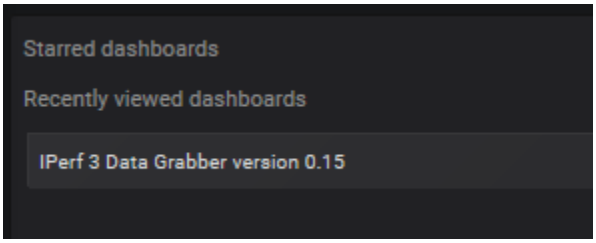


Figure 4 – Manage dashboard

Select the IPerf3 Data Grabber dashboard…



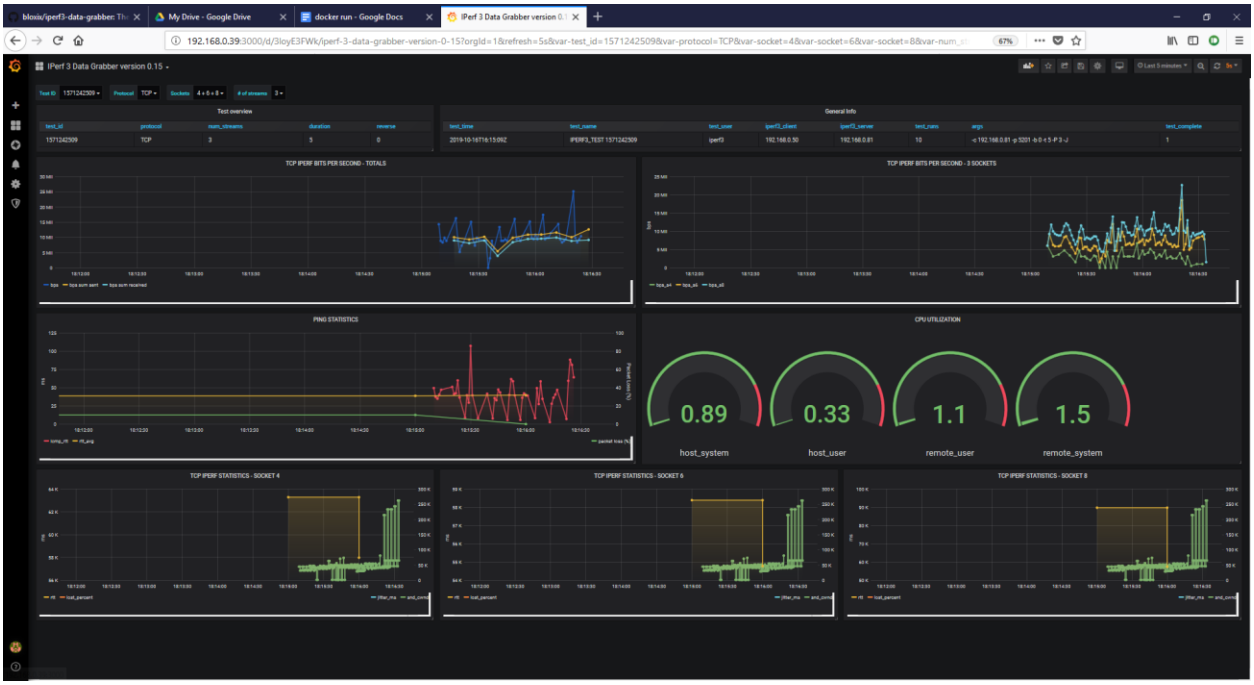


Figure 5 – Grabber dashboard first test

You should see the Grafana dashboard building up as shown above.

4 GRABBER COMMAND LINE INTERFACE

4.1 GRABBER ARGUMENTS

The tool is run in client mode and uses the same (optional) command line arguments as iperf3 where applicable:

The general format is:

```
t15.sh [-c host_ip] [-p port] [-x protocol] [-b bandwidth] [-t time] [-P Parallel] [-
o other] [-r runs] [-R] [-C client_host_ip]
```

[-c host_ip] specifies the destination host where the iperf server component resides. Default is localhost.

[-C chost_ip] specifies the client host where the iperf client component resides. Default is localhost.

[-p port] specifies the destination host where the iperf server component resides. Default is the iperf3 default port 5201.

[-x protocol] specifies the protocol to be used, which can be 'tcp' or 'udp'. If omitted, 'tcp' will be used.

[-b bandwidth] specifies the bandwidth, same as in iperf. if omitted, the maximum available bandwidth will be used (-b 0).

[-t time] specifies the time of each test run in seconds, same as in iperf.

[-P Parallel] streams, specifies the number of concurrent streams, same as in iperf.

[-R] will run tests in reverse mode (server transmits), same as in iperf.

[-o other] allows you to include other iperf switches, that will be passed over the command line. *Note this option has not been tested*!

[-r runs] specifies the number of test runs the grabber will carry out. Default is one.

4.2 GRABBER RUN SAMPLES

Hereafter a couple of examples

Max BW TCP test from 192.168.0.81 to 192.168.0.50 over 10 test runs of 15 seconds each with 3 parallel streams:

```
iperf3@grabber:~/iperf3$ ./t15.sh -r 10 -t 15 -c 192.168.0.50 -C 192.168.0.81 -P 3
```

Max BW reverse UDP test:

```
iperf3@grabber:~/iperf3$ ./t15.sh -r 10 -t 15 -c 192.168.0.50 -C 192.168.0.81 -P 3 -x
'udp' -R
```

5 GRABBER DASHBOARD

The grabber dashboard visualizes several panels related to the latest ongoing test.

In edit mode, it displays the following screen:

Figure 6 - Grabber dashboard – TCP test

Figure 7 - Grabber dashboard – UDP test

5.1 DASHBOARD STRUCTURE OVERVIEW

The dashboard structure is shown as follows:

- The tables at the top row of the dashboards provide details about the latest run test.
- The two main panels show the total throughput, in terms of totals as well as the single streams.
- The panels underneath summarize the ping statistics between iperf client and server, as well as the CPU utilization statistics.
- Finally, the bottom row holds panels for each stream, and they change in accordance with the selected protocol (rtt and snd_cnwd for tcp and jitter and packet loss for udp).

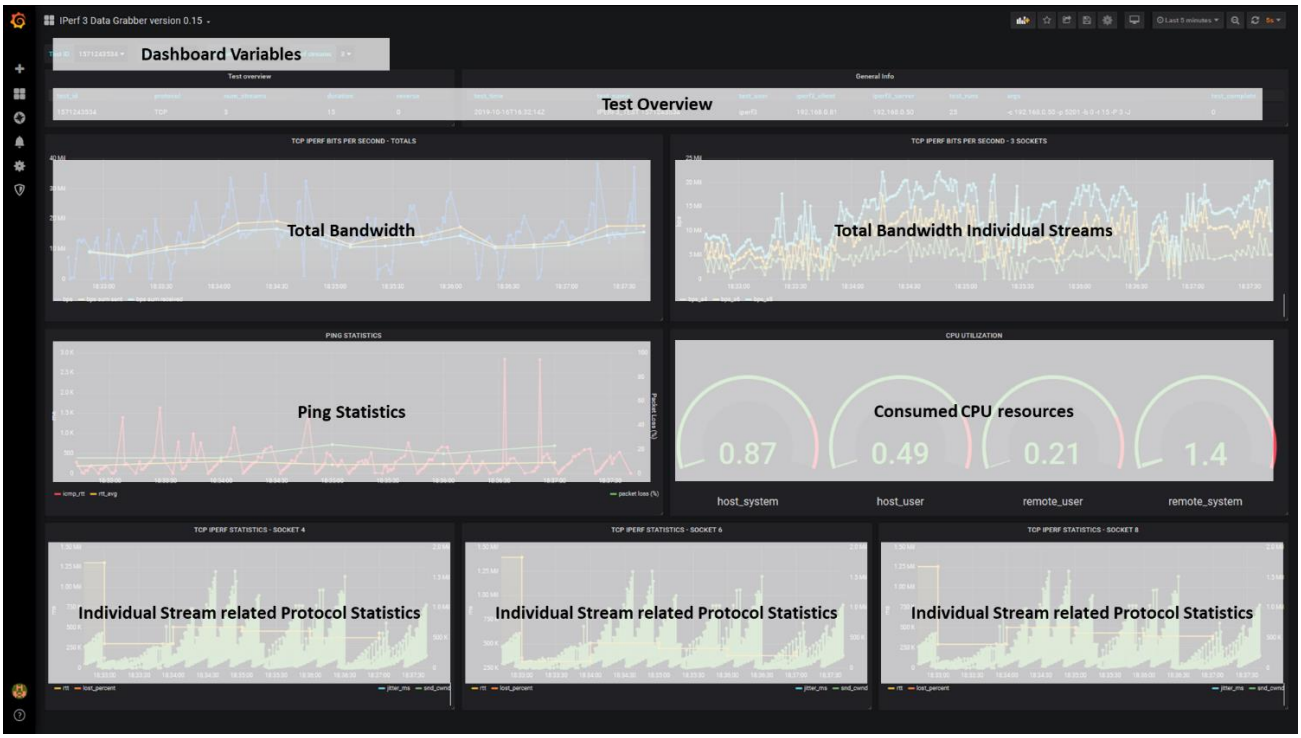


Figure 8 - Grabber dashboard structure

5.2 MAIN PANELS

5.2.1 Total Bandwidth

The figure below shows an example of the total bandwidth panel.

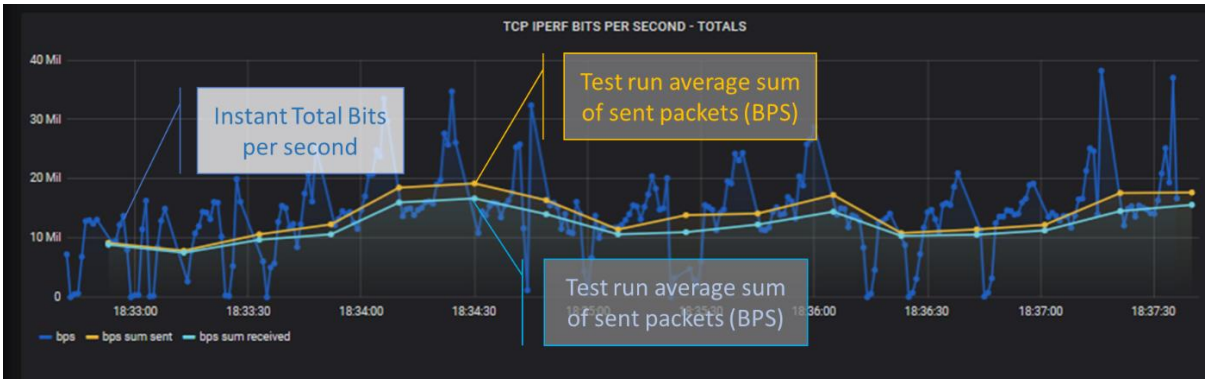


Figure 9 – Total bandwidth panel

The "Total Bandwidth" panel contains three data series:

- The single total bits-per-second entries fetched from the JSON intervals sum outputs
- The average sent and received bits per second (TCP) or the average sum of packets (UDP) fetched from the end

In the example above, a TCP test is performed with test runs of 3 sockets and 15 seconds per test run is performed. Performance drops every 15 seconds can be observed, due to the TCP slow-start algorithm. Moreover, sent packets are slightly higher than received packets which shows some retransmissions in the network.

The second panel shows a stacked view of the BW contributions of the individual streams, in our example these are three sockets.

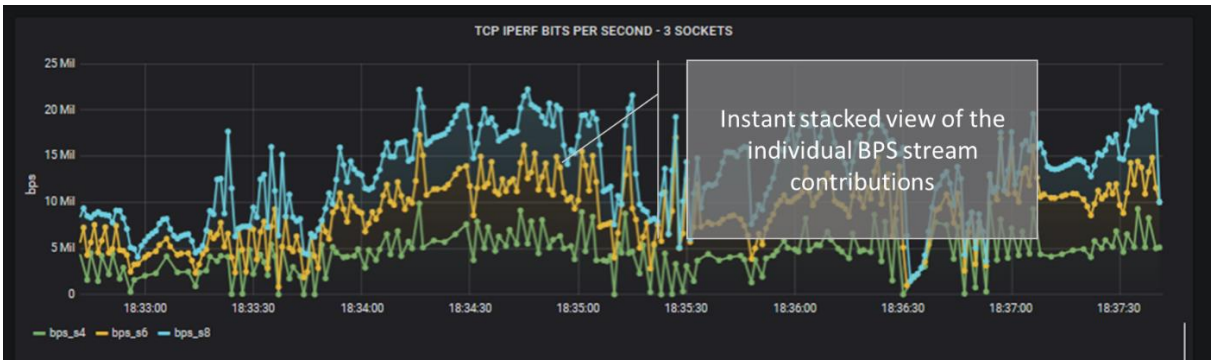


Figure 10 – Total bandwidth from individual streams.

The panel will display up to 4 sockets.

The next panel shows ping statistics:

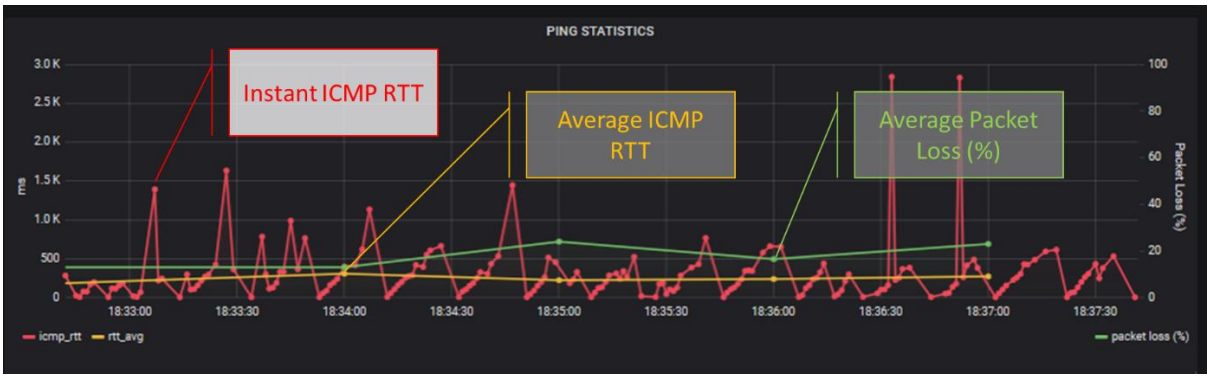


Figure 11 – Ping statistics

They include instantaneous ping rtt values, average and packet loss values of the latest run.

The next panel shows the CPU utilization statistics, as fetched from the JSON output provided by iperf at the end of each test run.

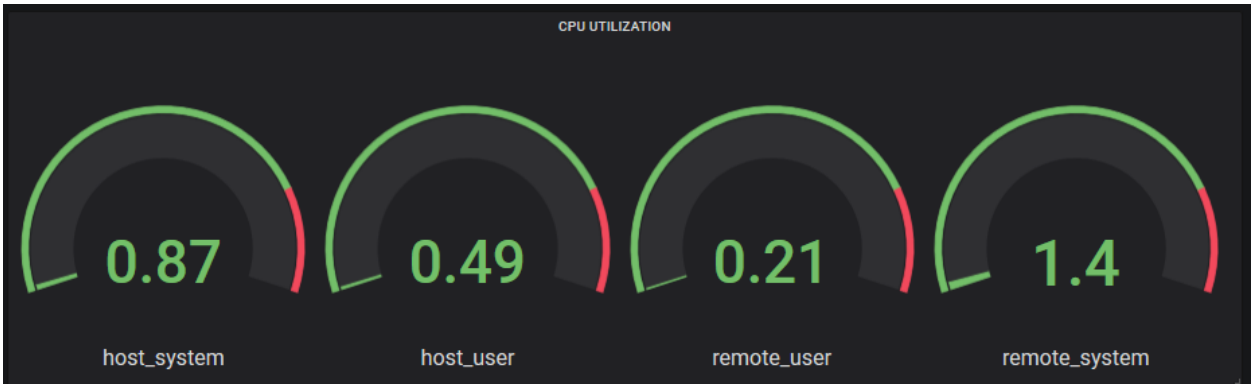


Figure 12 – CPU Utilization panel

It distinguishes CPU consumption of host system and user processes, as well as remote system and user processes.

6 Grabber Data and Table Structures

6.1 Overview

The grabber contains a one-to-one mapping of all JSON data that is produced by iperf, in addition to all ping data test results, that are also stored in the database. Each test is uniquely identified by a test_id value, which is common in all tables that refer to a single test.

Iperf3 structures its output based on three categories, classified as start, intervals and end.

Start data is produced at the beginning of the test, once for each test run, intervals data is produced for every interval in which iperf generates traffic, end data is produced as a summary aggregate, at the end of every test.

In addition, ping statistics for each ping event are captured in a ping table and the ping summary data at the end of each test run in a ping summary table.

6.2 DETAILED TABLE STRUCTURE

The detailed table structure of all tables is given hereafter.

6.2.1 General Info table

General info contains key information about the test.

```
DESCRIBE iperf3_general_info;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| test_id      | int(11)     | NO   | PRI | NULL    |       |
| test_time    | datetime    | YES  |     | NULL    |       |
| test_name    | varchar(255)| YES  |     | NULL    |       |
| test_user    | varchar(255)| YES  |     | NULL    |       |
| iperf3_client| varchar(20) | YES  |     | NULL    |       |
| iperf3_server| varchar(20) | YES  |     | NULL    |       |
| test_runs    | int(11)     | YES  |     | NULL    |       |
| args         | varchar(255)| YES  |     | NULL    |       |
| test_complete| tinyint(1)  | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

A table entry is generated once for each test:

test_id	contains the unique test identification, which is a UNIX timestamp.
test_time	date and time of the start of the test.
test_name	String IPERF3_TEST and unix timestamp
iperf3_client	IP address of the Iperf3 client
iperf3_server	IP address of the Iperf3 server
test_runs	Total number of test runs
args	Arguments used in the iperf3 cli command.
test_complete	The test_complete parameter is set to 0 (false) until the test reaches completion, when it will be set to 1 (true).

6.2.2 Start tables

Start tables are transposed directly from Iperf JSON start data and generated at the beginning of each test run.

```
DESCRIBE iperf3_start;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| test_id         | int(11)     | NO   | MUL | NULL    |       |
| version         | varchar(20) | YES  |     | NULL    |       |
| system_info     | varchar(255)| YES  |     | NULL    |       |
| cookie          | varchar(255)| YES  |     | NULL    |       |
| tcp_mss_default | int(11)     | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

test_id	contains the unique test identification, which is a UNIX timestamp.
version	iperf version used in the test
system_info	Linux Kernel information
cookie	Cookie value
tcp_mss_default	Maximum default TCP Segment Size. This parameter is set to NULL in case of UDP tests.

The iperf3_start_connected contains iperf test connection details

```
DESCRIBE iperf3_start_connected;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| test_id     | int(11)     | NO   | MUL | NULL    |       |
| socket      | int(11)     | YES  |     | NULL    |       |
| local_host  | varchar(127)| YES  |     | NULL    |       |
| local_port  | int(11)     | YES  |     | NULL    |       |
| remote_host | varchar(127)| YES  |     | NULL    |       |
| remote_port | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

The iperf3_start_timestamp table contains details about the timestamp generated by iperf itself.

```
DESCRIBE iperf3_start_timestamp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| test_id  | int(11)     | NO   | MUL | NULL    |       |
| time     | varchar(31) | YES  |     | NULL    |       |
| timesecs | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
DESCRIBE iperf3_start_connecting_to;
mysql: [Warning] Using a password on the command line interface can be insecure.
```

```
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| test_id | int(11)      | NO   | MUL | NULL    |       |
| host    | varchar(127) | YES  |     | NULL    |       |
| port    | int(11)      | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

will execute the following sql query:

```
DESCRIBE iperf3_start_test_start;
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| test_id     | int(11)     | NO   | MUL | NULL    |       |
| protocol    | varchar(10) | YES  |     | NULL    |       |
| num_streams | int(11)     | YES  |     | NULL    |       |
| blksize     | int(11)     | YES  |     | NULL    |       |
| omit        | tinyint(1)  | YES  |     | NULL    |       |
| duration    | int(11)     | YES  |     | NULL    |       |
| bytes       | int(11)     | YES  |     | NULL    |       |
| blocks      | int(11)     | YES  |     | NULL    |       |
| reverse     | tinyint(1)  | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

6.2.3 Intervals tables

Data is added to Intervals tables at each iperf test interval. TCP and UDP test data is integrated into a single table. Non applicable values are set to NULL. Interval streams table hold data based on each specific socket, Interval sum data holds the sum of all individual streams.

```
DESCRIBE iperf3_intervals_streams;
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----------------+------------+------+-----+---------+-------+
| Field           | Type       | Null | Key | Default | Extra |
+-----------------+------------+------+-----+---------+-------+
| time            | datetime   | YES  |     | NULL    |       |
| test_id         | int(11)    | NO   | MUL | NULL    |       |
| test_run        | int(11)    | YES  |     | NULL    |       |
| socket          | int(11)    | YES  |     | NULL    |       |
| start           | double     | YES  |     | NULL    |       |
| end             | double     | YES  |     | NULL    |       |
| seconds         | double     | YES  |     | NULL    |       |
| bytes           | bigint(20) | YES  |     | NULL    |       |
| bits_per_second | bigint(20) | YES  |     | NULL    |       |
| retransmits     | int(11)    | YES  |     | NULL    |       |
| snd_cwnd        | int(11)    | YES  |     | NULL    |       |
| rtt             | int(11)    | YES  |     | NULL    |       |
| jitter_ms       | double     | YES  |     | NULL    |       |
| lost_packets    | int(11)    | YES  |     | NULL    |       |
| packets         | int(11)    | YES  |     | NULL    |       |
| lost_percent    | double     | YES  |     | NULL    |       |
| omitted         | tinyint(1) | YES  |     | NULL    |       |
+-----------------+------------+------+-----+---------+-------+
```

```
DESCRIBE iperf3_intervals_sum;
mysql: [Warning] Using a password on the command line interface can be insecure.
+----------------+------------+------+-----+---------+-------+
| Field          | Type       | Null | Key | Default | Extra |
+----------------+------------+------+-----+---------+-------+
| time           | datetime   | YES  |     | NULL    |       |
| test_id        | int(11)    | NO   | MUL | NULL    |       |
| test_run       | int(11)    | YES  |     | NULL    |       |
| start          | double     | YES  |     | NULL    |       |
| end            | double     | YES  |     | NULL    |       |
| seconds        | double     | YES  |     | NULL    |       |
| bytes          | bigint(20) | YES  |     | NULL    |       |
| bits_per_second | bigint(20) | YES  |     | NULL    |       |
| retransmits    | int(11)    | YES  |     | NULL    |       |
| jitter_ms      | double     | YES  |     | NULL    |       |
| lost_packets   | int(11)    | YES  |     | NULL    |       |
| packets        | int(11)    | YES  |     | NULL    |       |
| lost_percent   | double     | YES  |     | NULL    |       |
| omitted        | tinyint(1) | YES  |     | NULL    |       |
+----------------+------------+------+-----+---------+-------+
```

6.2.4 End tables

End tables are populated at the end of each test run.

```
DESCRIBE iperf3_end_streams;
+----------------+------------+------+-----+---------+-------+
| Field          | Type       | Null | Key | Default | Extra |
+----------------+------------+------+-----+---------+-------+
| time           | datetime   | YES  |     | NULL    |       |
| test_id        | int(11)    | NO   | MUL | NULL    |       |
| test_run       | int(11)    | YES  |     | NULL    |       |
| sender         | tinyint(1) | YES  |     | NULL    |       |
| socket         | int(11)    | YES  |     | NULL    |       |
| start          | double     | YES  |     | NULL    |       |
| end            | double     | YES  |     | NULL    |       |
| seconds        | double     | YES  |     | NULL    |       |
| bytes          | bigint(20) | YES  |     | NULL    |       |
| bits_per_second | bigint(20) | YES  |     | NULL    |       |
| retransmits    | int(11)    | YES  |     | NULL    |       |
| max_snd_cwnd   | int(11)    | YES  |     | NULL    |       |
| max_rtt        | int(11)    | YES  |     | NULL    |       |
| min_rtt        | int(11)    | YES  |     | NULL    |       |
| mean_rtt       | int(11)    | YES  |     | NULL    |       |
| jitter_ms      | double     | YES  |     | NULL    |       |
| lost_packets   | bigint(20) | YES  |     | NULL    |       |
| packets        | bigint(20) | YES  |     | NULL    |       |
| lost_percent   | double     | YES  |     | NULL    |       |
| out_of_order   | bigint(20) | YES  |     | NULL    |       |
+----------------+------------+------+-----+---------+-------+
```

```
DESCRIBE iperf3_end_sum;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| time            | datetime    | YES  |     | NULL    |       |
| test_id         | int(11)     | NO   | MUL | NULL    |       |
| test_run        | int(11)     | YES  |     | NULL    |       |
| type            | varchar(20) | YES  |     | NULL    |       |
| start           | double      | YES  |     | NULL    |       |
| end             | double      | YES  |     | NULL    |       |
| seconds         | double      | YES  |     | NULL    |       |
| bytes           | bigint(20)  | YES  |     | NULL    |       |
| bits_per_second | bigint(20)  | YES  |     | NULL    |       |
| retransmits     | int(11)     | YES  |     | NULL    |       |
| jitter_ms       | double      | YES  |     | NULL    |       |
| lost_packets    | bigint(20)  | YES  |     | NULL    |       |
| packets         | bigint(20)  | YES  |     | NULL    |       |
| lost_percent    | double      | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

```
DESCRIBE iperf3_end_cpu_utilization;
mysql: [Warning] Using a password on the command line interface can be insecure.
+---------------+----------+------+-----+---------+-------+
| Field         | Type     | Null | Key | Default | Extra |
+---------------+----------+------+-----+---------+-------+
| time          | datetime | YES  |     | NULL    |       |
| test_id       | int(11)  | NO   | MUL | NULL    |       |
| test_run      | int(11)  | YES  |     | NULL    |       |
| host_total    | double   | YES  |     | NULL    |       |
| host_user     | double   | YES  |     | NULL    |       |
| host_system   | double   | YES  |     | NULL    |       |
| remote_total  | double   | YES  |     | NULL    |       |
| remote_user   | double   | YES  |     | NULL    |       |
| remote_system | double   | YES  |     | NULL    |       |
+---------------+----------+------+-----+---------+-------+
```

6.2.5 Ping tables

The ping tables contain all ping statistics. Data is captured both for each individual ping as well as for the ping summary.

```
DESCRIBE iperf3_ping_table;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| time        | datetime    | YES  |     | NULL    |       |
| test_id     | int(11)     | NO   | MUL | NULL    |       |
| test_run    | int(11)     | YES  |     | NULL    |       |
| remote_host | varchar(20) | YES  |     | NULL    |       |
| bytes       | int(11)     | YES  |     | NULL    |       |
| icmp_seq    | int(11)     | YES  |     | NULL    |       |
```

```
| ttl          | int(11)     | YES  |     | NULL    |       |
| icmp_rtt     | double      | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

```
DESCRIBE iperf3_ping_summary_table;
+---------------------+-------------+------+-----+---------+-------+
| Field               | Type        | Null | Key | Default | Extra |
+---------------------+-------------+------+-----+---------+-------+
| time                | datetime    | YES  |     | NULL    |       |
| test_id             | int(11)     | NO   | MUL | NULL    |       |
| test_run            | int(11)     | YES  |     | NULL    |       |
| remote_host         | varchar(20) | YES  |     | NULL    |       |
| remote_host2        | varchar(20) | YES  |     | NULL    |       |
| bytes_pp_sent       | int(11)     | YES  |     | NULL    |       |
| bytes_pp_received   | int(11)     | YES  |     | NULL    |       |
| packets_transmitted | int(11)     | YES  |     | NULL    |       |
| packets_received    | int(11)     | YES  |     | NULL    |       |
| packet_loss         | double      | YES  |     | NULL    |       |
| total_time          | int(11)     | YES  |     | NULL    |       |
| rtt_min             | double      | YES  |     | NULL    |       |
| rtt_avg             | double      | YES  |     | NULL    |       |
| rtt_max             | double      | YES  |     | NULL    |       |
| rtt_mdev            | double      | YES  |     | NULL    |       |
+---------------------+-------------+------+-----+---------+-------+
```