TED Replication by Galera
Experimental test in lab
Report #7

Maziar Sedghisaray (524923)
Maziar.sedghisaray2@gmail.com
Master in Computer Science and Networking (mcsn)

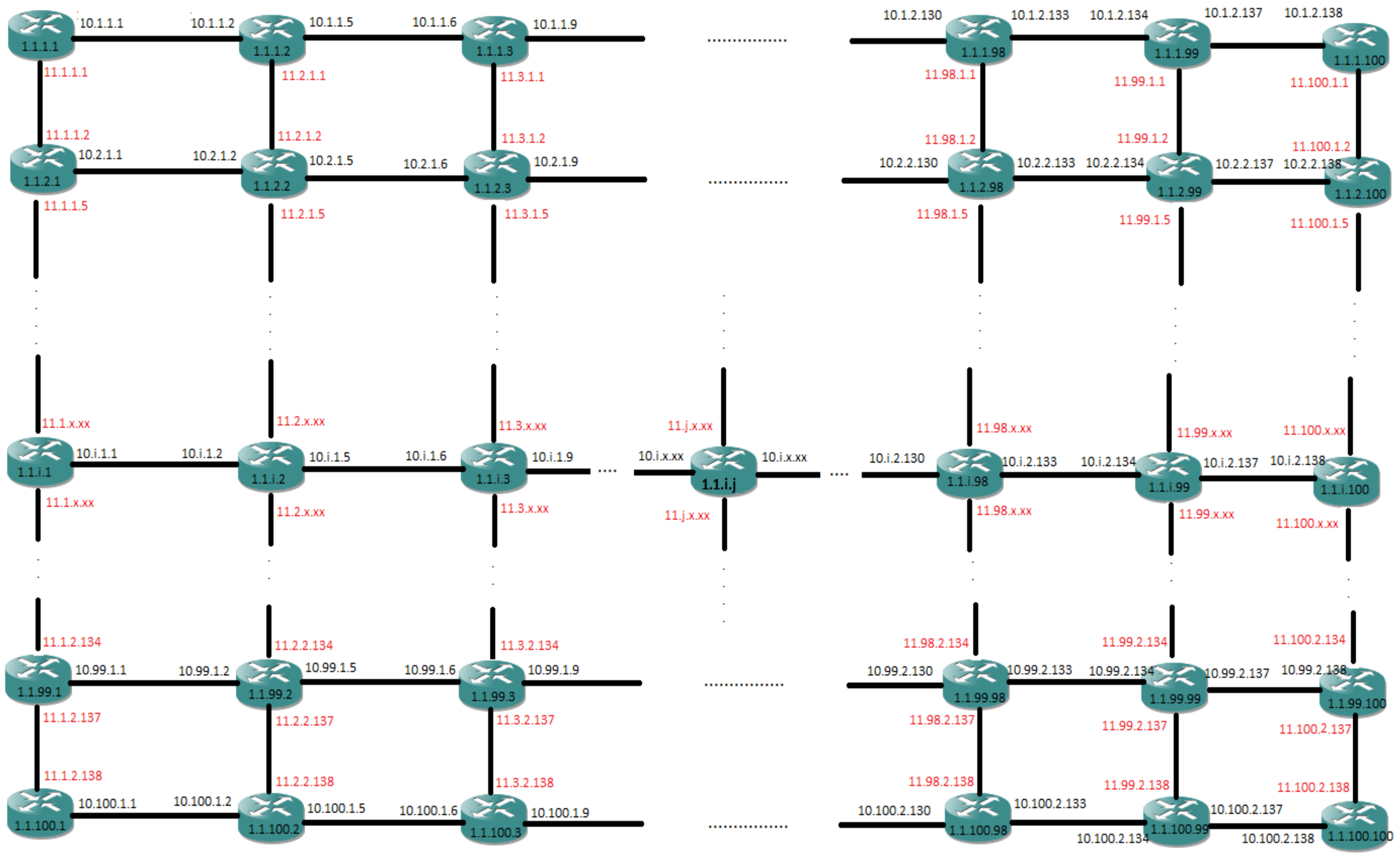
Testbed Configuration

CentOS Linux release 7.3.1611 (Core) 3.10.0-514.6.1.el7.x86_64
Server version: 10.0.29-MariaDB-wsrep MariaDB Server, wsrep_25.16.rc3fc46e

Each VM has 2 cores 2GB of RAM 8GB disk

- SYNC_BINLOG=0
- Innidb_flush_log_at_trx_commit =2

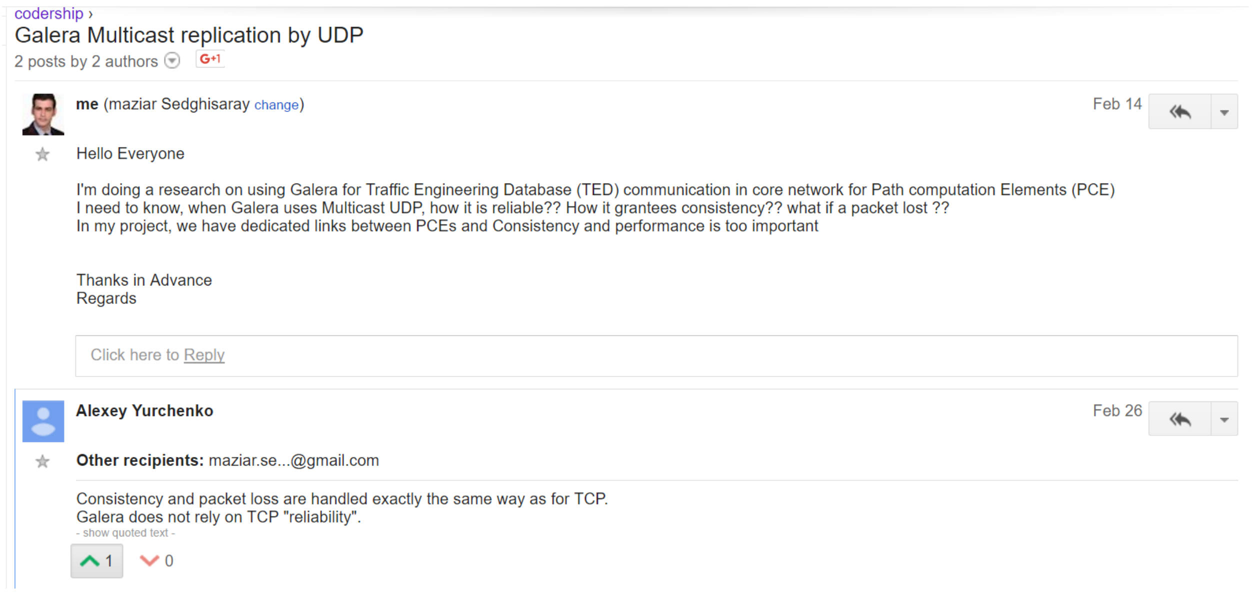
Topology Configuration



Insert Performance Evaluation

Inserting a whole Row of data which includes information about **Router ID**, **Link ID**, **Link Type**, **Local Interface Address**, **Remote Interface Address**, **Traffic Engineering metric**, **Maximum Bandwidth**, **Unreserved Bandwidth for 8 different priority**, and **Administration Group**. Exactly same items which we defined in TED Database based on IETF reference model.

I notice that Inserting Data takes around 0.5 ms for each ROW of data. Size of database has not any effect on duration of Insert because it appends the new data to table. However there is some points which takes more 0.5 ms which is unavoidable like the very first 2 Insert takes about 1 to 2 ms. That time is for preparing InnoDB cache and also when there is many Insert continuously (more than 2000 Insert), at some pint InnoDB cache must copy to InnoDB Disk and prepare again the cache for Inserting rest of data. That takes about 25 to 30 ms. However still I'm not sure about that, it can be due to TCP congestion avoidance also.
For that reason, in the next step I'm going to redo the test with Multicast and UDP
I discussed with codership group in Finland (The group who developed Galera) I asked the how Galera grantees reliability and consistency with UDP and the answer was:
" Consistency and packet loss are handled exactly the same way as for TCP.
Galera does not rely on TCP "reliability".



codership ›

Galera Multicast replication by UDP

2 posts by 2 authors

G+1

me (maziar Sedghisaray change) Feb 14

Hello Everyone

I'm doing a research on using Galera for Traffic Engineering Database (TED) communication in core network for Path computation Elements (PCE)
I need to know, when Galera uses Multicast UDP, how it is reliable?? How it grantees consistency?? what if a packet lost ??
In my project, we have dedicated links between PCEs and Consistency and performance is too important

Thanks in Advance
Regards

Click here to Reply

Alexey Yurchenko Feb 26

Other recipients: maziar.se...@gmail.com

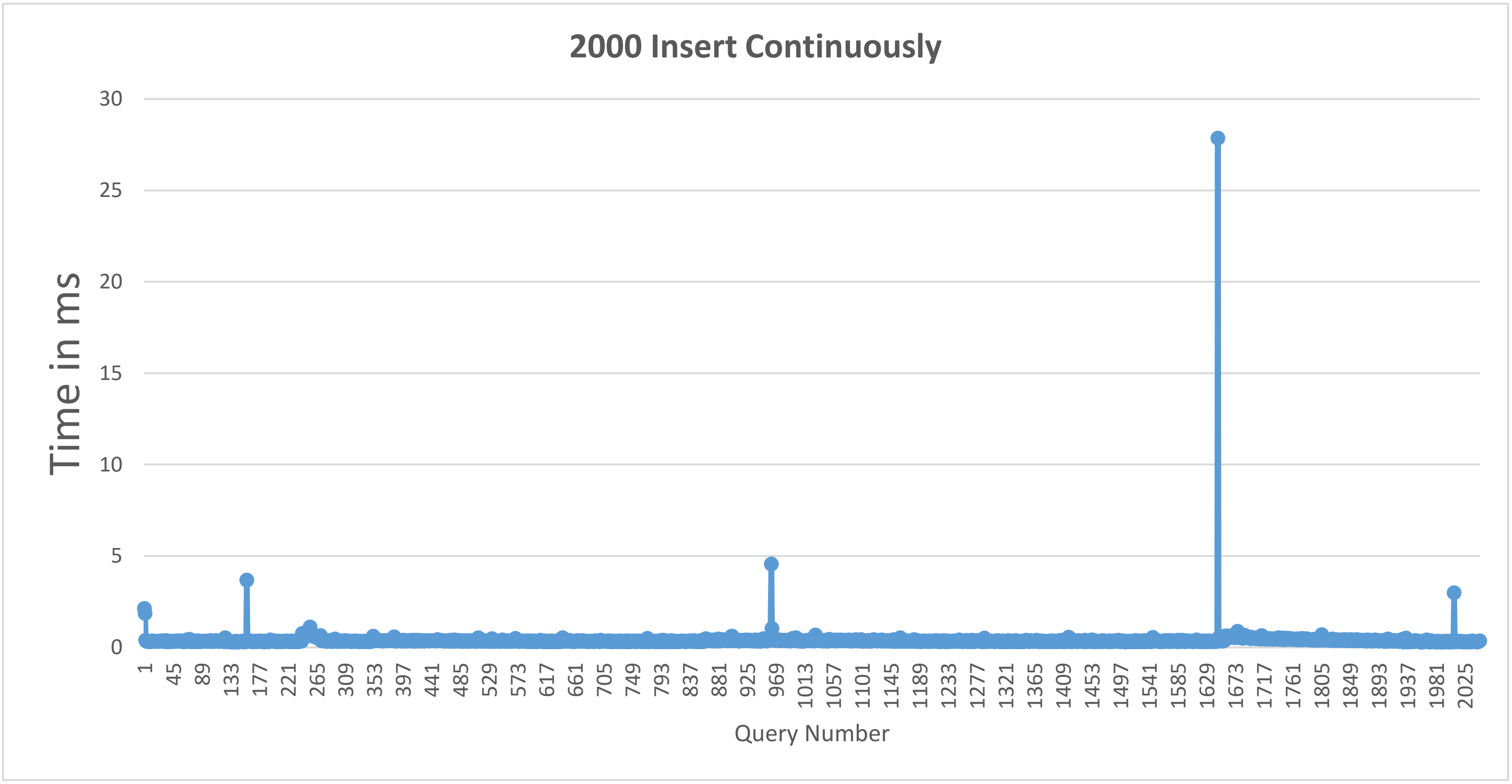
Consistency and packet loss are handled exactly the same way as for TCP.
Galera does not rely on TCP "reliability".
- show quoted text -

1 0

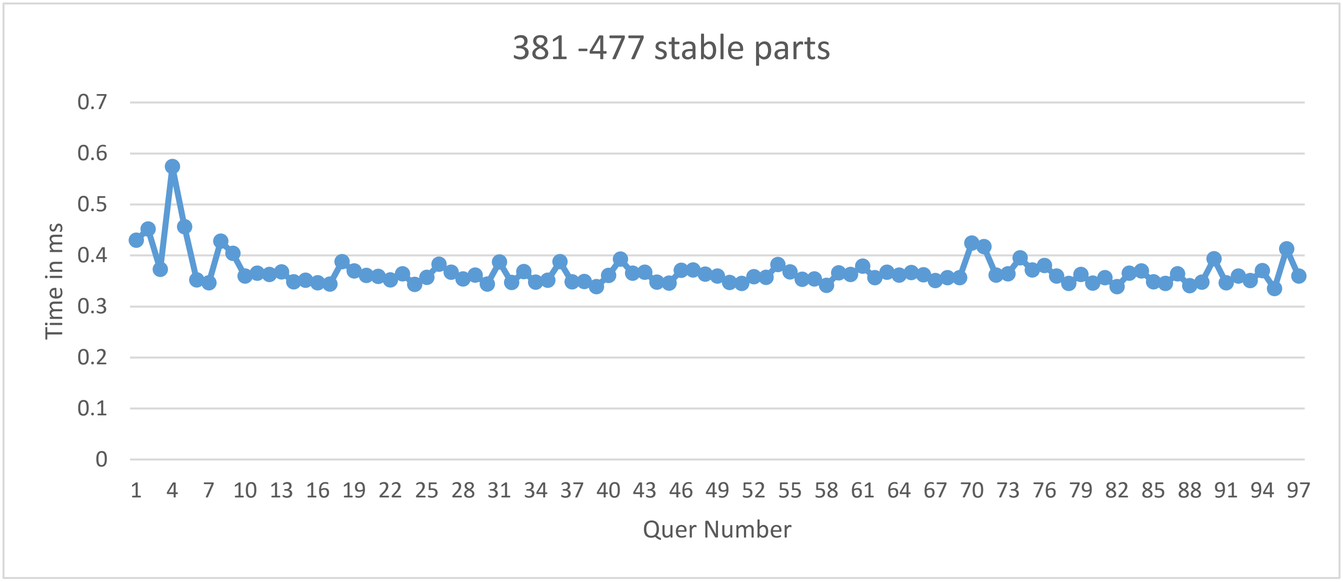
By the way we will find out about that very soon.

Here are some result of performance evaluation during the Insertion of Data.

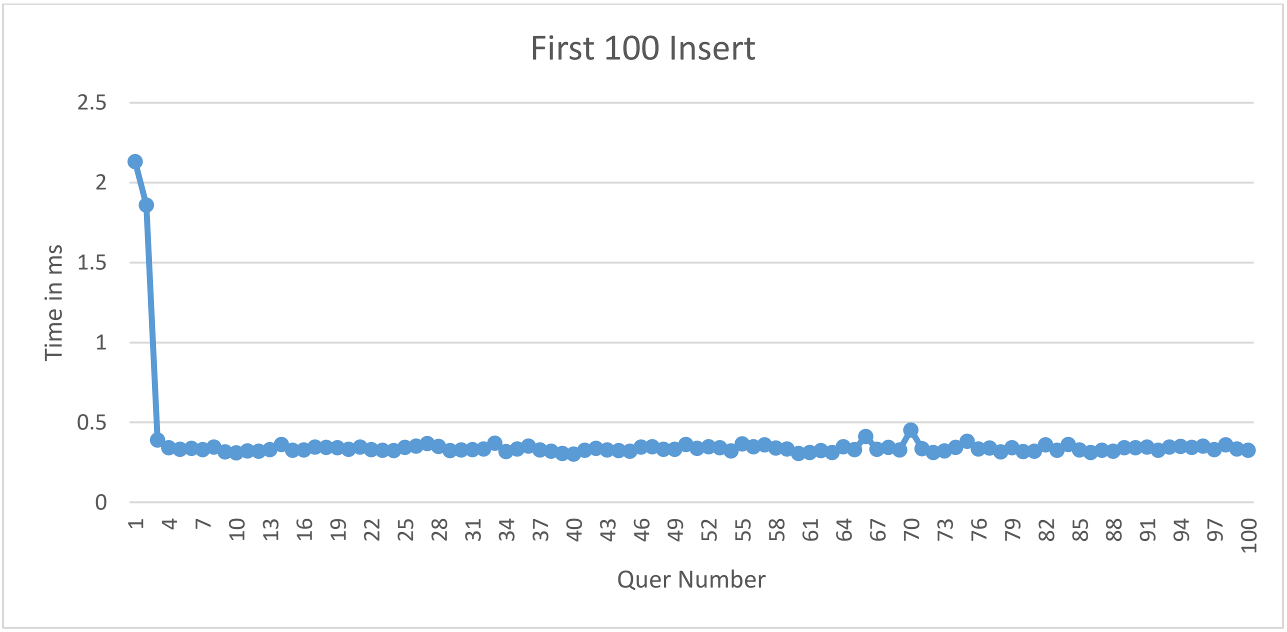
2000 Insert #1



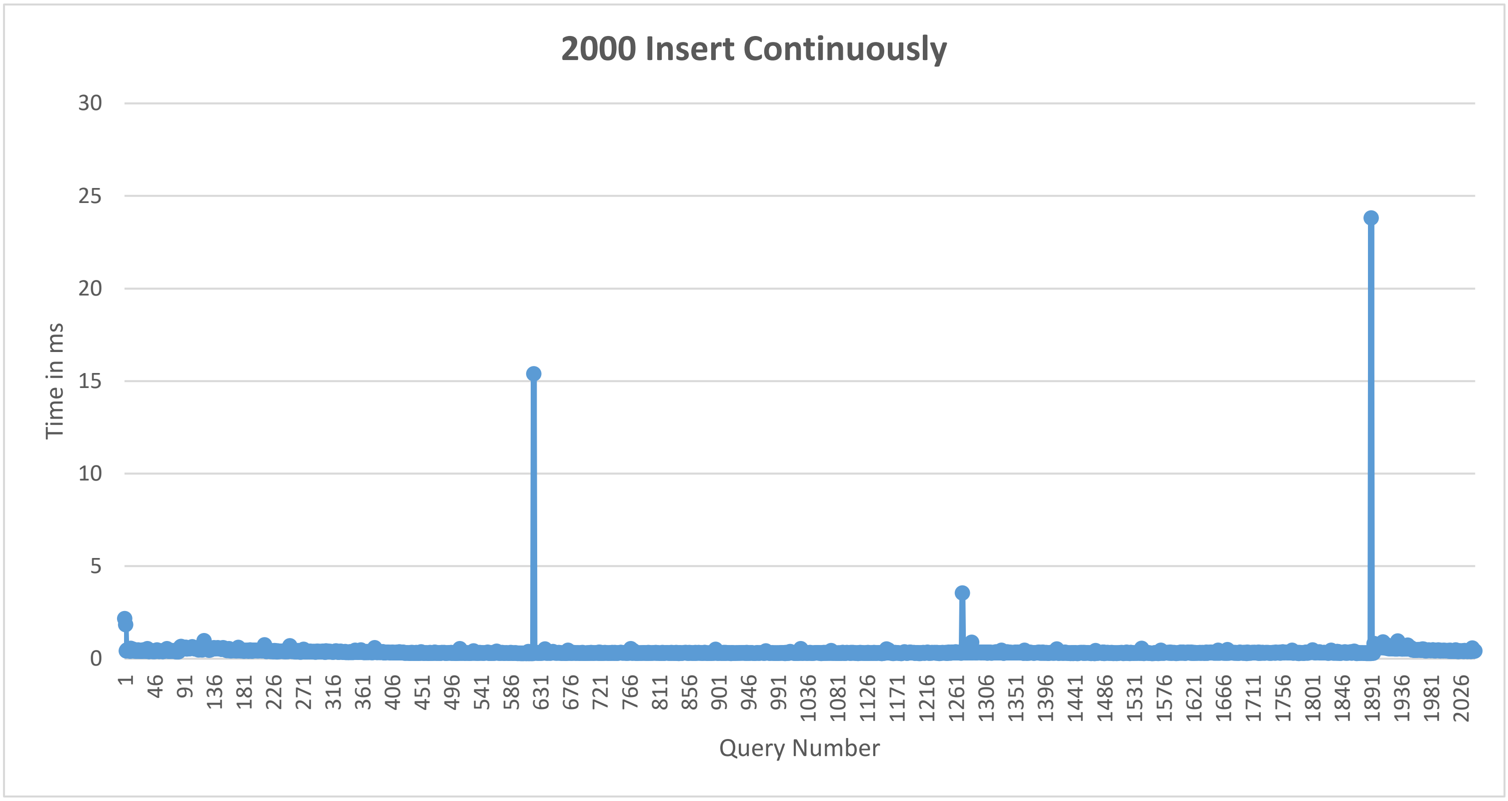
If we take closer look at sable part, we can see each Insertion took almost less than 0.5 ms.



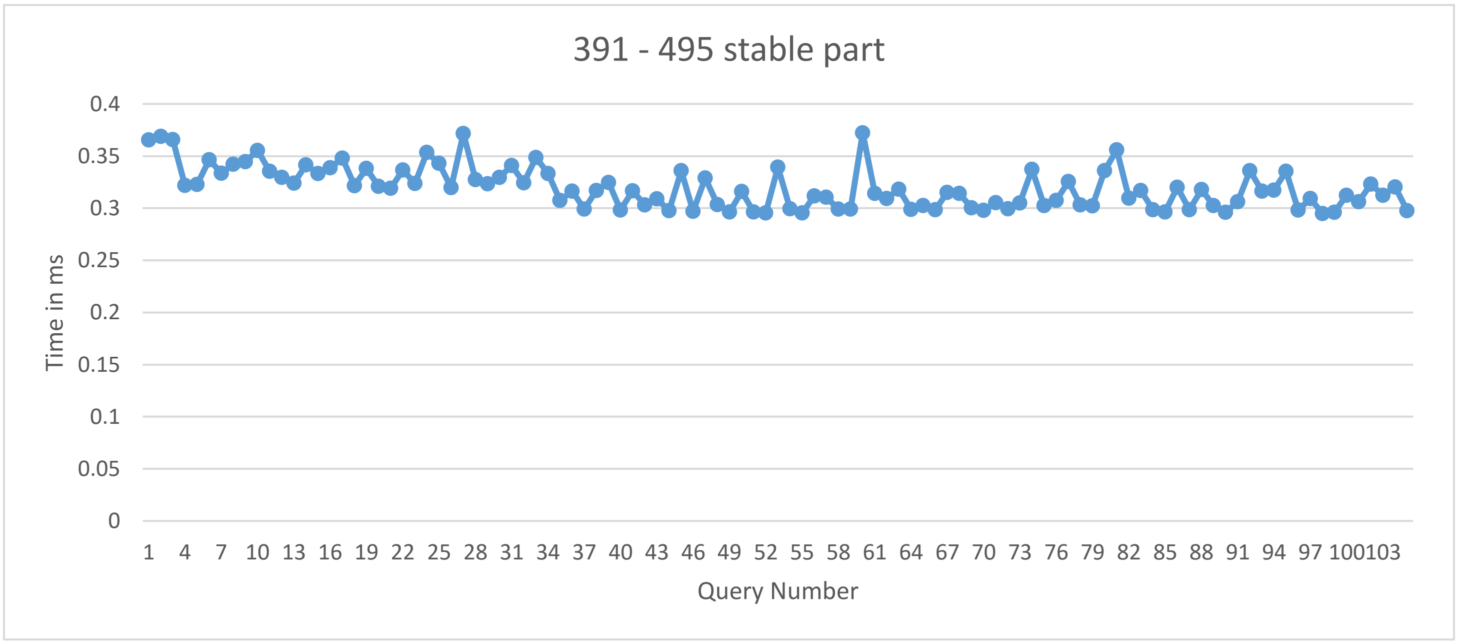
If we take closer look at to First 100 Insertion, we can see each Insertion took almost less than 0.5 ms BUT the very two first took more time.



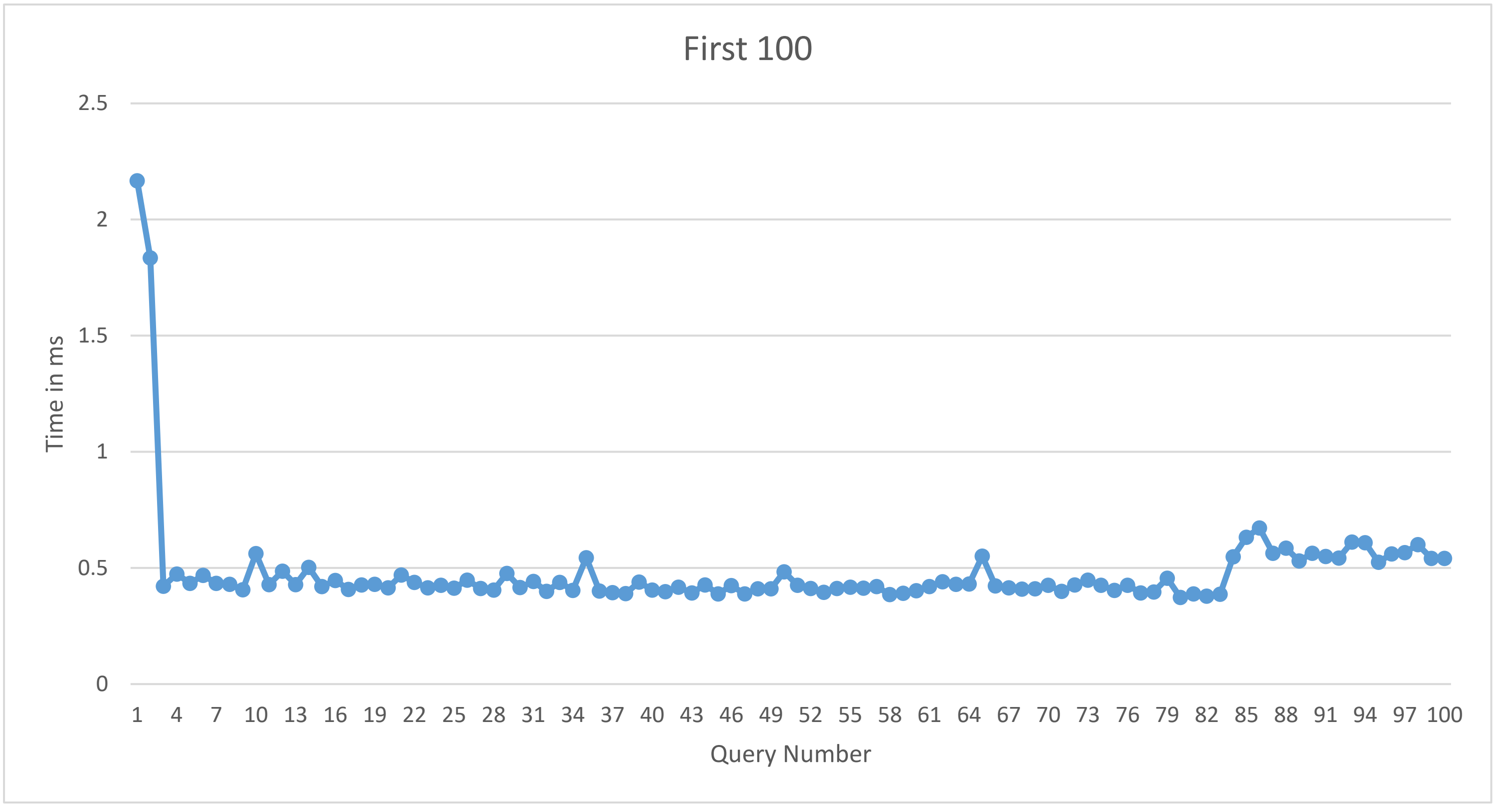
2000 Insert #2



Stable Part closer look



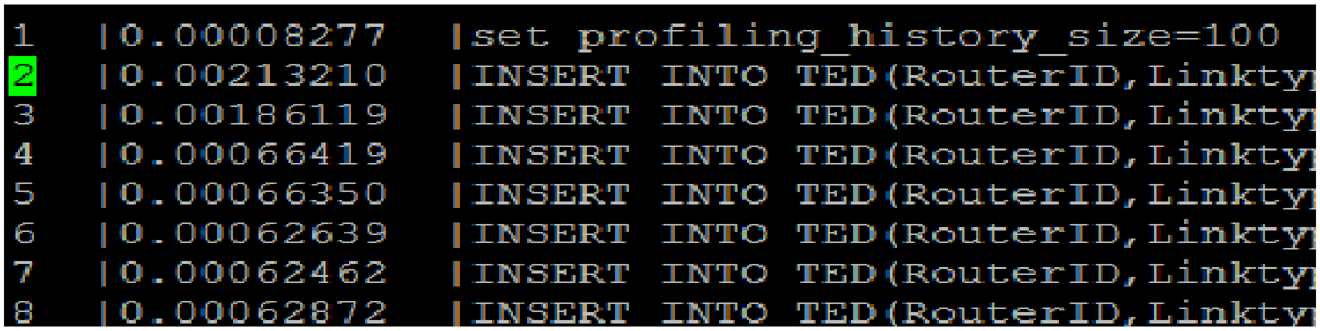
First 100 Insertion time



As you can see, there is some unavoidable time which is larger than normal time

Like the very two first Insertion and sometimes in middle.

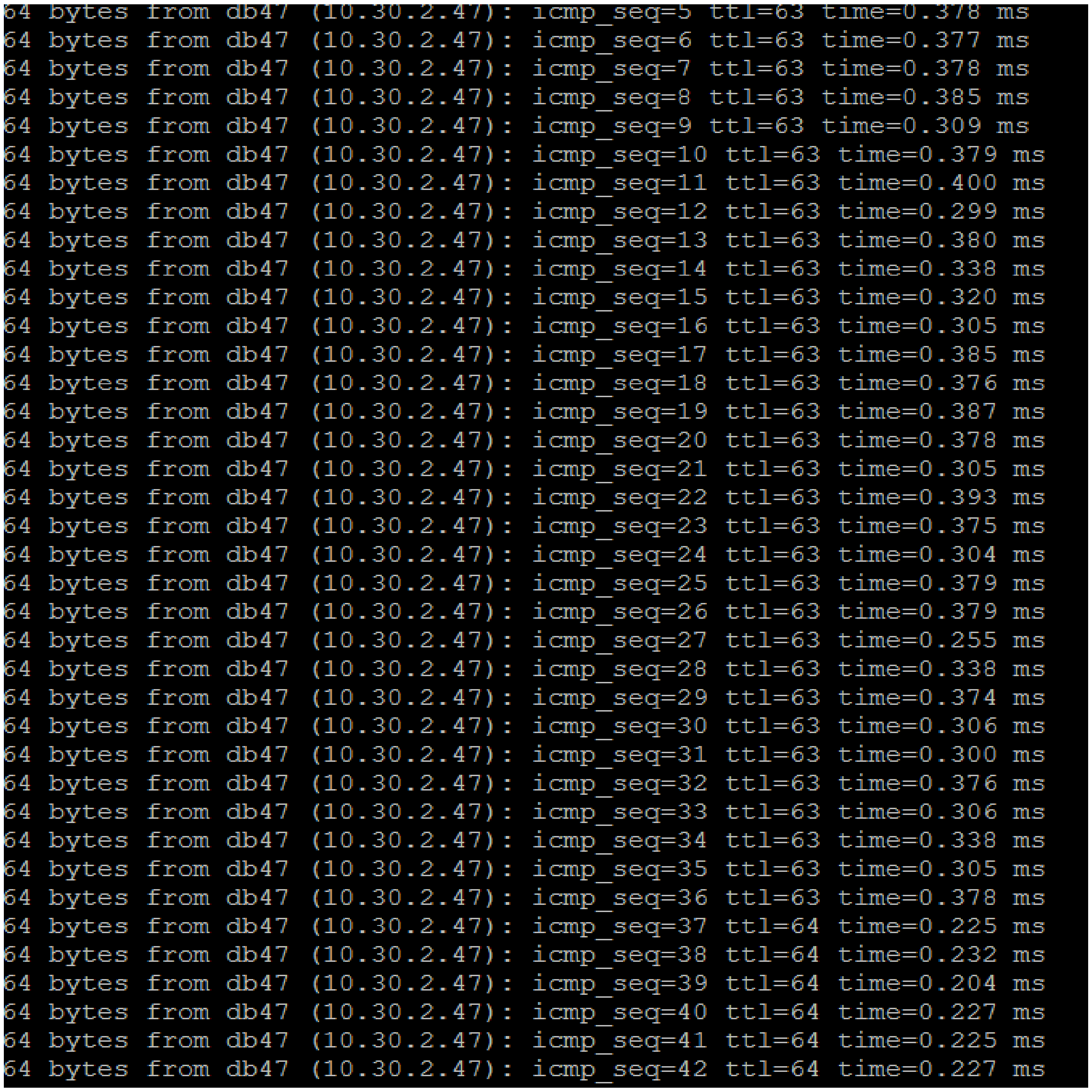
The very first two Insertion time



```
1    |0.00008277   |set profiling_history_size=100
2    |0.00213210   |INSERT INTO TED(RouterID,Linkty
3    |0.00186119   |INSERT INTO TED(RouterID,Linkty
4    |0.00066419   |INSERT INTO TED(RouterID,Linkty
5    |0.00066350   |INSERT INTO TED(RouterID,Linkty
6    |0.00062639   |INSERT INTO TED(RouterID,Linkty
7    |0.00062462   |INSERT INTO TED(RouterID,Linkty
8    |0.00062872   |INSERT INTO TED(RouterID,Linkty
```

Small fluctuations in graph which are less than 0.2 ms are due to network RTT variation which is normal. We know that for each query we need at least one RTT. (We can execute different queries in one RTT if and only if they happen at exactly same time and they work on different ROWs in the table).

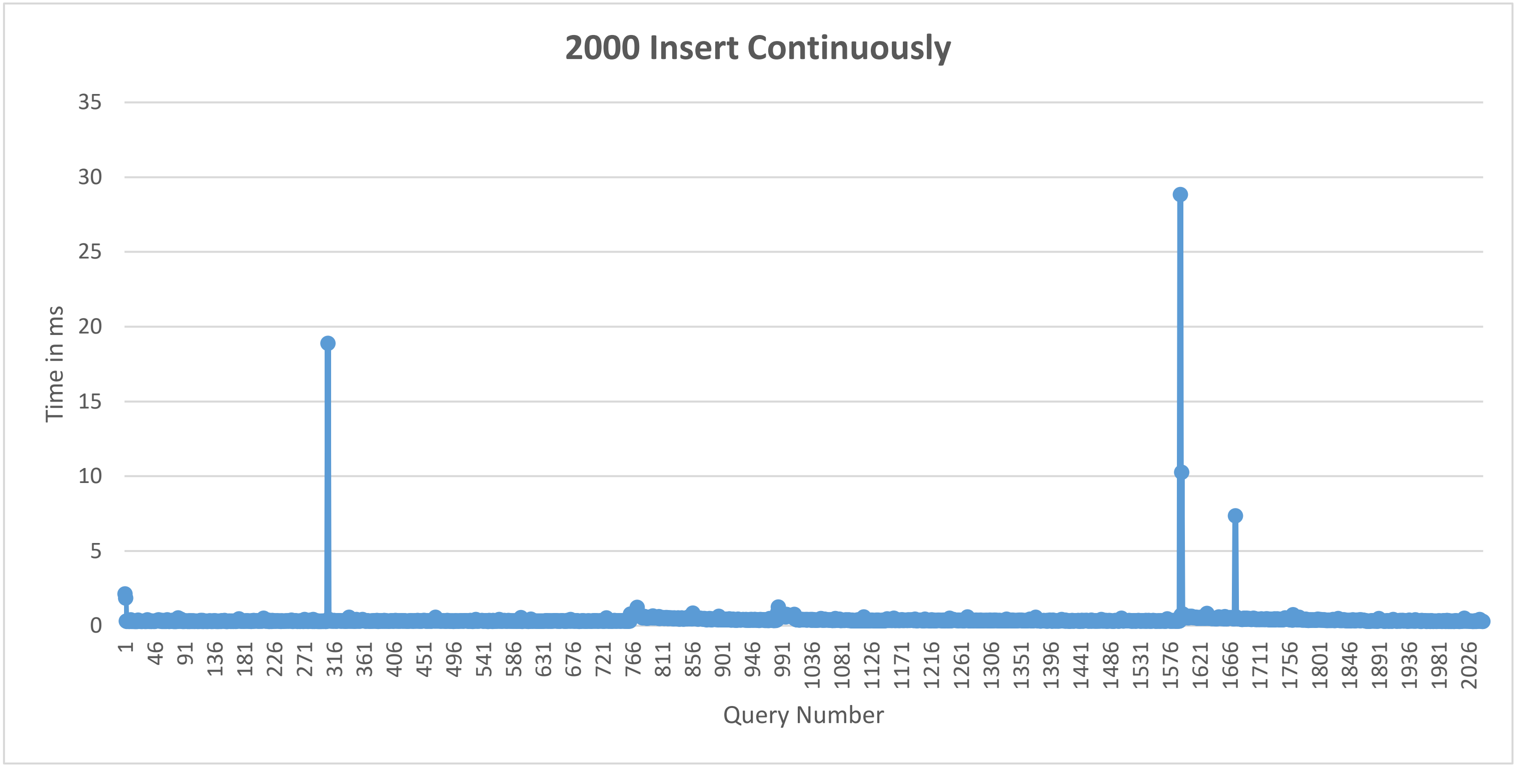
RTT between two nodes:

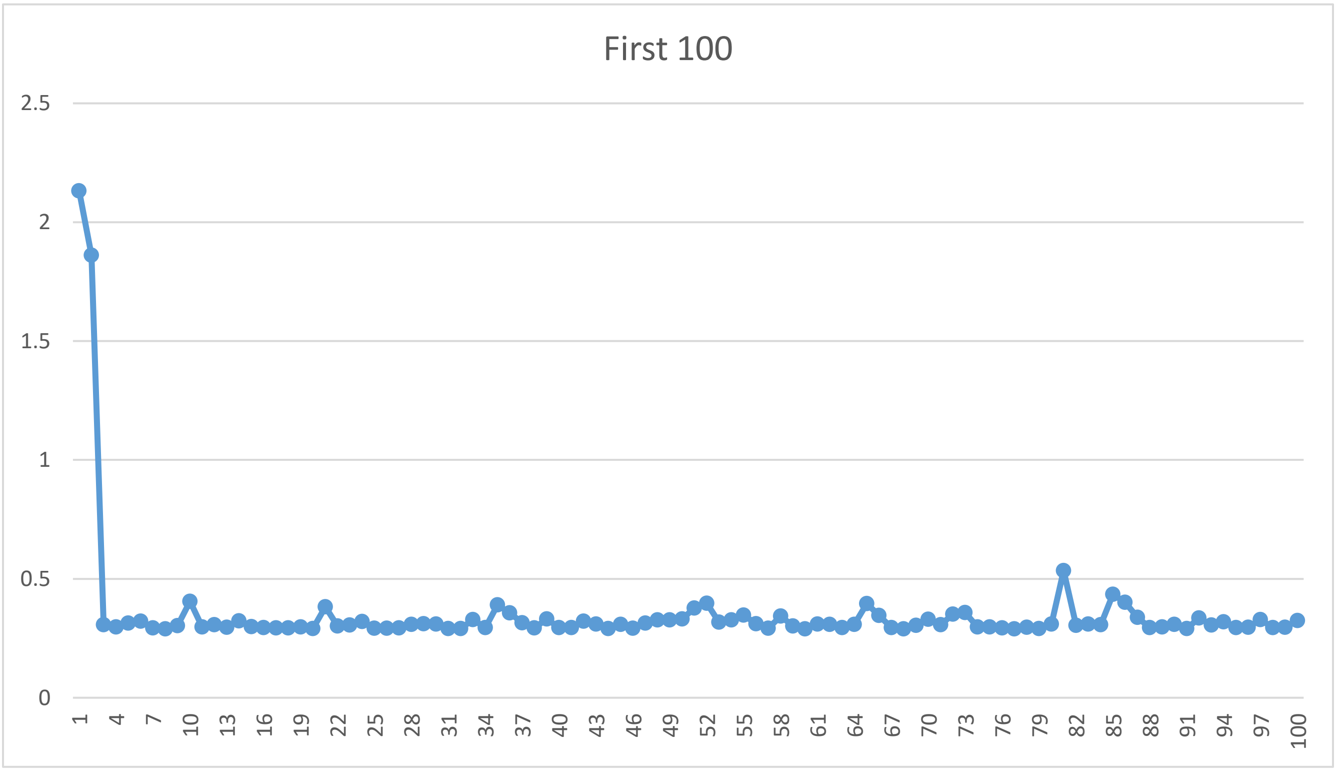


```
64 bytes from db47 (10.30.2.47): icmp_seq=5 ttl=63 time=0.378 ms
64 bytes from db47 (10.30.2.47): icmp_seq=6 ttl=63 time=0.377 ms
64 bytes from db47 (10.30.2.47): icmp_seq=7 ttl=63 time=0.378 ms
64 bytes from db47 (10.30.2.47): icmp_seq=8 ttl=63 time=0.385 ms
64 bytes from db47 (10.30.2.47): icmp_seq=9 ttl=63 time=0.309 ms
64 bytes from db47 (10.30.2.47): icmp_seq=10 ttl=63 time=0.379 ms
64 bytes from db47 (10.30.2.47): icmp_seq=11 ttl=63 time=0.400 ms
64 bytes from db47 (10.30.2.47): icmp_seq=12 ttl=63 time=0.299 ms
64 bytes from db47 (10.30.2.47): icmp_seq=13 ttl=63 time=0.380 ms
64 bytes from db47 (10.30.2.47): icmp_seq=14 ttl=63 time=0.338 ms
64 bytes from db47 (10.30.2.47): icmp_seq=15 ttl=63 time=0.320 ms
64 bytes from db47 (10.30.2.47): icmp_seq=16 ttl=63 time=0.305 ms
64 bytes from db47 (10.30.2.47): icmp_seq=17 ttl=63 time=0.385 ms
64 bytes from db47 (10.30.2.47): icmp_seq=18 ttl=63 time=0.376 ms
64 bytes from db47 (10.30.2.47): icmp_seq=19 ttl=63 time=0.387 ms
64 bytes from db47 (10.30.2.47): icmp_seq=20 ttl=63 time=0.378 ms
64 bytes from db47 (10.30.2.47): icmp_seq=21 ttl=63 time=0.305 ms
64 bytes from db47 (10.30.2.47): icmp_seq=22 ttl=63 time=0.393 ms
64 bytes from db47 (10.30.2.47): icmp_seq=23 ttl=63 time=0.375 ms
64 bytes from db47 (10.30.2.47): icmp_seq=24 ttl=63 time=0.304 ms
64 bytes from db47 (10.30.2.47): icmp_seq=25 ttl=63 time=0.379 ms
64 bytes from db47 (10.30.2.47): icmp_seq=26 ttl=63 time=0.379 ms
64 bytes from db47 (10.30.2.47): icmp_seq=27 ttl=63 time=0.255 ms
64 bytes from db47 (10.30.2.47): icmp_seq=28 ttl=63 time=0.338 ms
64 bytes from db47 (10.30.2.47): icmp_seq=29 ttl=63 time=0.374 ms
64 bytes from db47 (10.30.2.47): icmp_seq=30 ttl=63 time=0.306 ms
64 bytes from db47 (10.30.2.47): icmp_seq=31 ttl=63 time=0.300 ms
64 bytes from db47 (10.30.2.47): icmp_seq=32 ttl=63 time=0.376 ms
64 bytes from db47 (10.30.2.47): icmp_seq=33 ttl=63 time=0.306 ms
64 bytes from db47 (10.30.2.47): icmp_seq=34 ttl=63 time=0.338 ms
64 bytes from db47 (10.30.2.47): icmp_seq=35 ttl=63 time=0.305 ms
64 bytes from db47 (10.30.2.47): icmp_seq=36 ttl=63 time=0.378 ms
64 bytes from db47 (10.30.2.47): icmp_seq=37 ttl=64 time=0.225 ms
64 bytes from db47 (10.30.2.47): icmp_seq=38 ttl=64 time=0.232 ms
64 bytes from db47 (10.30.2.47): icmp_seq=39 ttl=64 time=0.204 ms
64 bytes from db47 (10.30.2.47): icmp_seq=40 ttl=64 time=0.227 ms
64 bytes from db47 (10.30.2.47): icmp_seq=41 ttl=64 time=0.225 ms
64 bytes from db47 (10.30.2.47): icmp_seq=42 ttl=64 time=0.227 ms
```

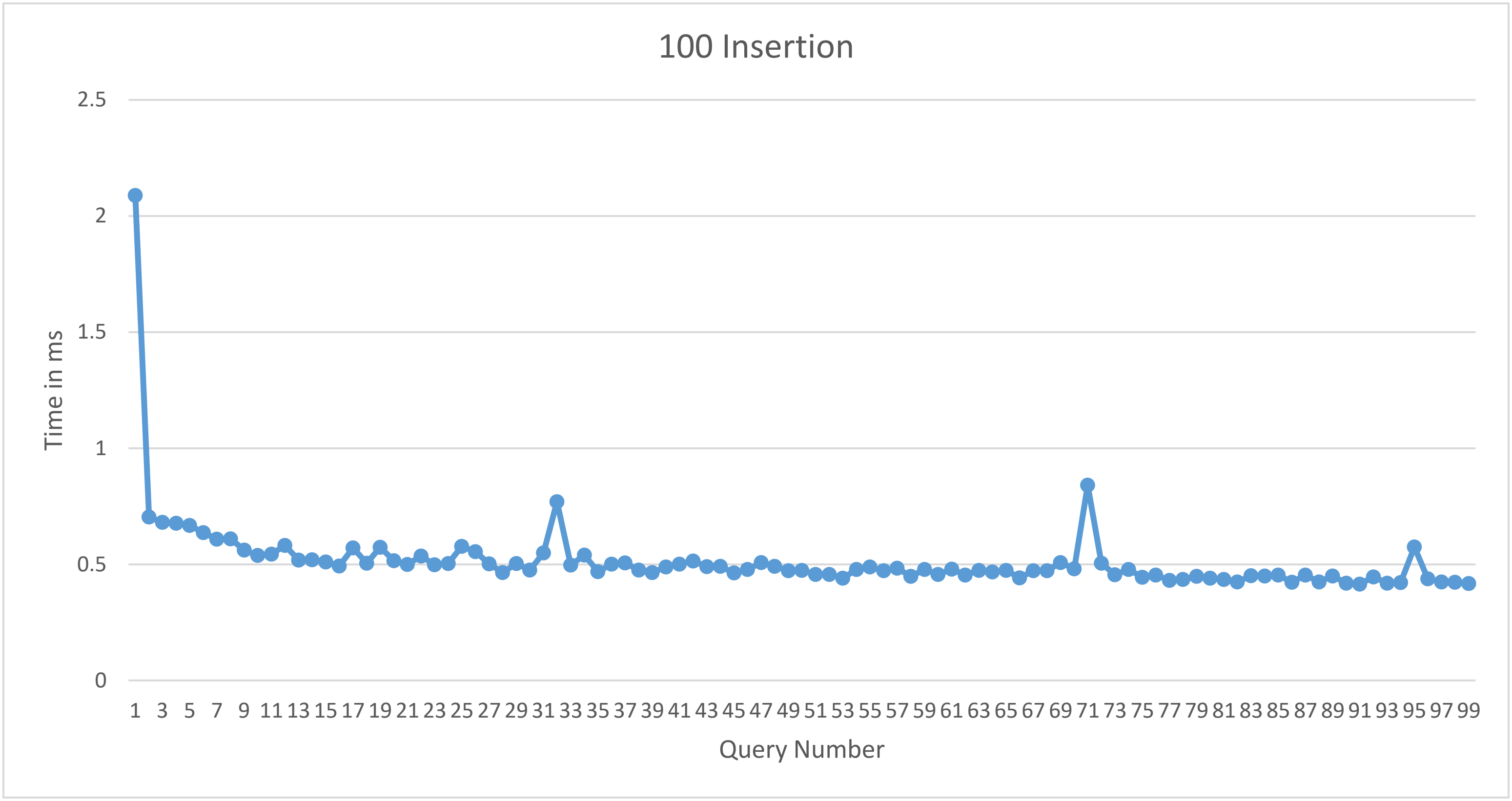
As you can see RTT variation between two nodes is from 0.225 ms to 400 ms.

2000 Insert #3





100 Insert #4



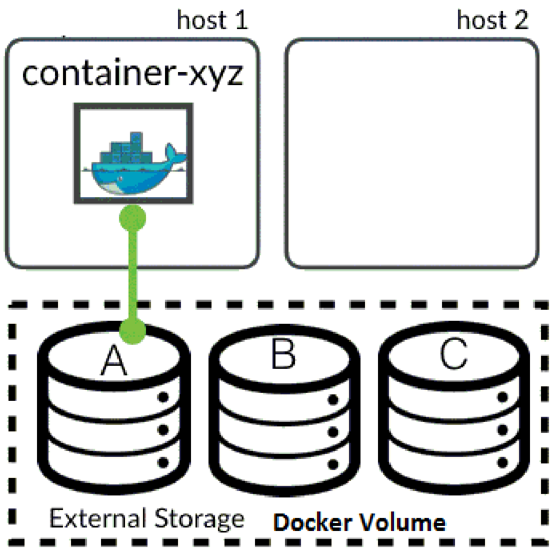
However, in our scenario, we are not going to insert that amount of data continuously except for recovery after failure by IST (Incremental State Transfer) And to avoid that I have some interesting idea of implementation by Docker Container with Auto Redeploy on Failure and one more extra node at same Domain with shared Docker data Volume.

The idea is that, in each Domain instead of deploying 1 Database we deploy 2 Database but both databases shares same Data volume.

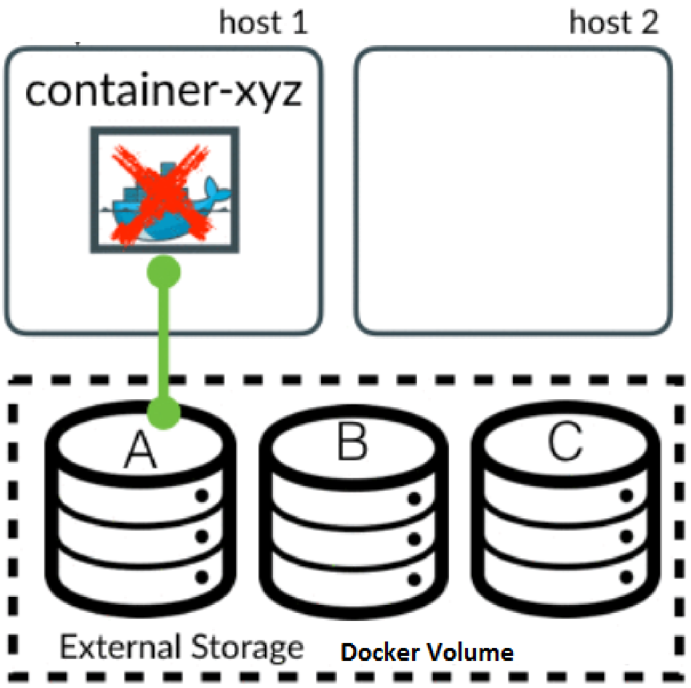
If one of our node which runs MariaDB Galera crashes Docker automatically will connect the data volume to other node and the failed node will automatically redeployed to fix its issues and while it's on recovery the other node takes the responsibility. Since in Galera we have for each update a Global Transaction ID

and by each change in Database the Database ID will increase, no transaction will be lost. Reference:
https://clusterhq.com/2015/12/09/difference-docker-volumes-flocker-volumes/

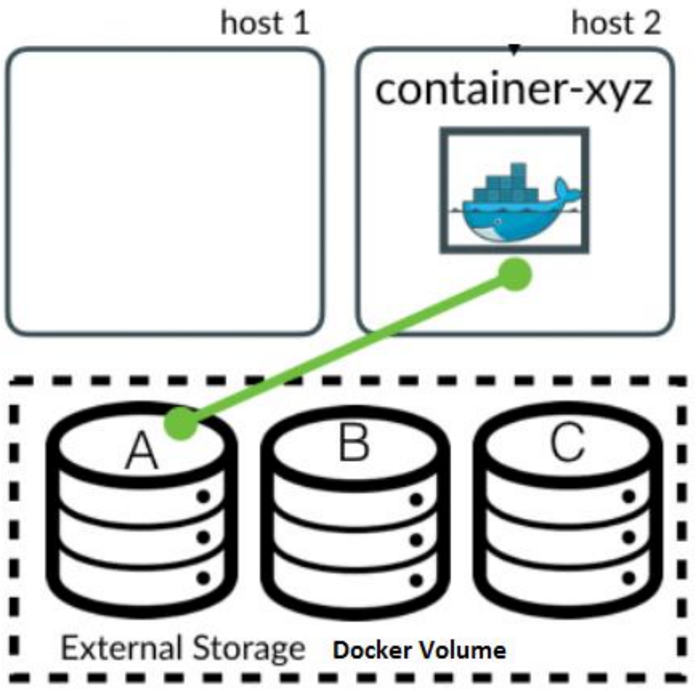
Host 1 works fine and connected to Data Volume



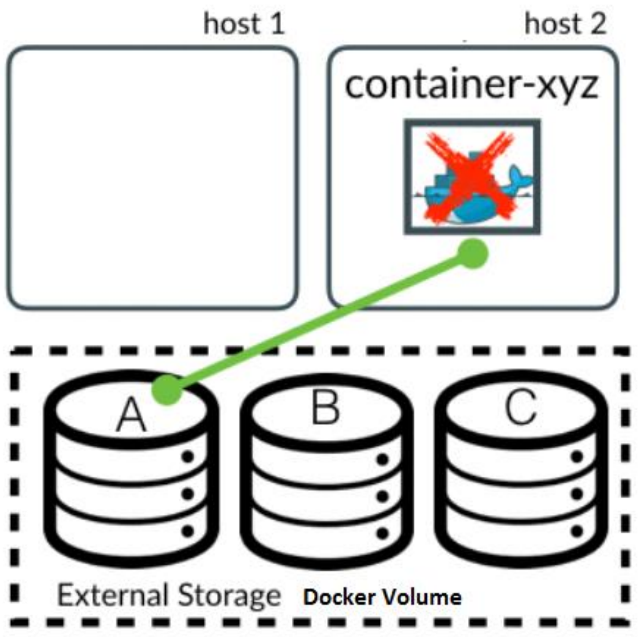
In case of Failure on Host 1



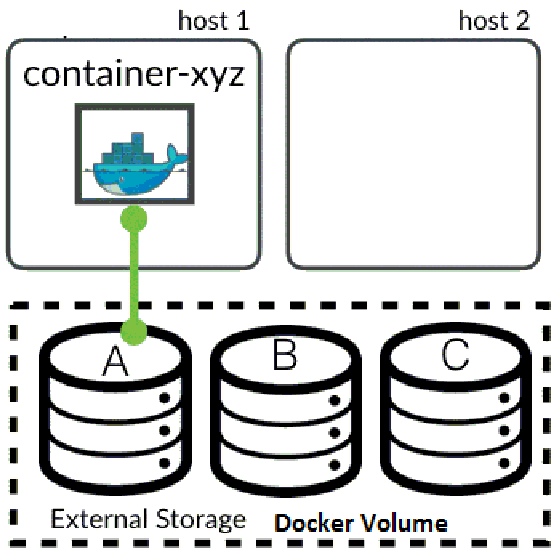
Automatically give responsibility to Host 2



And Host 1 will go to recovery state to be ready for back up In case of Failure in Host 2



Responsibility will come back to Host 1



It’s a simple scenario to have redundancy and have 24/7 back up and auto recovery for our Databases. Also by using Docker we will benefit from monitoring and many other features which nowadays become popular.

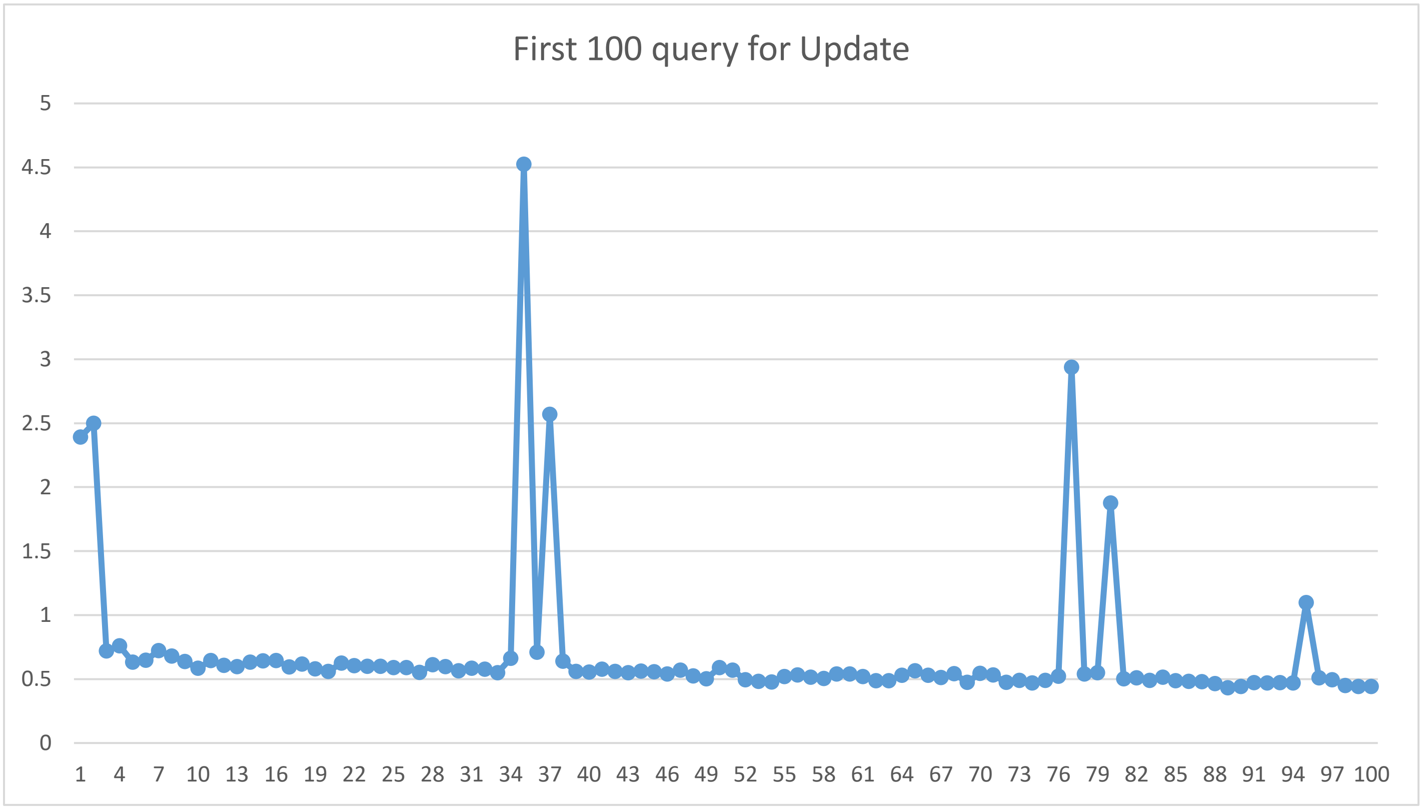
Update Performance Evaluation

For updating one Row in TED table, I Update 9 different element in each update query. The fields are such as **Maximum Reserve Bandwidth**, and **Unreserved Bandwidth** with 8 different priority.

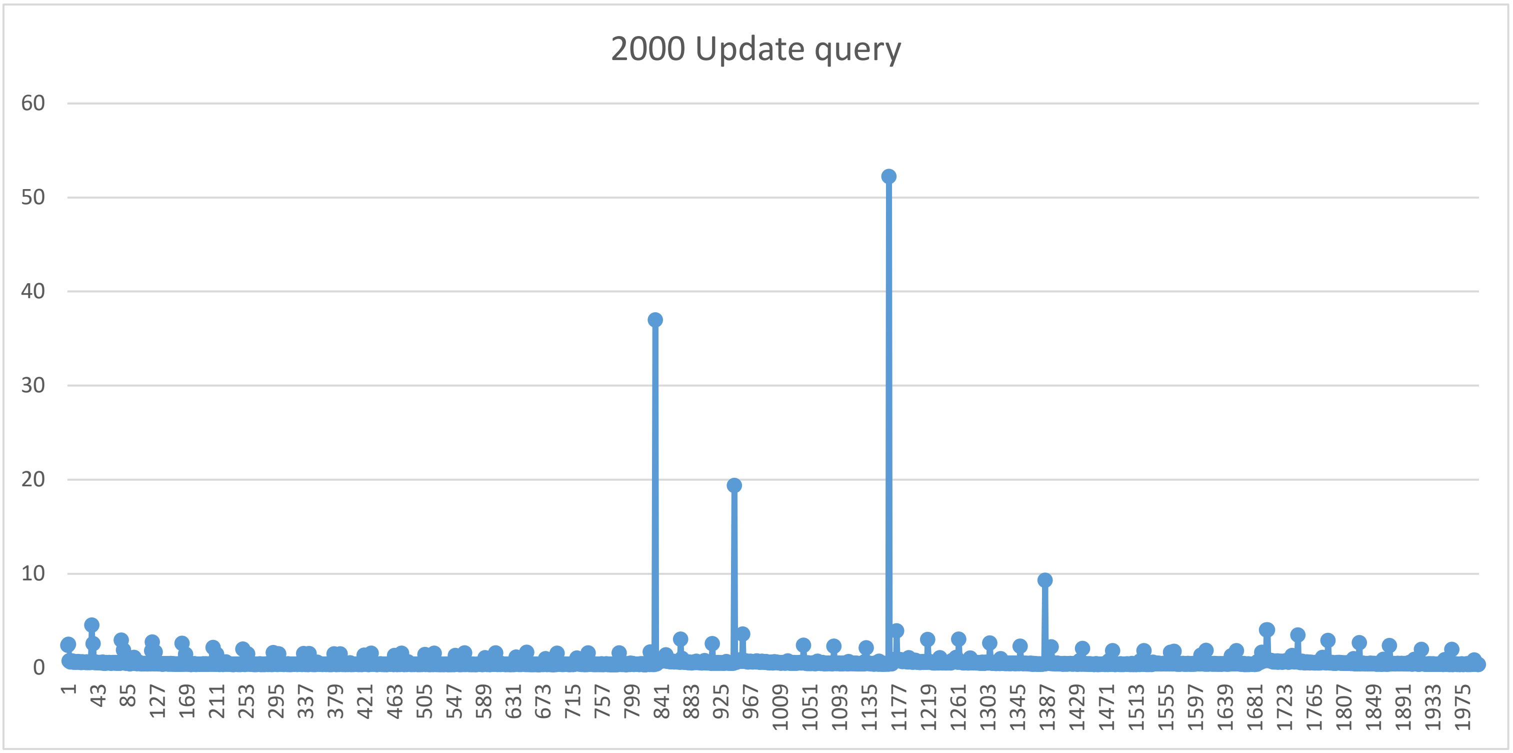
Normally the result is around 0.7 ms for each update, of course update need to fetch the data from database to cache and then update it and then propagate the update to all cluster and after getting commit it stamps the duration time.

Fluctuation in Update graph is more than others because there is many interaction between Database and cache.

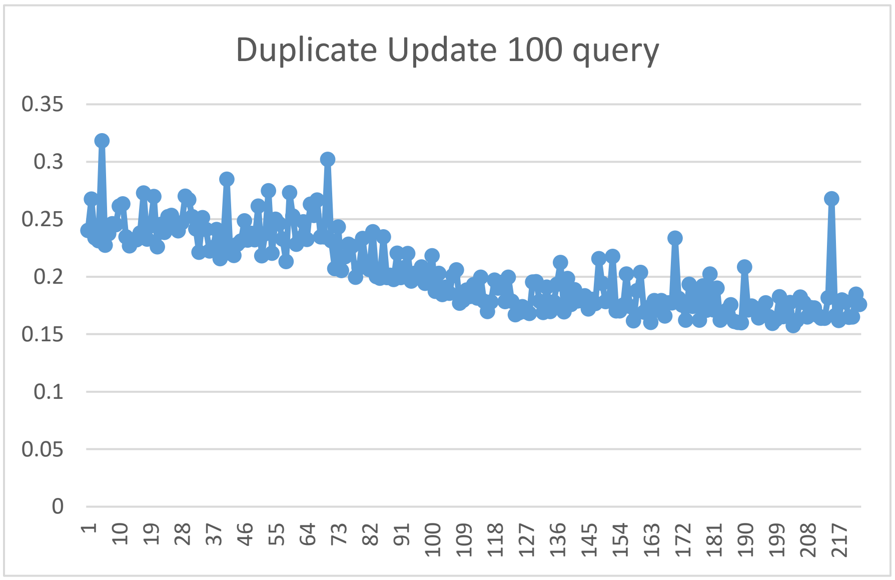
100 Update #1



2000 Update #1

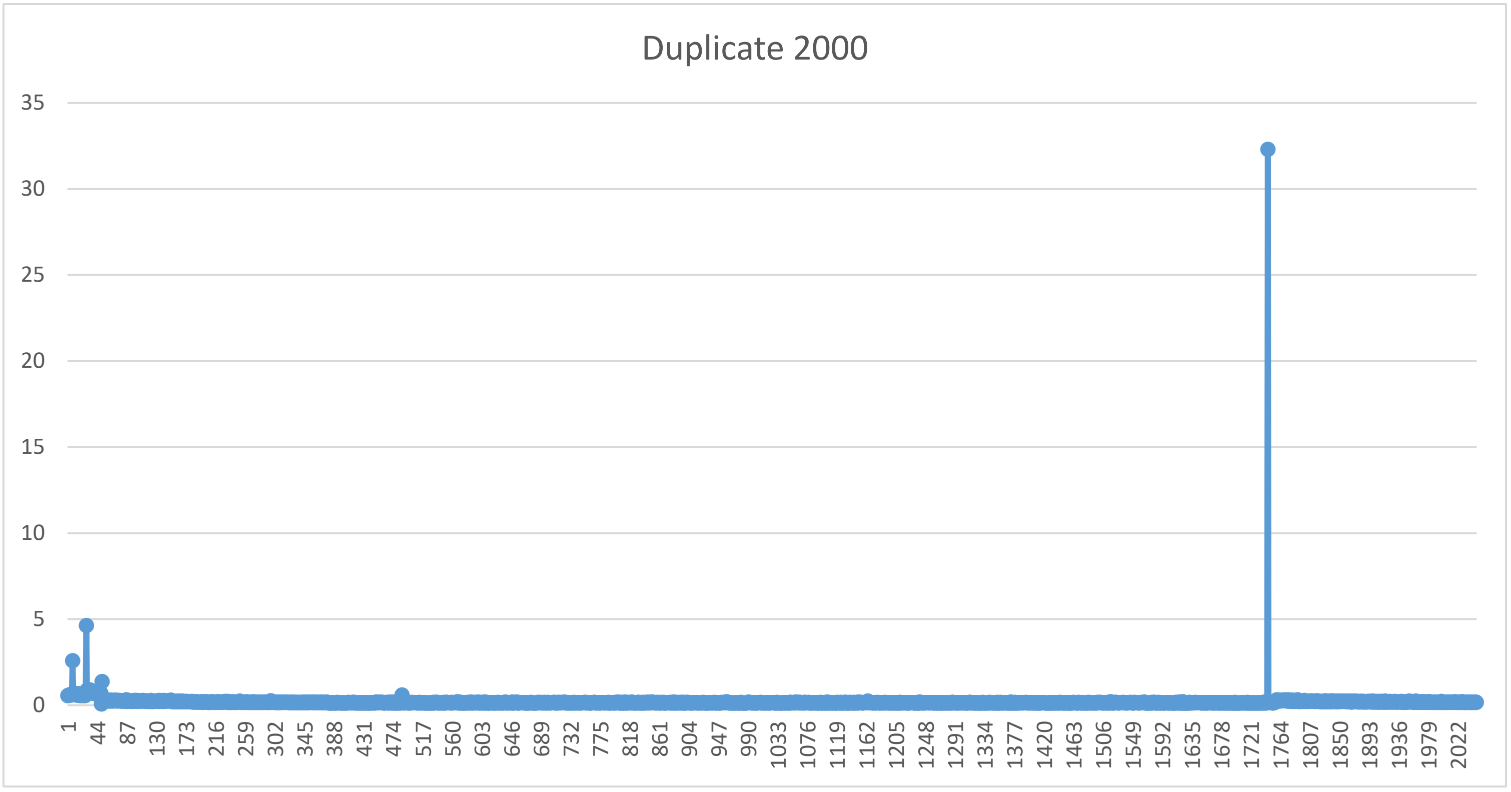


Also I tried to understand if I execute update queries which are not changing any data, how much time it going to take. For understanding that I execute the same code without changing any data and the result was interesting because there will not be any replication for Updating with no change.

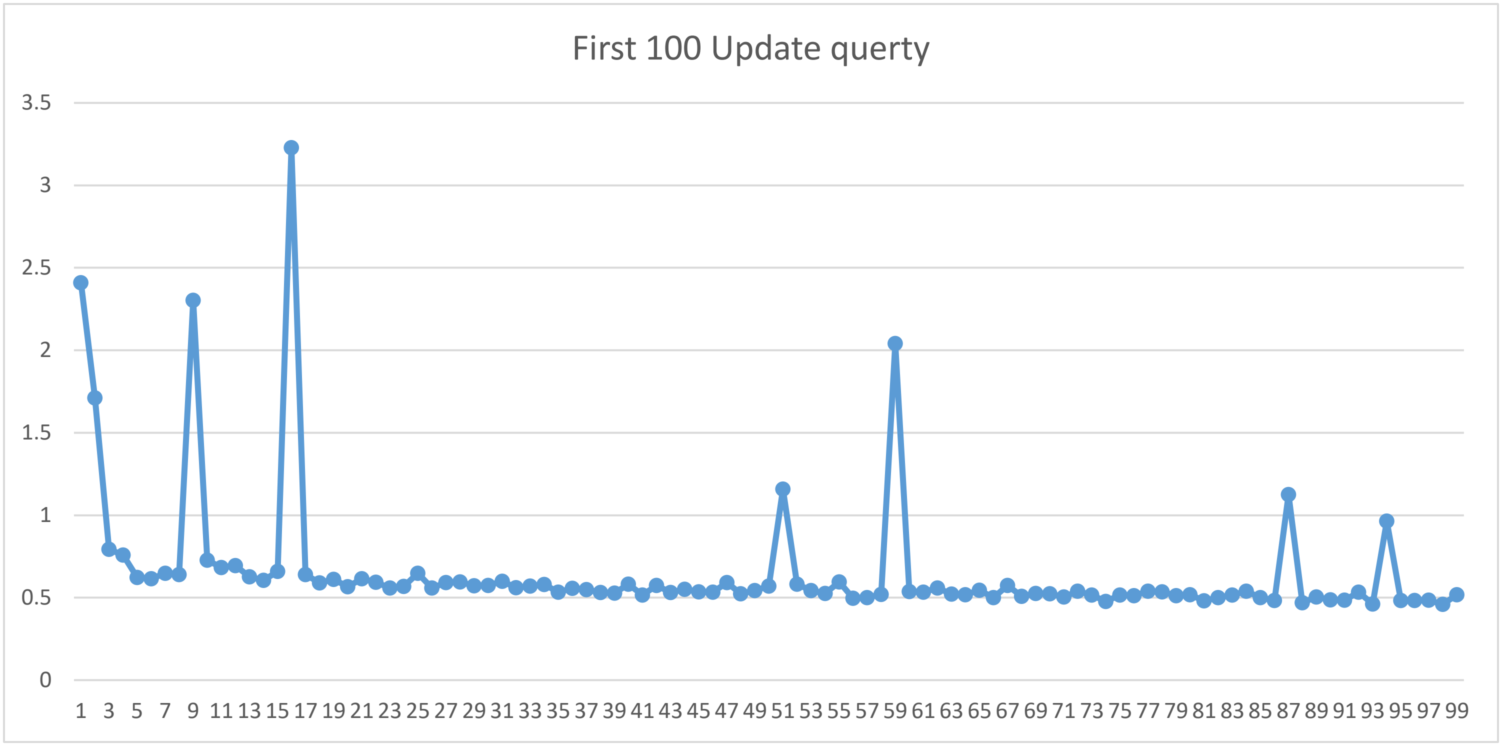


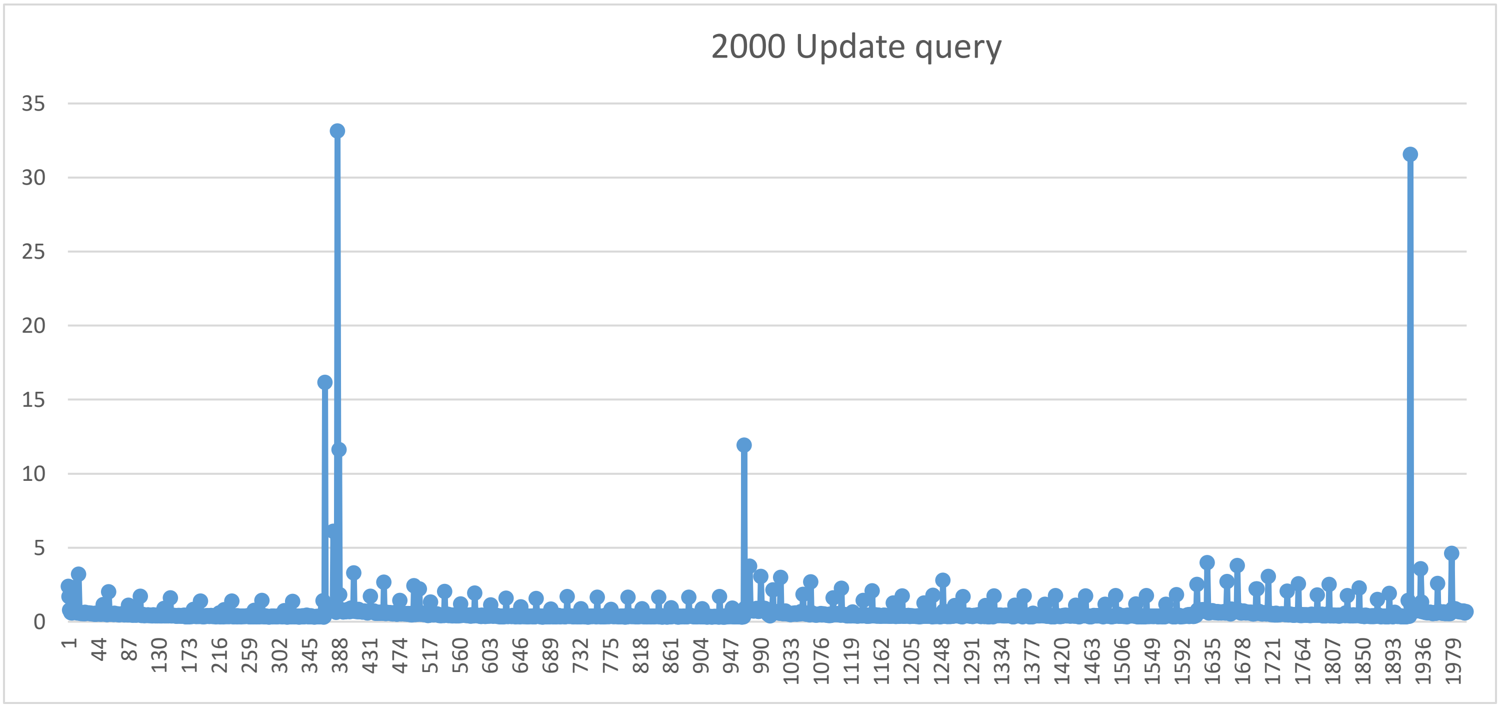
The time is almost a RTT time, so there is no replication for it.

And if we want to take look at over all result for 2000 duplicated Update:



100 Update #2


First 100 Update querty
3.5
3
2.5
2
1.5
1
0.5
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99


2000 Update query
35
30
25
20
15
10
5
0
1 44 87 130 173 216 259 302 345 388 431 474 517 560 603 646 689 732 775 818 861 904 947 990 1033 1076 1119 1162 1205 1248 1291 1334 1377 1420 1463 1506 1549 1592 1635 1678 1721 1764 1807 1850 1893 1936 1979

Performance Evaluation on WAN

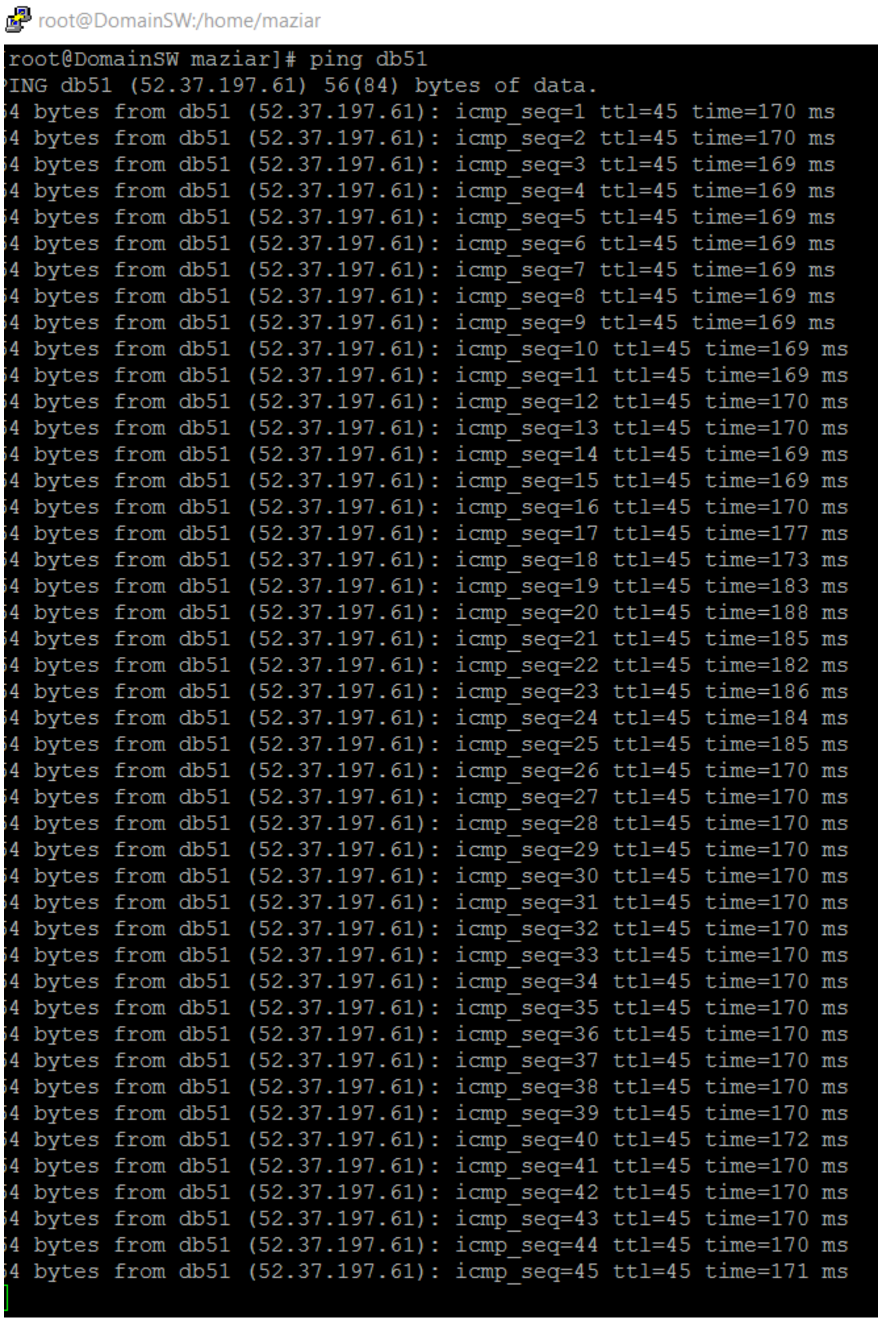
I succeed to deploy 2 node on the cloud,

First node on Amazon EC2 cloud in North America (US), Los Angeles Boardman

Second Node: on Cloudsigma Ag cloud in Europe (EU), Switzerland, Zurich

I describe the same table with name TED2 under database Domain_2

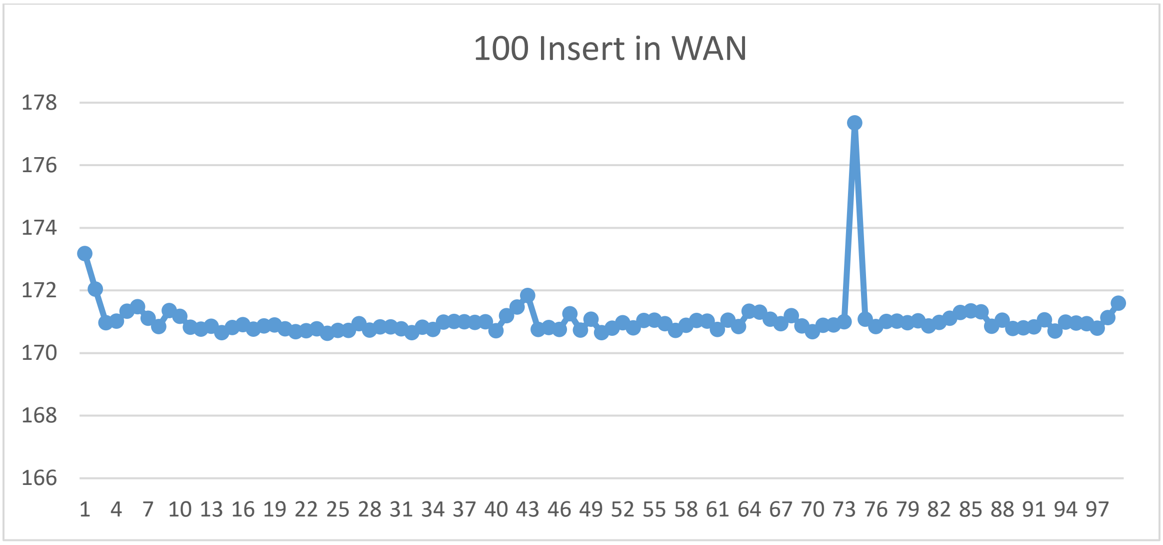
RTT between two nodes is around 170 ms (169 - 185)



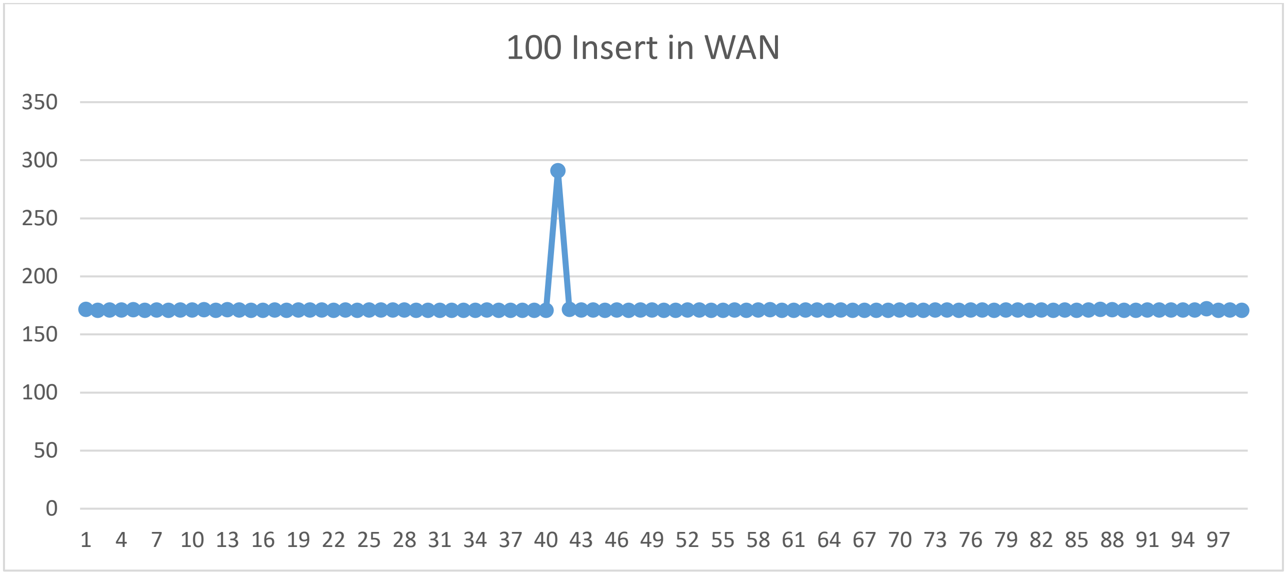
```
root@DomainSW:/home/maziar
[root@DomainSW maziar]# ping db51
PING db51 (52.37.197.61) 56(84) bytes of data.
64 bytes from db51 (52.37.197.61): icmp_seq=1 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=2 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=3 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=4 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=5 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=6 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=7 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=8 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=9 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=10 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=11 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=12 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=13 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=14 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=15 ttl=45 time=169 ms
64 bytes from db51 (52.37.197.61): icmp_seq=16 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=17 ttl=45 time=177 ms
64 bytes from db51 (52.37.197.61): icmp_seq=18 ttl=45 time=173 ms
64 bytes from db51 (52.37.197.61): icmp_seq=19 ttl=45 time=183 ms
64 bytes from db51 (52.37.197.61): icmp_seq=20 ttl=45 time=188 ms
64 bytes from db51 (52.37.197.61): icmp_seq=21 ttl=45 time=185 ms
64 bytes from db51 (52.37.197.61): icmp_seq=22 ttl=45 time=182 ms
64 bytes from db51 (52.37.197.61): icmp_seq=23 ttl=45 time=186 ms
64 bytes from db51 (52.37.197.61): icmp_seq=24 ttl=45 time=184 ms
64 bytes from db51 (52.37.197.61): icmp_seq=25 ttl=45 time=185 ms
64 bytes from db51 (52.37.197.61): icmp_seq=26 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=27 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=28 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=29 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=30 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=31 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=32 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=33 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=34 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=35 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=36 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=37 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=38 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=39 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=40 ttl=45 time=172 ms
64 bytes from db51 (52.37.197.61): icmp_seq=41 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=42 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=43 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=44 ttl=45 time=170 ms
64 bytes from db51 (52.37.197.61): icmp_seq=45 ttl=45 time=171 ms
```

I took exactly same strategy in Insertion and updating described above

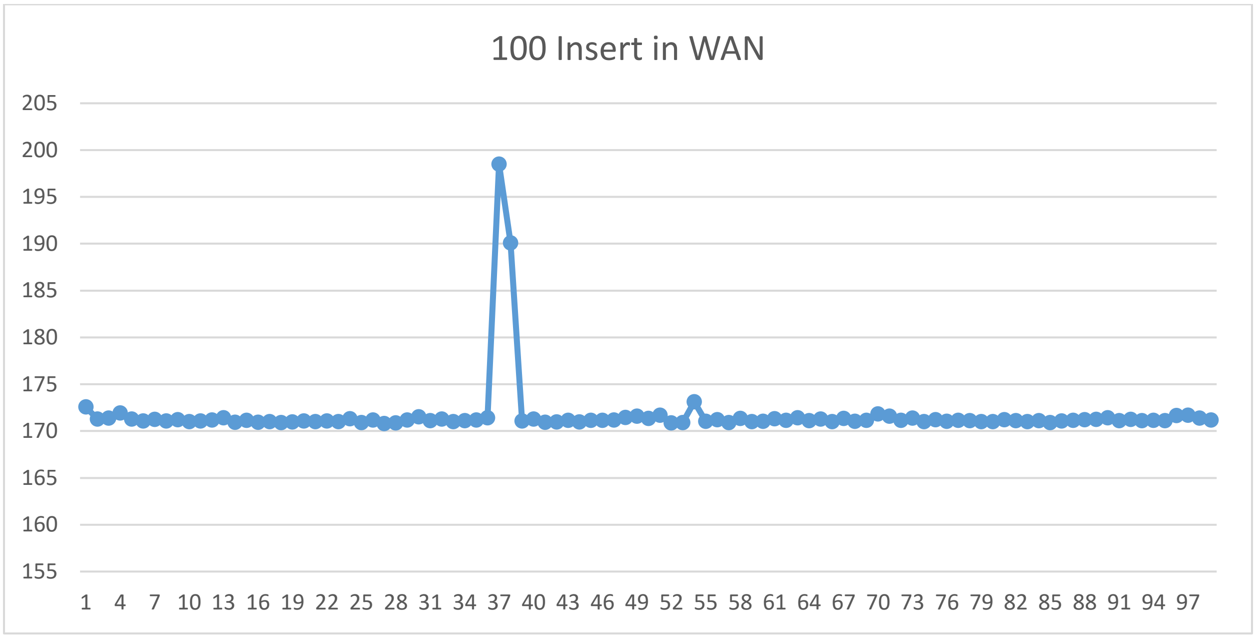
100 Insertion in WAN #1



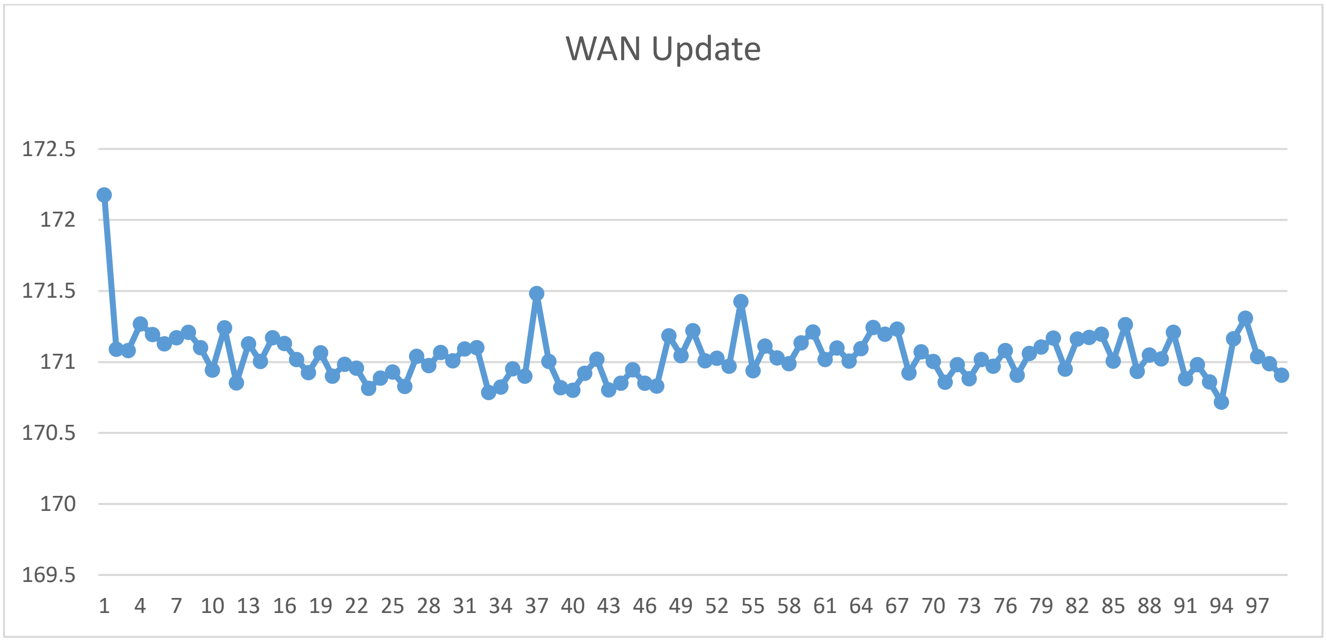
100 Insertion in WAN #2



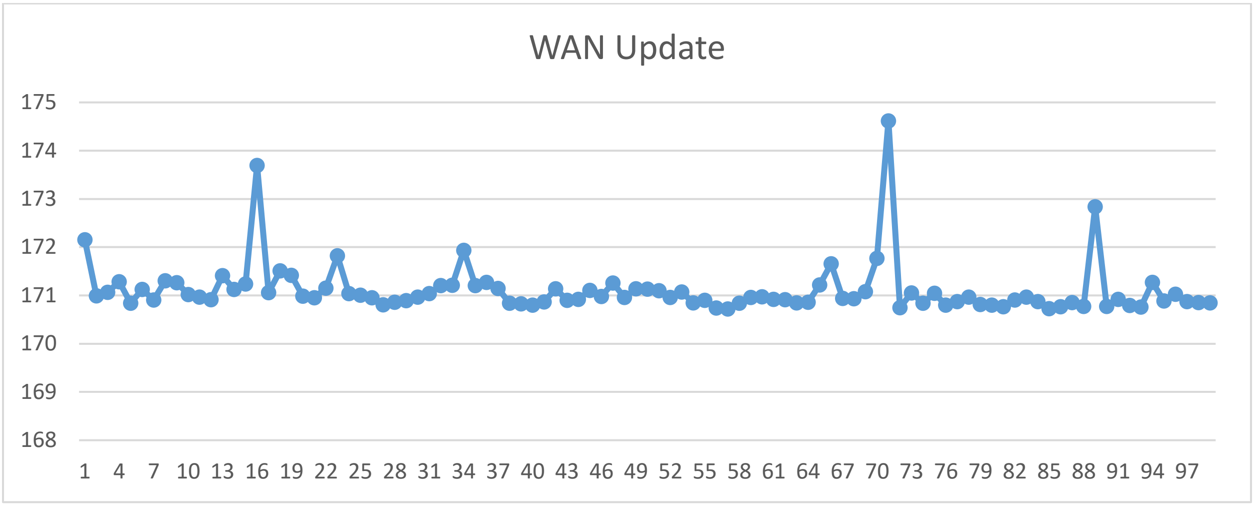
100 Insertion in WAN #3



100 Update in WAN #1



100 Update in WAN #2



As you can see, Galera gets done the replication in minimum amount of time and Inserting or updating continuously depands on many other factor such as parallelization degree (depands on number of CPU core), Size of InnoDB cache, and most important RTT. We will find out soon about using UDP instead of TCP very soon.

Thank you

Maziar

6/3/2017

SCUOLA SUPERIORE SANT'ANNA PISA