PostgreSQL HA Database Clusters through Containment

Le Quan Ha

IPG Database Team,

BlackBerry RIM,

176 Columbia St. W., Waterloo, ON, Canada N2L 3L3

403-828-1846

quan-ha.le@tenzing.com

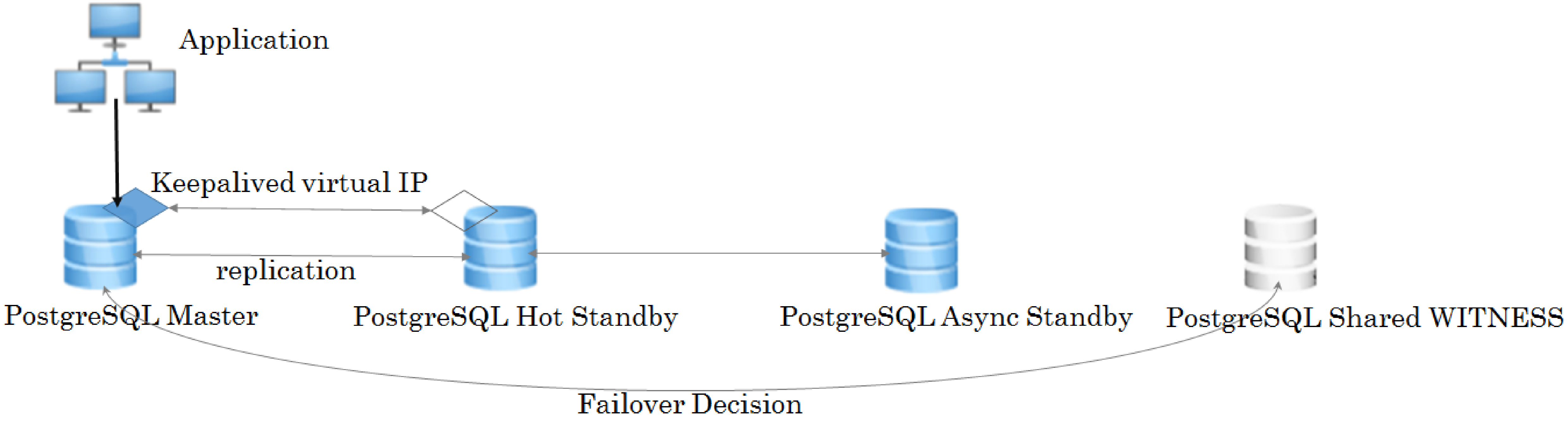
OVERVIEW OF THE REPORT

- ❑ PART 1: THEORETICAL MODELS
- ❑ PART 2: IMPLEMENTATION
- ❑ PART 3: PERFORMANCE ANALYSIS
- ❑ CONCLUSIONS

PART 1: THEORETICAL MODELS

- What is keepalived-repmgr cluster ?
- What is HAProxy-PgBouncer cluster ?

KEEAPLIVED-REPMGR



- Same containment
- Keepalived: VRRP is a fundamental brick for failover
- Repmgr: an open-source tool suite to manage replication in a cluster of PostgreSQL servers
- Failover:
  - Master fails, keepalived will switch the virtual IP to the hot standby
  - Hot standby's VRRP instance of keepalived changes to MASTER state
  - notify_master script is automatically called to promote the hot standby to be a new master

HAPROXY-PGBOUNCER

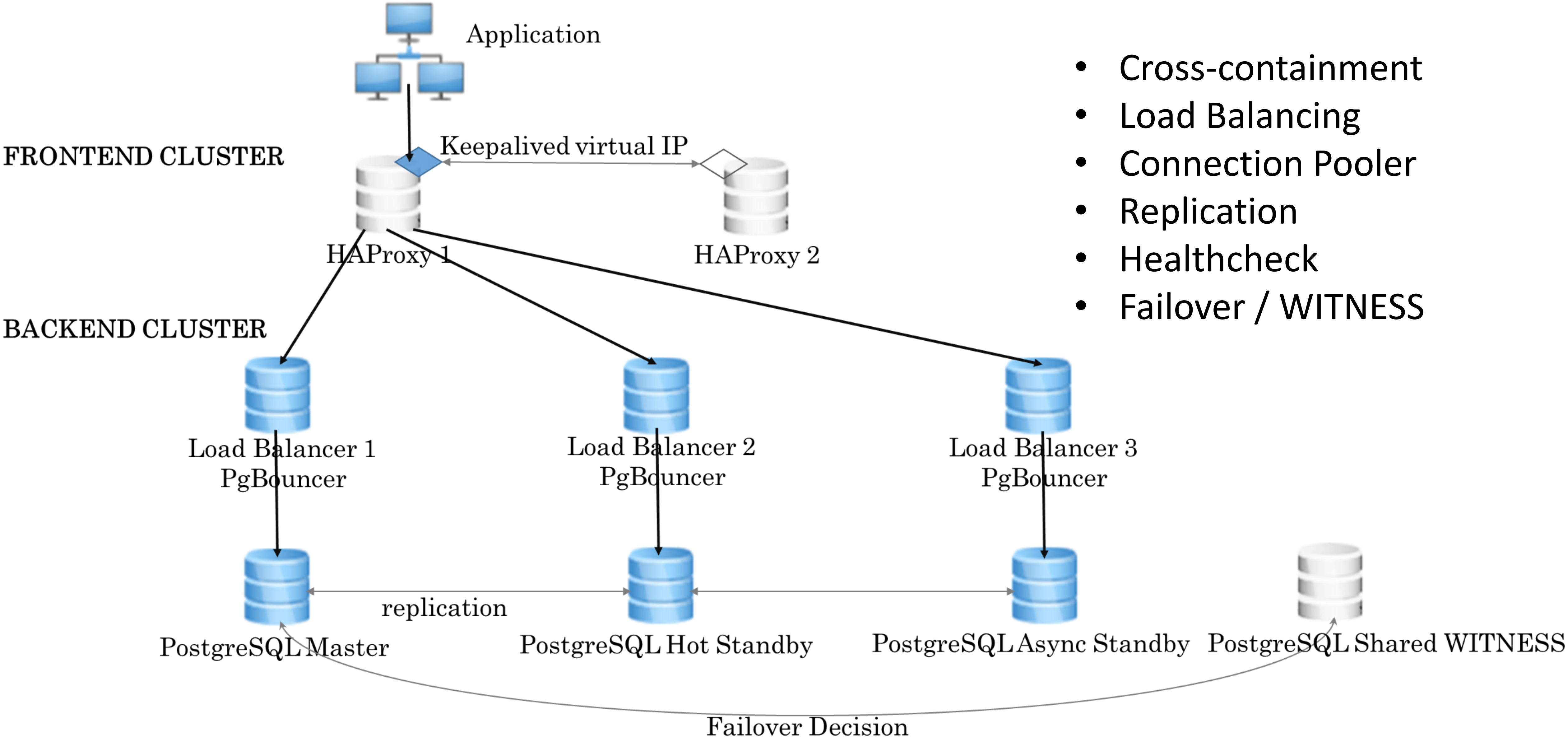
- HAProxy (High Availability Proxy): an open source software TCP/HTTP Load Balancer and proxying solution

- PgBouncer: a lightweight connection pooler for PostgreSQL
  - ➢ Three modes of pooling: session pooling, transaction pooling and statement pooling.
  - ➢ Low memory requirements

- The Frontend servers are inside the same subnet
- Cross-containment: The Backend servers can be in different subnets
- Load balancing: distributing the workload across multiple computing resources

- Shared witness server in clusters: to avoid a "split-brain" situation and control / decide to failover to a privilege standby

HAPROXY-PGBOUNCER



- Cross-containment
- Load Balancing
- Connection Pooler
- Replication
- Healthcheck
- Failover / WITNESS

PART 2: IMPLEMENTATION

- Development of keepalived-repmgr clusters

- Research and development of HAProxy-PgBouncer cluster
  - Flow of Read Requests
  - Flow of Write Requests
  - Statistics Report
  - Farm Failover
  - Auto Failover
  - Frontend cluster: keepalived
  - Backend cluster: PgBouncer
  - Shared WITNESS
  - Switchback

- Development of the Shared WITNESS between 2 different clusters

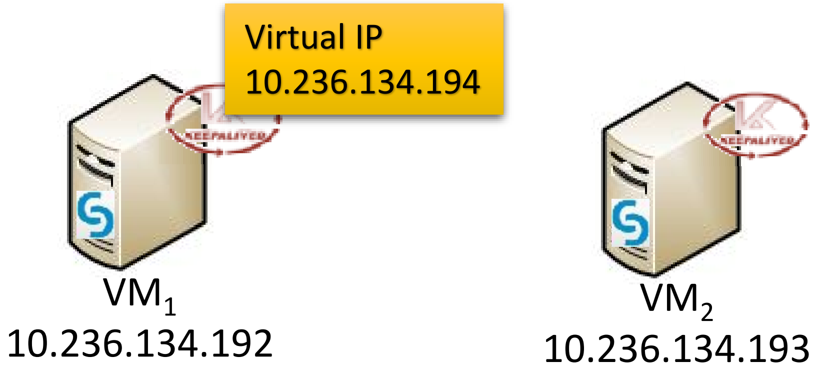
KEEPALIVED-REPMGR

- Altus cloud: 20 network zones (16 productions zones, 2 laboratory zones and 2 restricted pre-production zones)
- SATURN RING software: on 15 production zones
  - 10/02 GE North container 2
  - 10/03 GE North container 3
  - 11/02 GE South container 2
  - 11/03 GE South container 3
  - 5/02 Spirit East container 2
  - 5/03 Spirit East container 3
  - 6/02 Spirit West container 2
  - 6/03 Spirit West container 3
  - 7/01 Viking Container 1
  - ONELAB - Orion
  - ONELAB - Thor
  - 21/01 Stardust
  - Casino
  - HongKong
  - Mercury

HAPROXY-PGBOUNCER

FRONTEND CLUSTER

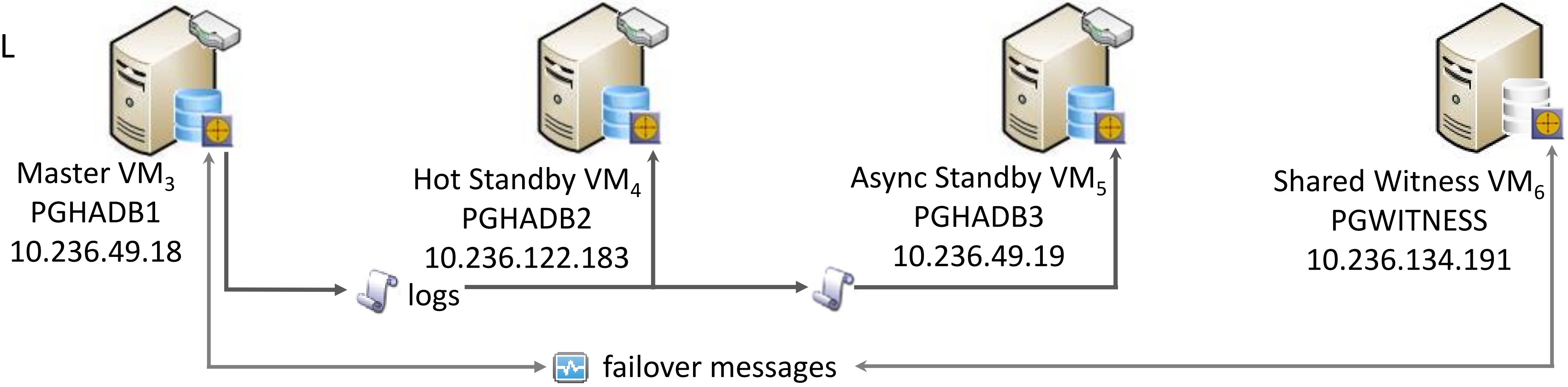
- Empty PostgreSQL
- HAProxy
- Keepalived



- Our executable actual implementation from the theoretical model

BACKEND CLUSTER

- PostgreSQL
- repmgr
- repmgrd
- PgBouncer



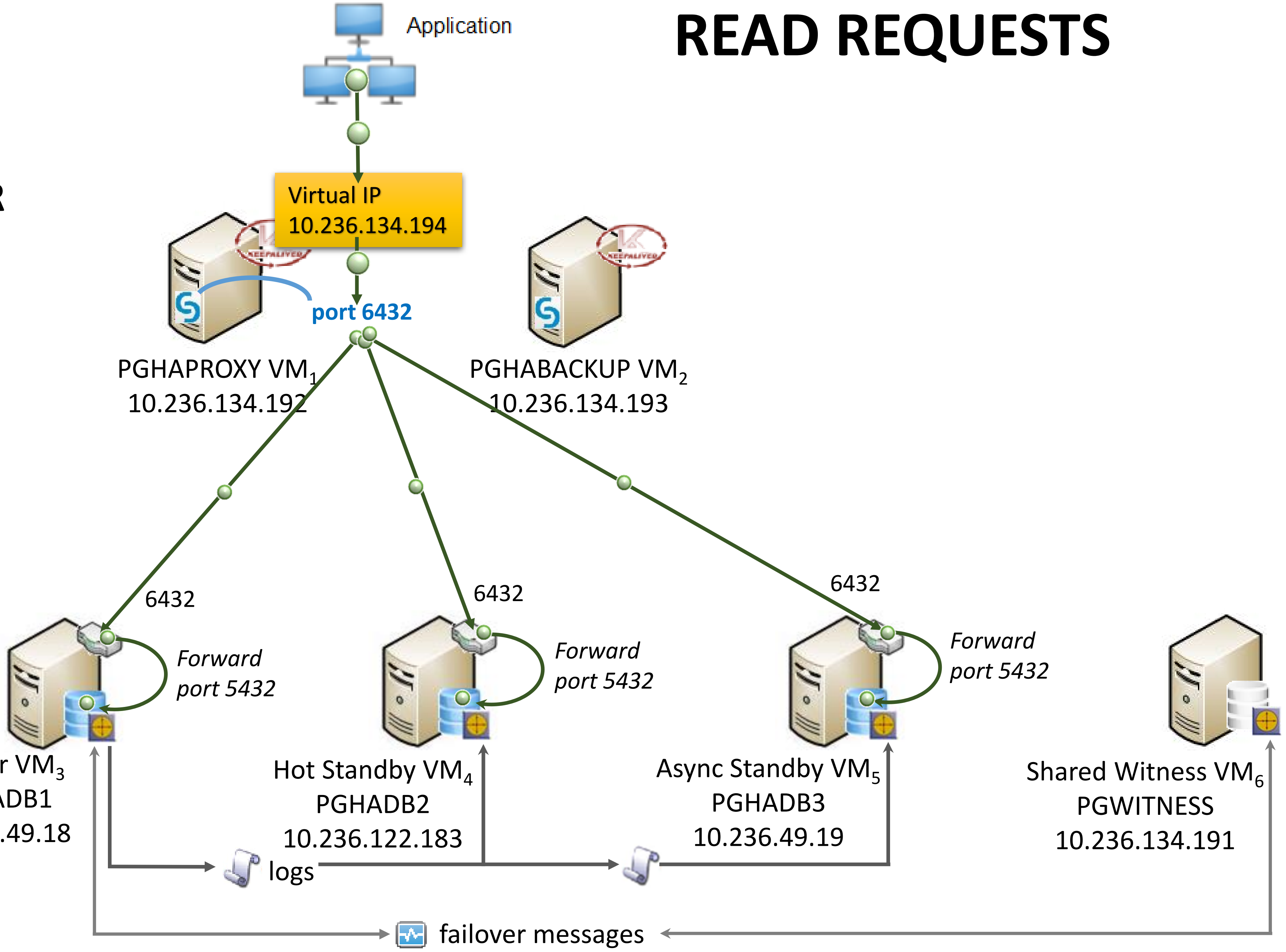
FLOW OF READ REQUESTS

FRONTEND CLUSTER

- Empty PostgreSQL
- HAProxy
- Keepalived

BACKEND CLUSTER

- PostgreSQL
- repmgr
- repmgrd
- PgBouncer



Application

Virtual IP
10.236.134.194

port 6432

PGHAPROXY VM$_1$
10.236.134.192

PGHABACKUP VM$_2$
10.236.134.193

6432

6432

6432

Forward port 5432

Forward port 5432

Forward port 5432

Master VM$_3$
PGHADB1
10.236.49.18

Hot Standby VM$_4$
PGHADB2
10.236.122.183

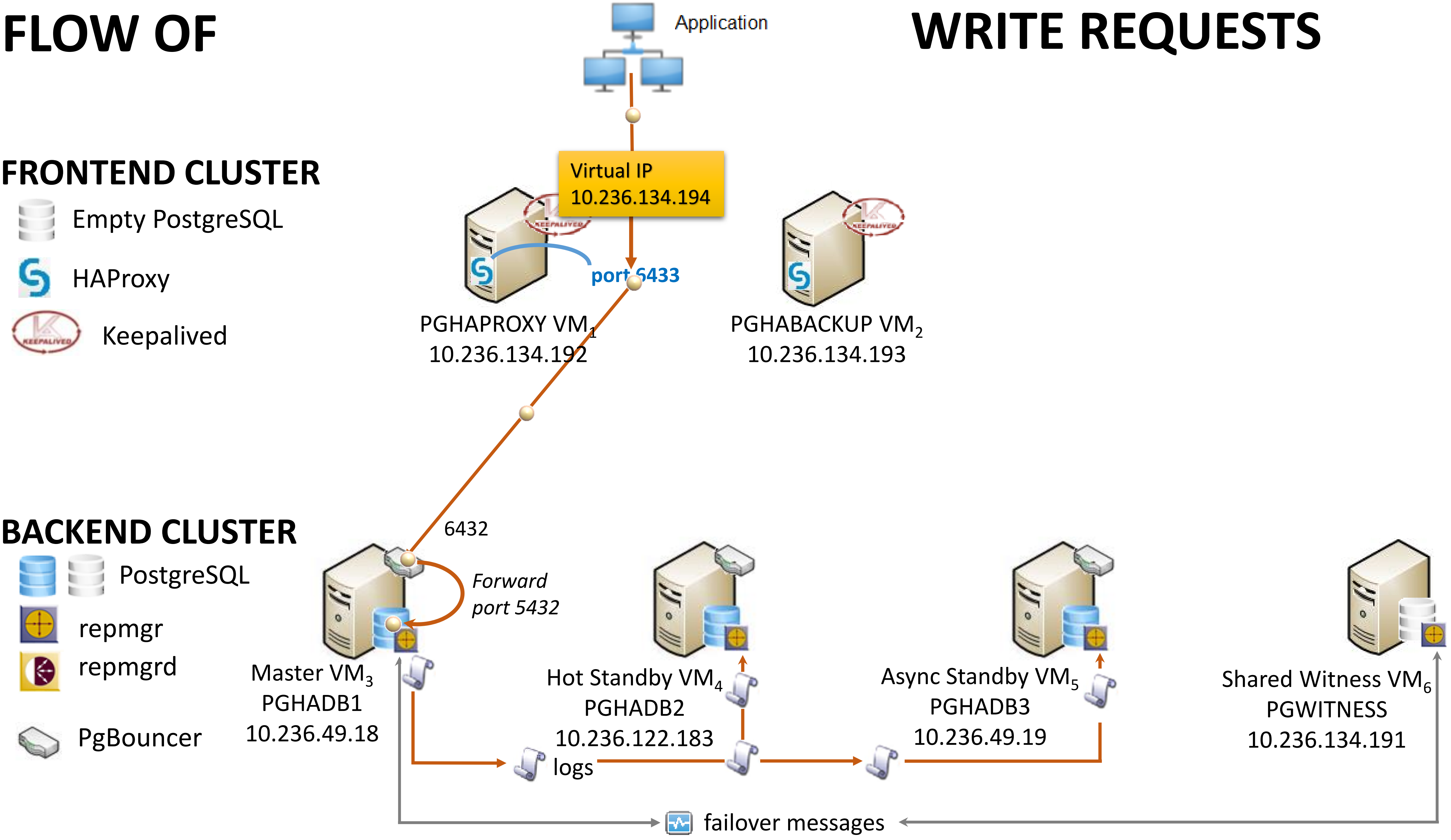
Async Standby VM$_5$
PGHADB3
10.236.49.19

Shared Witness VM$_6$
PGWITNESS
10.236.134.191

logs

failover messages

FLOW OF WRITE REQUESTS



Application

Virtual IP 10.236.134.194

port 6433

FRONTEND CLUSTER

- Empty PostgreSQL
- HAProxy
- Keepalived

PGHAPROXY VM$_1$
10.236.134.192

PGHABACKUP VM$_2$
10.236.134.193

6432

Forward port 5432

BACKEND CLUSTER

- PostgreSQL
- repmgr
- repmgrd
- PgBouncer

Master VM$_3$
PGHADB1
10.236.49.18

Hot Standby VM$_4$
PGHADB2
10.236.122.183

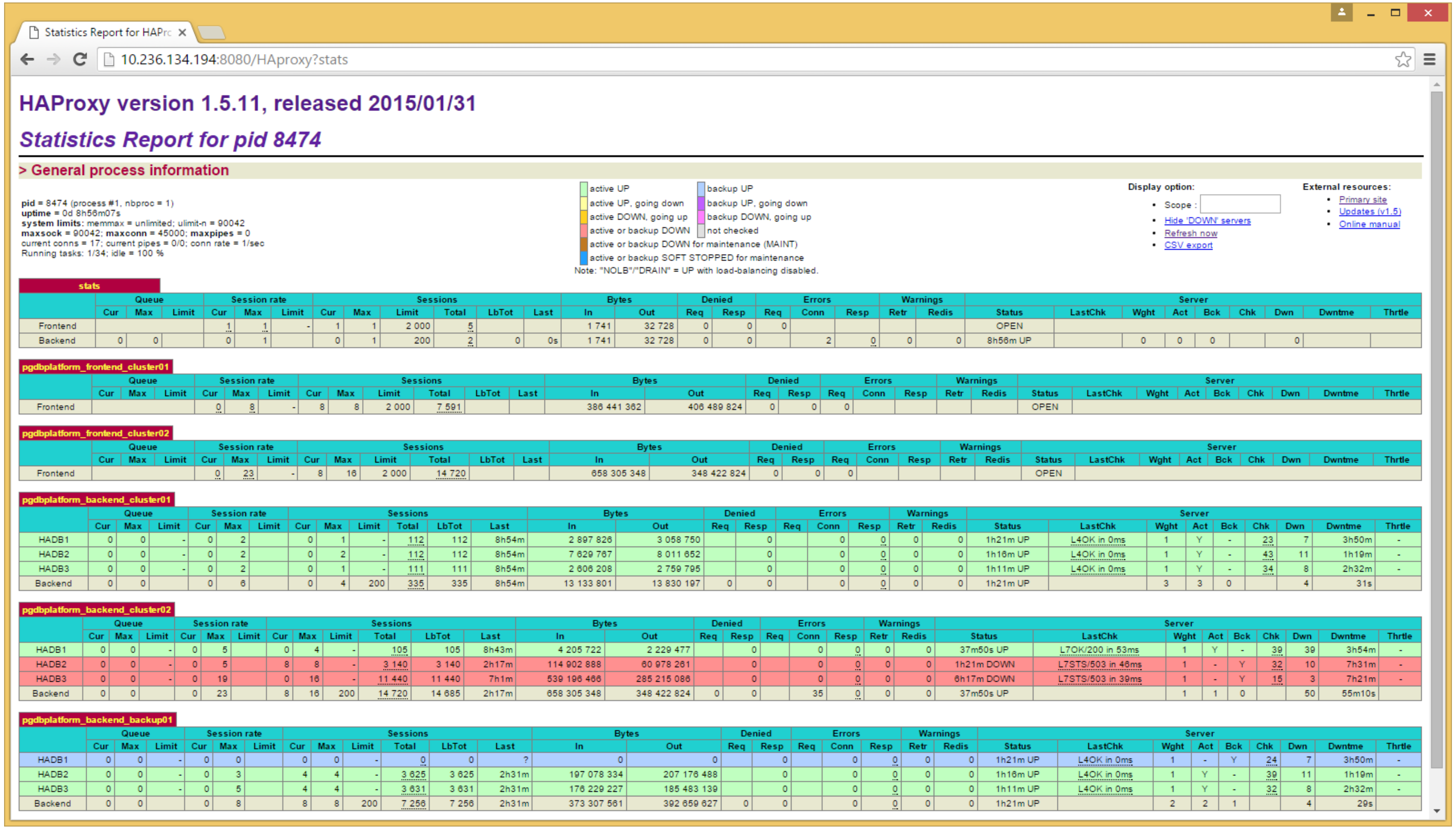
Async Standby VM$_5$
PGHADB3
10.236.49.19

Shared Witness VM$_6$
PGWITNESS
10.236.134.191

logs

failover messages

http://10.236.134.194:8080/HAproxy?stats


Statistics Report for HAPro
10.236.134.194:8080/HAproxy?stats

HAProxy version 1.5.11, released 2015/01/31

Statistics Report for pid 8474

> General process information

pid = 8474 (process #1, nbproc = 1)
uptime = 0d 8h56m07s
system limits: memmax = unlimited; ulimit-n = 90042
maxsock = 90042; **maxconn** = 45000; **maxpipes** = 0
current conns = 17; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/34; idle = 100 %

active UP	backup UP
active UP, going down	backup UP, going down
active DOWN, going up	backup DOWN, going up
active or backup DOWN	not checked
active or backup DOWN for maintenance (MAINT)	
active or backup SOFT STOPPED for maintenance	

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	1	-	1	1	2 000	5			1 741	32 728	0	0	0					OPEN								
Backend	0	0		0	1		0	1	200	2	0	0s	1 741	32 728	0	0		2	0	0	0	8h56m UP		0	0	0		0		

pgdbplatform_frontend_cluster01

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	8	-	8	8	2 000	7 591			386 441 362	406 489 824	0	0	0					OPEN								

pgdbplatform_frontend_cluster02

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	23	-	8	16	2 000	14 720			658 305 348	348 422 824	0	0	0					OPEN								

pgdbplatform_backend_cluster01

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
HADB1	0	0	-	0	2		0	1	-	112	112	8h54m	2 897 826	3 058 750		0		0	0	0	0	1h21m UP	L4OK in 0ms	1	Y	-	23	7	3h50m	-
HADB2	0	0	-	0	2		0	2	-	112	112	8h54m	7 629 767	8 011 652		0		0	0	0	0	1h16m UP	L4OK in 0ms	1	Y	-	43	11	1h19m	-
HADB3	0	0	-	0	2		0	1	-	111	111	8h54m	2 606 208	2 759 795		0		0	0	0	0	1h11m UP	L4OK in 0ms	1	Y	-	34	8	2h32m	-
Backend	0	0		0	6		0	4	200	335	335	8h54m	13 133 801	13 830 197	0	0		0	0	0	0	1h21m UP		3	3	0		4	31s	

pgdbplatform_backend_cluster02

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
HADB1	0	0	-	0	5		0	4	-	105	105	8h43m	4 205 722	2 229 477		0		0	0	0	0	37m50s UP	L7OK/200 in 53ms	1	Y	-	39	39	3h54m	-
HADB2	0	0	-	0	5		8	8	-	3 140	3 140	2h17m	114 902 888	60 978 261		0		0	0	0	0	1h21m DOWN	L7STS/503 in 46ms	1	-	Y	32	10	7h31m	-
HADB3	0	0	-	0	19		0	16	-	11 440	11 440	7h1m	539 196 466	285 215 086		0		0	0	0	0	6h17m DOWN	L7STS/503 in 39ms	1	-	Y	15	3	7h21m	-
Backend	0	0		0	23		8	16	200	14 720	14 685	2h17m	658 305 348	348 422 824	0	0		35	0	0	0	37m50s UP		1	1	0		50	55m10s	

pgdbplatform_backend_backup01

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
HADB1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	1h21m UP	L4OK in 0ms	1	-	Y	24	7	3h50m	-
HADB2	0	0	-	0	3		4	4	-	3 625	3 625	2h31m	197 078 334	207 176 488		0		0	0	0	0	1h16m UP	L4OK in 0ms	1	Y	-	39	11	1h19m	-
HADB3	0	0	-	0	5		4	4	-	3 631	3 631	2h31m	176 229 227	185 483 139		0		0	0	0	0	1h11m UP	L4OK in 0ms	1	Y	-	32	8	2h32m	-
Backend	0	0		0	8		8	8	200	7 256	7 256	2h31m	373 307 561	392 659 627	0	0		0	0	0	0	1h21m UP		2	2	1		4	29s	

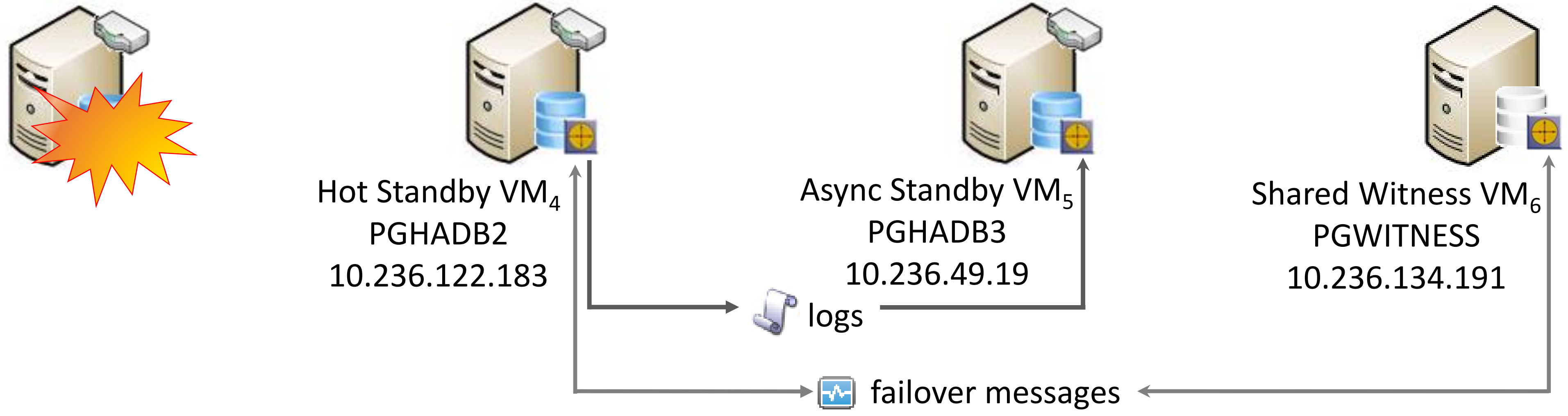
Statistics can be defined as

```
listen stats 0.0.0.0:8080
    mode http
    stats enable
    stats uri /HAproxy?stats
    stats realm Strictly\ Private
    stats auth admin:admin
```

FARM Failover of HAProxy

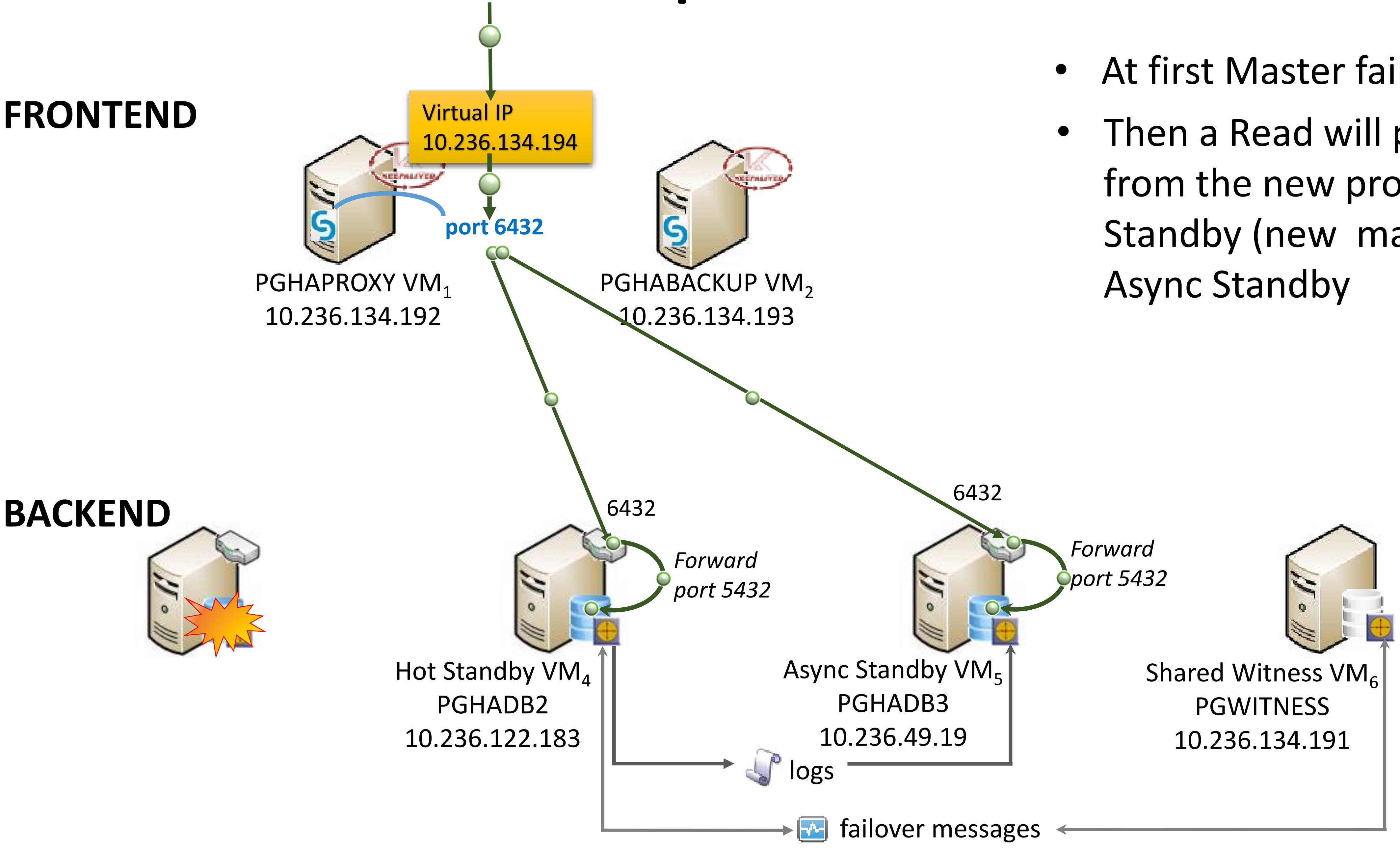
- When the master database fails
- The Hot Standby will be promoted to be the new master

BACKEND CLUSTER



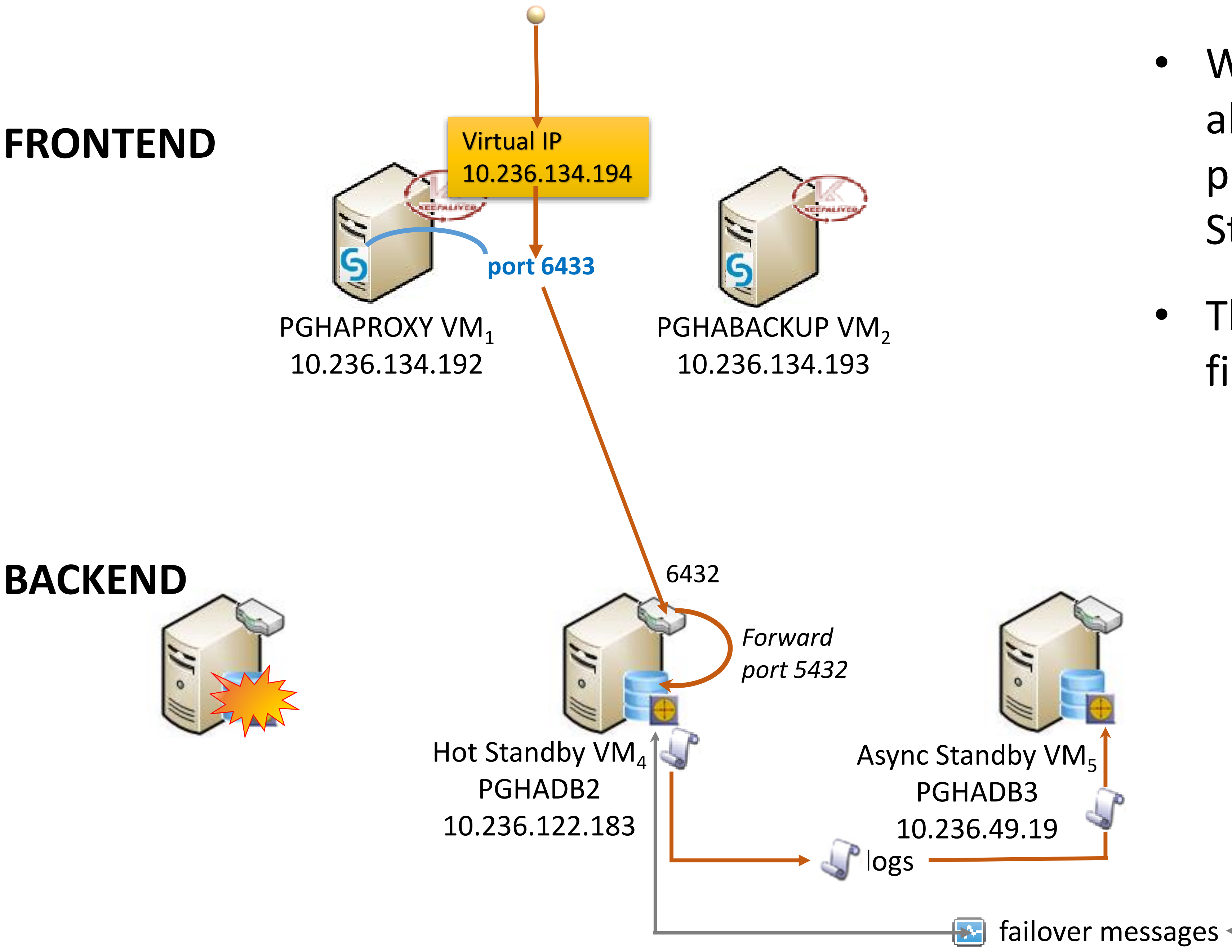
Master fails

FARM Failover – Read Requests



- At first Master fails
- Then a Read will performs from the new promoted Hot Standby (new master) and Async Standby

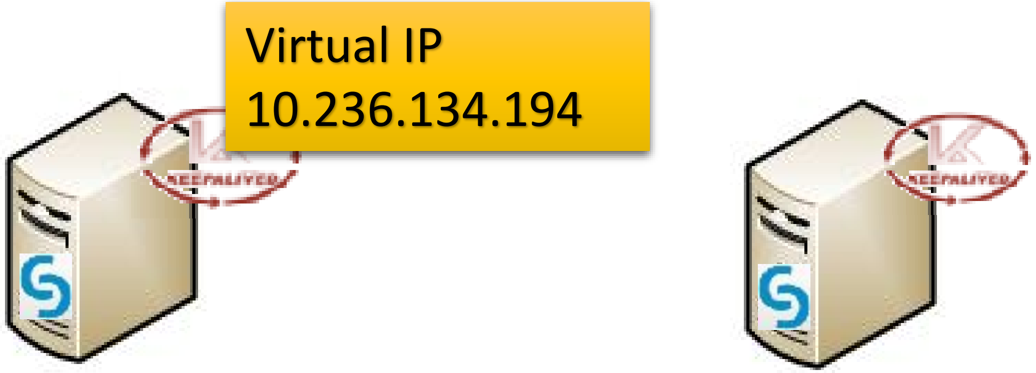
AUTO-FAILOVER



- When the Master fails, Writes also performs to the new promoted master (Hot Standby) only
- The Async Standby receives log files to sync its Database

FRONTEND CLUSTER

Keepalived



Virtual IP
10.236.134.194

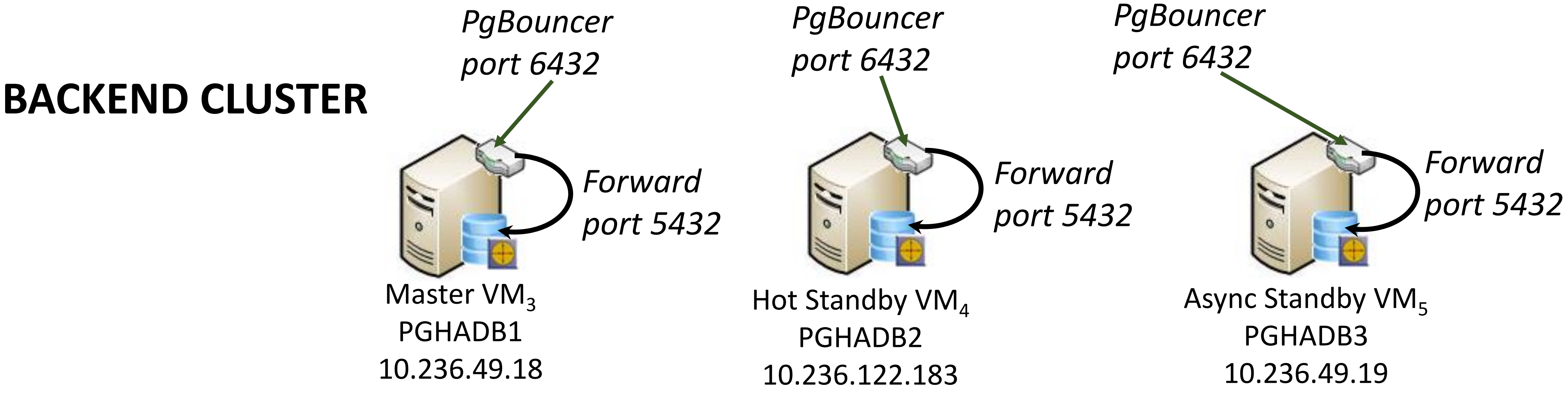
PGHAPROXY VM$_1$
10.236.134.192

PGHABACKUP VM$_2$
10.236.134.193

- Keepalived understands that the 2 VMs are sharing one instance
- There should be one Master VM and one Backup VM
- A virtual IP is defined
- The Master will keep the virtual IP by default
- If the Master fails, timing control to switch the virtual IP for the Backup VM

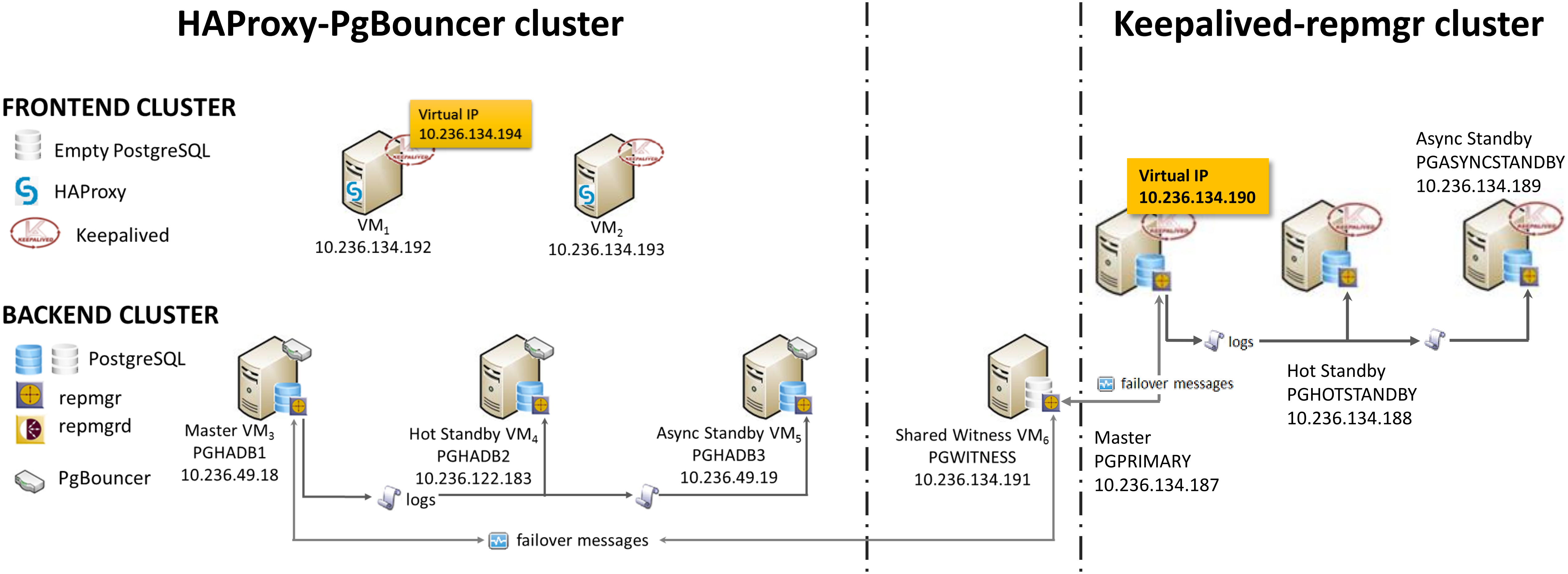
BACKEND CLUSTER: PgBouncer

- PgBouncer are installed on Master, Hot Standby and Async Standby of the Backend
- Connection Pooler by PgBouncer
- Receiving requests from PgBouncer port 6432
- Forward to port 5432



Shared WITNESS

- The WITNESS is shared between the current HAProxy-PgBouncer cross-containment cluster and a second Keepalived-repmgr cluster that is not cross-containment



Shared WITNESS

- In Witness, there are 2 different repmgr.conf config files for 2 clusters

  priority=-1
- HAProxy-PgBouncer

  /db/postgres_config/main/repmgr.conf
- Keepalived-repmgr

  /db/postgres_config/main/repmgr/witness/repmgr.conf

- How to create

```
postgres@PGWITNESS:~$ repmgr -d postgres -U repmgr -h   10.236.49.18
   -D /var/lib/postgresql/9.4/main -f   /db/postgres_config/main/repmgr.conf
   witness create --force --verbose

postgres@PGWITNESS:~$ repmgr -d postgres -U repmgr -h   10.236.134.187
   -D /var/lib/postgresql/9.4/main -f   /db/postgres_config/main/repmgr/witness/repmgr.conf
   witness create --force --verbose
```

Shared WITNESS

- How to show 2 different clusters on the same WITNESS

```
root@PGWITNESS:~# repmgr -f      /db/postgres_config/main/repmgr.conf          cluster show
Role      | Connection String
* master  | host=10.236.49.18 user=repmgr password=passw0rd dbname=postgres
  witness | host=10.236.134.191 user=repmgr password=passw0rd dbname=postgres
  standby | host=10.236.122.183 user=repmgr password=passw0rd dbname=postgres
  standby | host=10.236.49.19 user=repmgr password=passw0rd dbname=postgres
```

```
root@PGWITNESS:~# repmgr –f   /db/postgres_config/main/repmgr/witness/repmgr.conf  cluster show
Role      | Connection String
* master  | host=10.236.134.187 user=repmgr password=passw0rd dbname=postgres
  standby | host=10.236.134.188 user=repmgr password=passw0rd dbname=postgres
  standby | host=10.236.134.189 user=repmgr password=passw0rd dbname=postgres
  witness | host=10.236.134.191 user=repmgr password=passw0rd dbname=postgres
```

Switchback for the HAProxy-PgBouncer cluster

BACKEND CLUSTER:

- repmgr clone the failed Master to the new promoted master (Hot Standby)
- Start service on the failed Master
- Follow the new promoted master on the failed Master
- ➢ THE FAILED MASTER IS NOW RESTARTED AS A NEW STANDBY

- Stop service on the new promoted master (Hot Standby)
- Promote the failed Master to be back to Master node
- ➢ THE FAILED MASTER IS NOW BACK AS MASTER AGAIN

- repmgr clone the Hot Standby to the Master
- Start service on the Hot Standby
- Follow the Master on the Hot Standby

THE NEW PROMOTED MASTER (HOT STANDBY) IS NOW BACK AS HOT STANDBY AGAIN

- Follow the Master on the Async Standby (and any other Standbys)

FRONTEND CLUSTER

- Restart haproxy service on HAProxy-1 to return haproxy’s load balancing setting

PART 3: PERFORMANCE ANALYSIS

- Methodology

- Keepalived-repmgr Throughputs
- Keepalived-repmgr I/O and CPU Graphs
- Keepalived Failover CPU Graph

- HAProxy-PgBouncer Throughputs
- HAProxy-PgBouncer Load Balancing
- HAProxy-PgBouncer I/O and CPU Graphs
- HAProxy-PgBouncer Failover CPU Graphs

- Performance Comparison: keepalived-repmgr vs. HAProxy-PgBouncer

METHODOLOGY

- Apache JMeter v2.13 to create test plans of 1 million samples/each

$$Throughput = \frac{Number\ of\ Transactions}{Real\ Execution\ in\ seconds} \qquad (1)$$

$$KB/sec = \frac{Throughput * Avg.\ Bytes}{1024} \qquad (2)$$

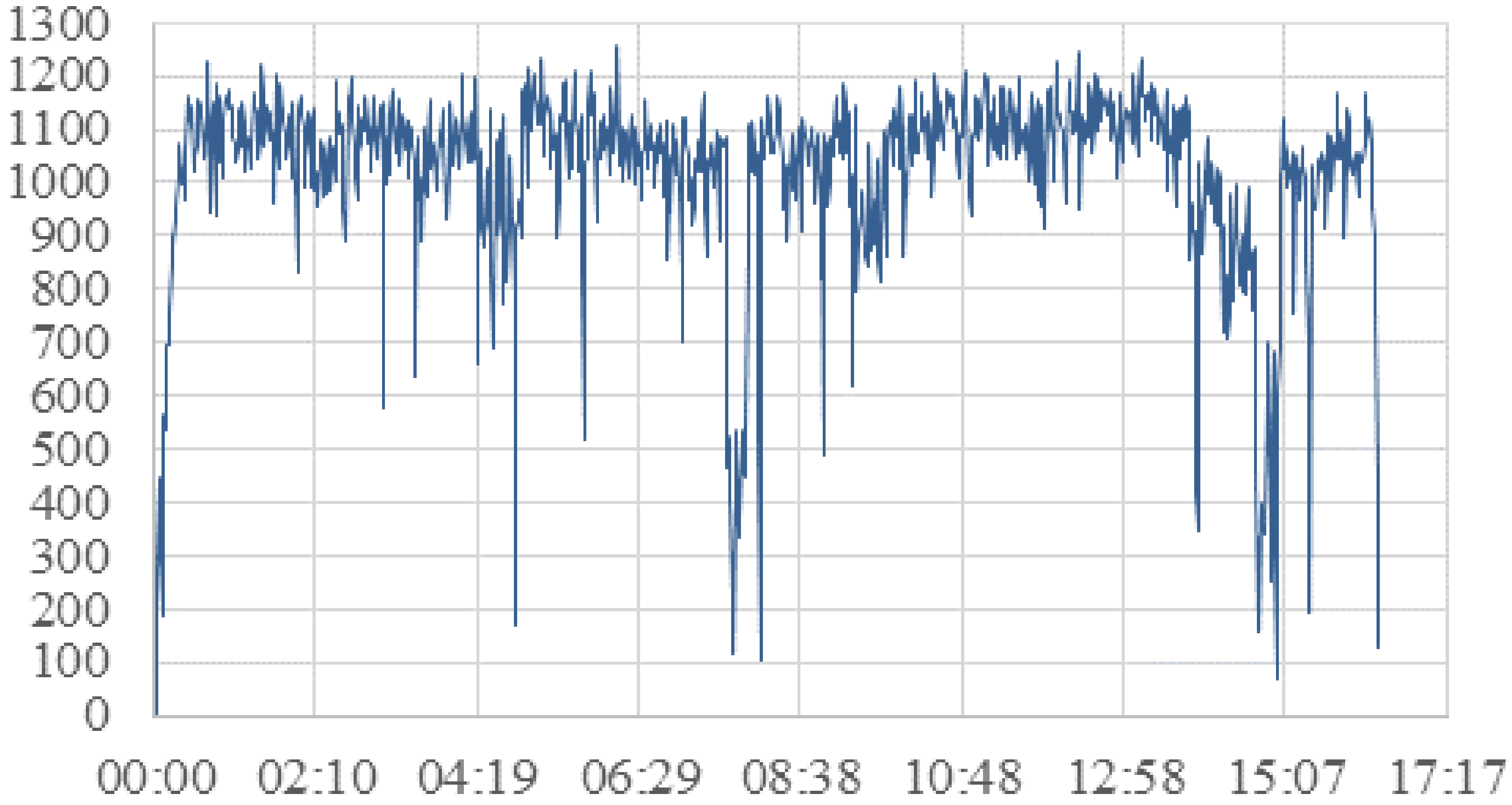
- 6 performance tests by HTTP Requests:
  - ➢ Read Only without data execution
  - ➢ Read Only with data execution
  - ➢ Simple Write with Inserts and Updates
  - ➢ Simple Write with Deletes
  - ➢ Read Write with Selects, Inserts and Updates
  - ➢ Read Write with Selects and Deletes.

- Each performance test, we report 8 graphs for
  - ➢ Transactions per Second
  - ➢ CPU Usages
  - ➢ Active Threads
  - ➢ Response Time
  - ➢ Bytes Throughput over Time
  - ➢ Response Times Percentiles
  - ➢ Response Times vs Threads
  - ➢ Transaction Throughput vs Threads

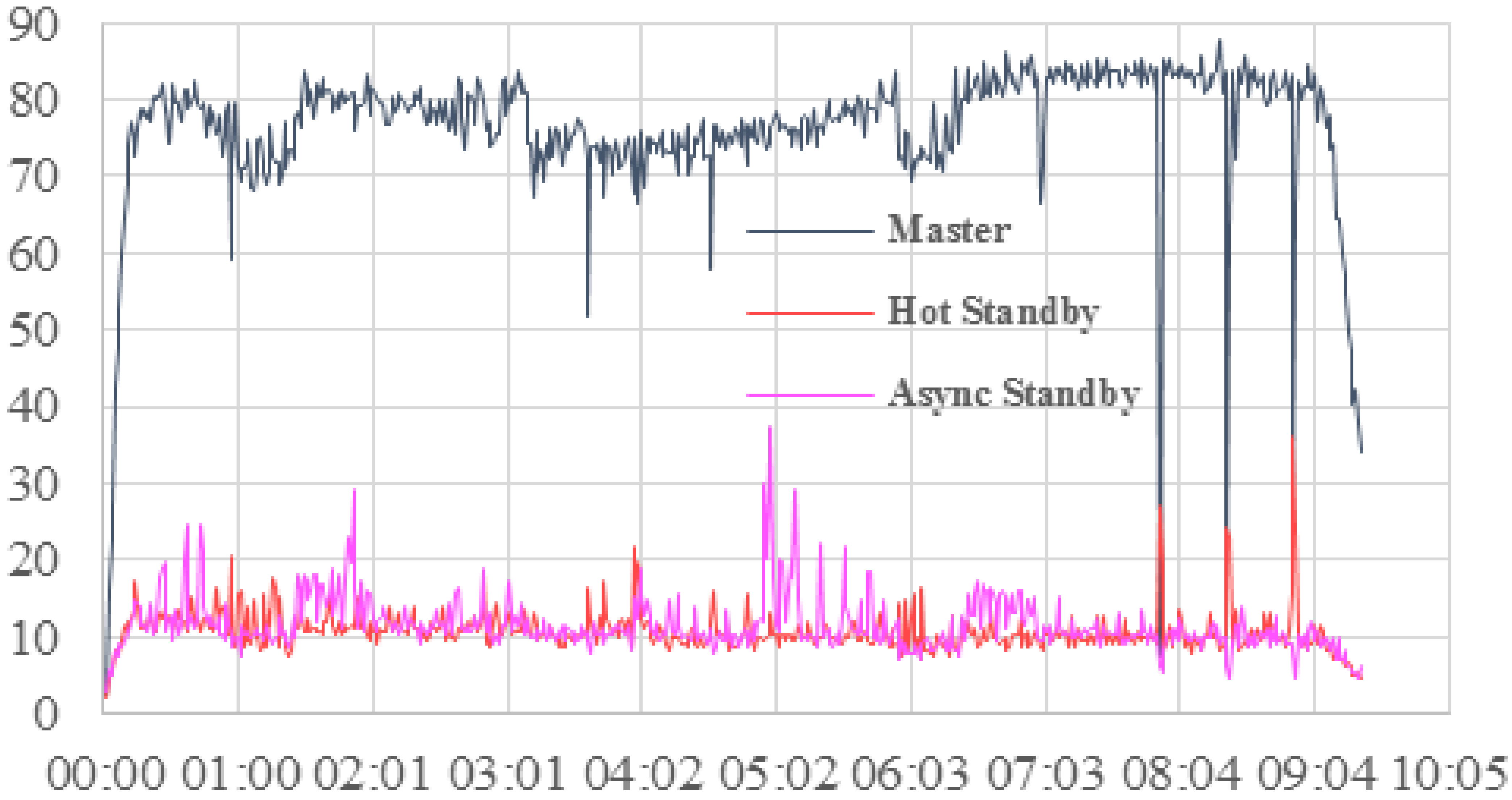
KEEPALIVED-REPMGR THROUGHPUT

HTTP Request	Test duration	Avg. Response Time /sample	Throughput	KB/sec	Avg. Bytes /transaction	Avg. Latency /transaction
Read Only without data execution	463.976s	27.494s	2,155.284	609.467	289.565	27.443
Read Only with data execution	478.775s	29.951s	2,088.664	8,063.858	3,953.432	29.909
Simple Write with Inserts and Updates	753.529s	58.480s	1,327.089	376.119	290.219	58.418
Simple Write with Deletes	533.122s	31.481s	1,875.743	530.419	289.565	31.421
Read Write with Selects, Inserts and Updates	981.059s	80.431s	1,019.307	288.666	289.995	80.372
Read Write with Selects and Deletes	570.773s	37.486s	1,752.010	493.719	288.565	37.426

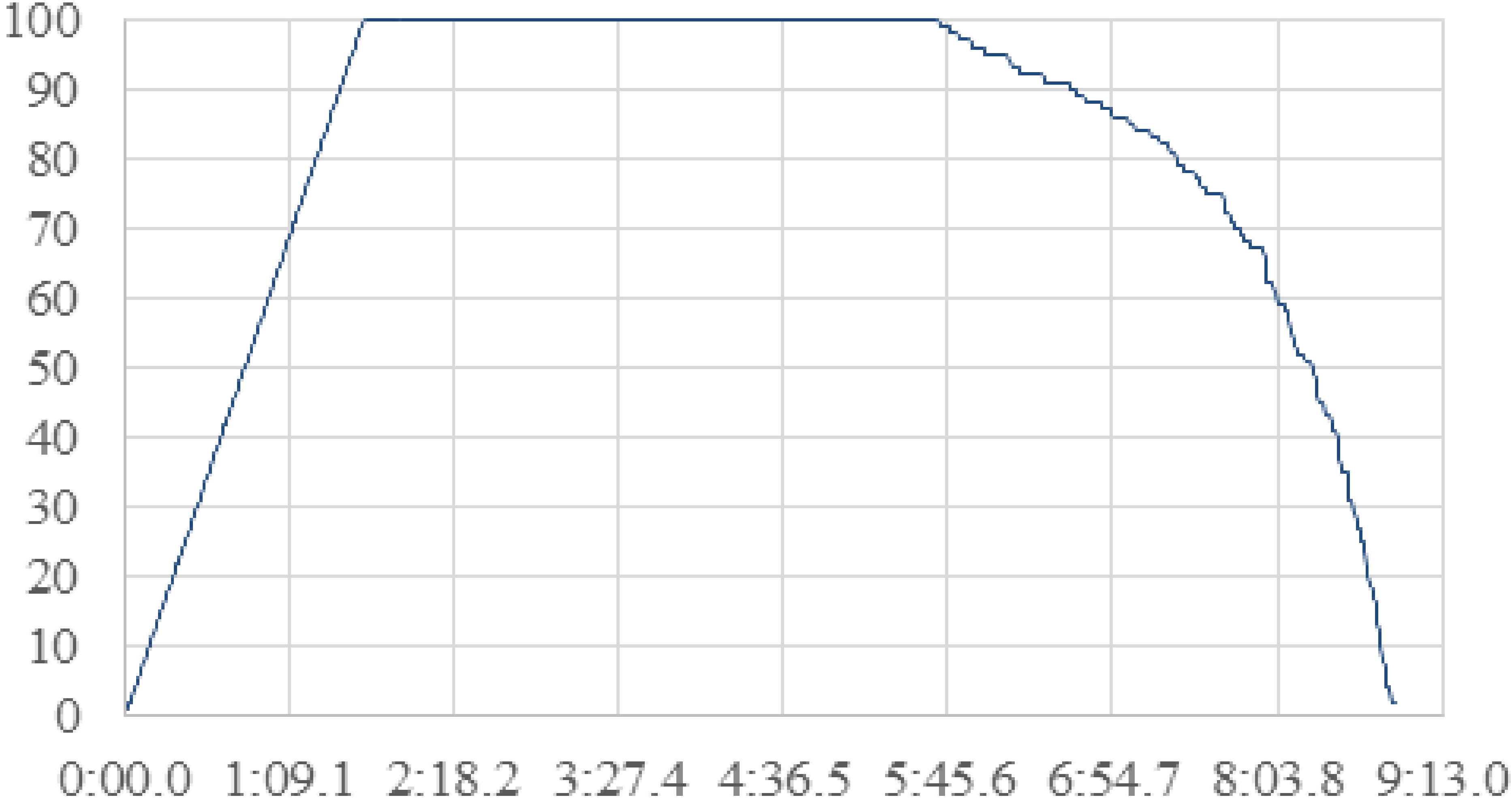
Keepalived-repmgr: Read Write with Selects, Inserts and Updates - Transactions per Second



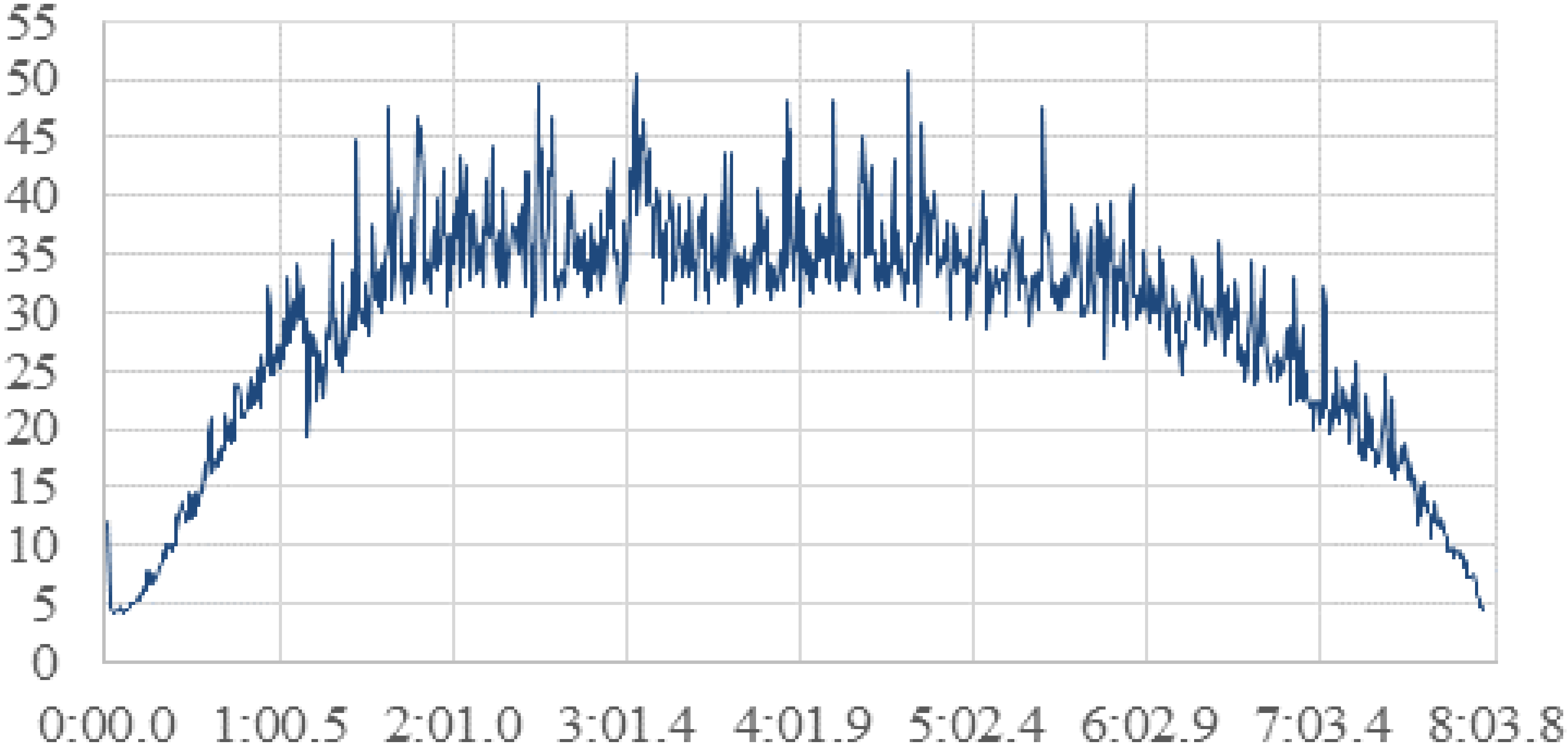
Keepalived-repmgr: Read Write with Selects and Deletes - CPU Usages



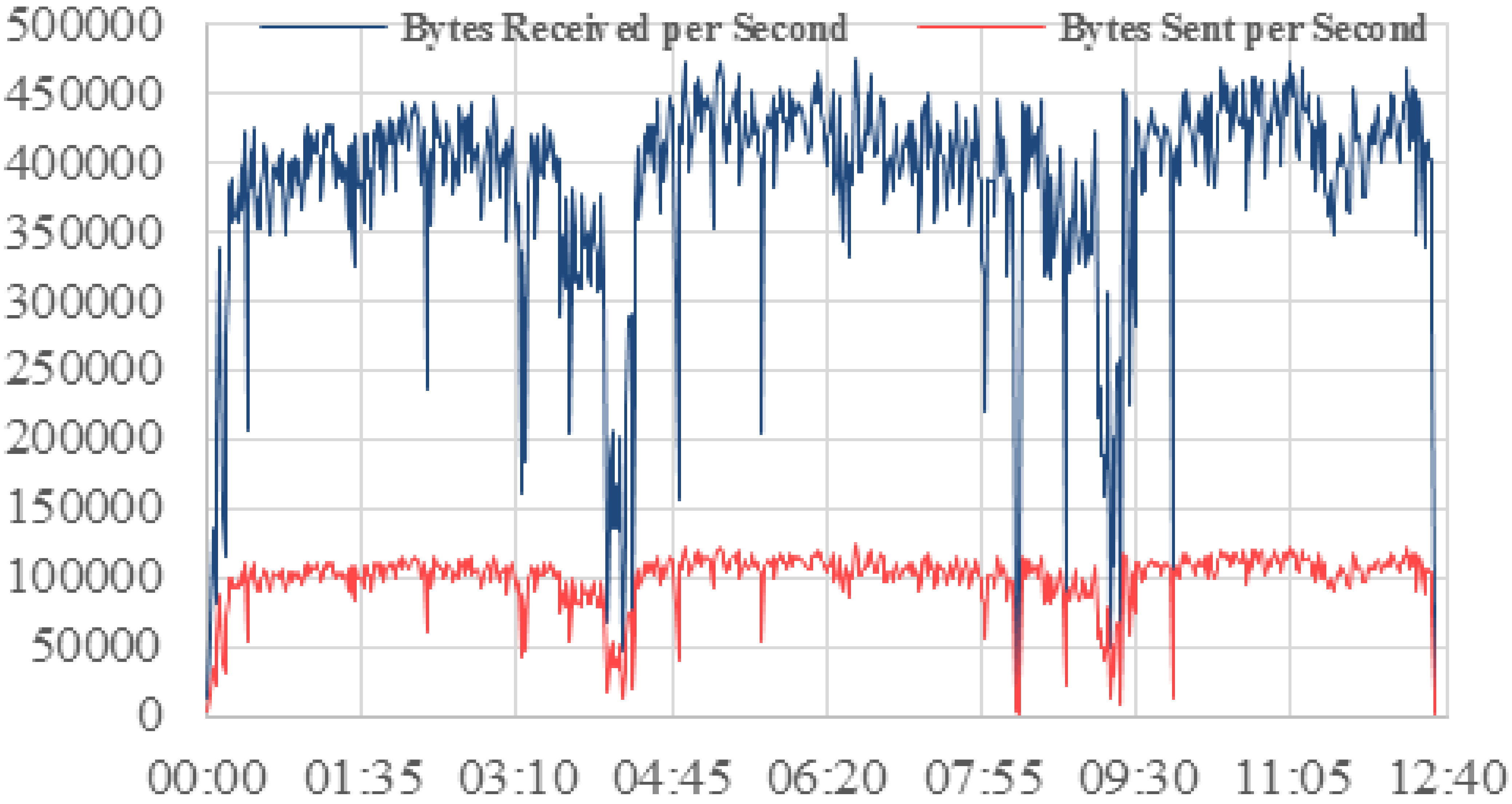
Keepalived-repmgr: Simple Write with Deletes - Active Threads



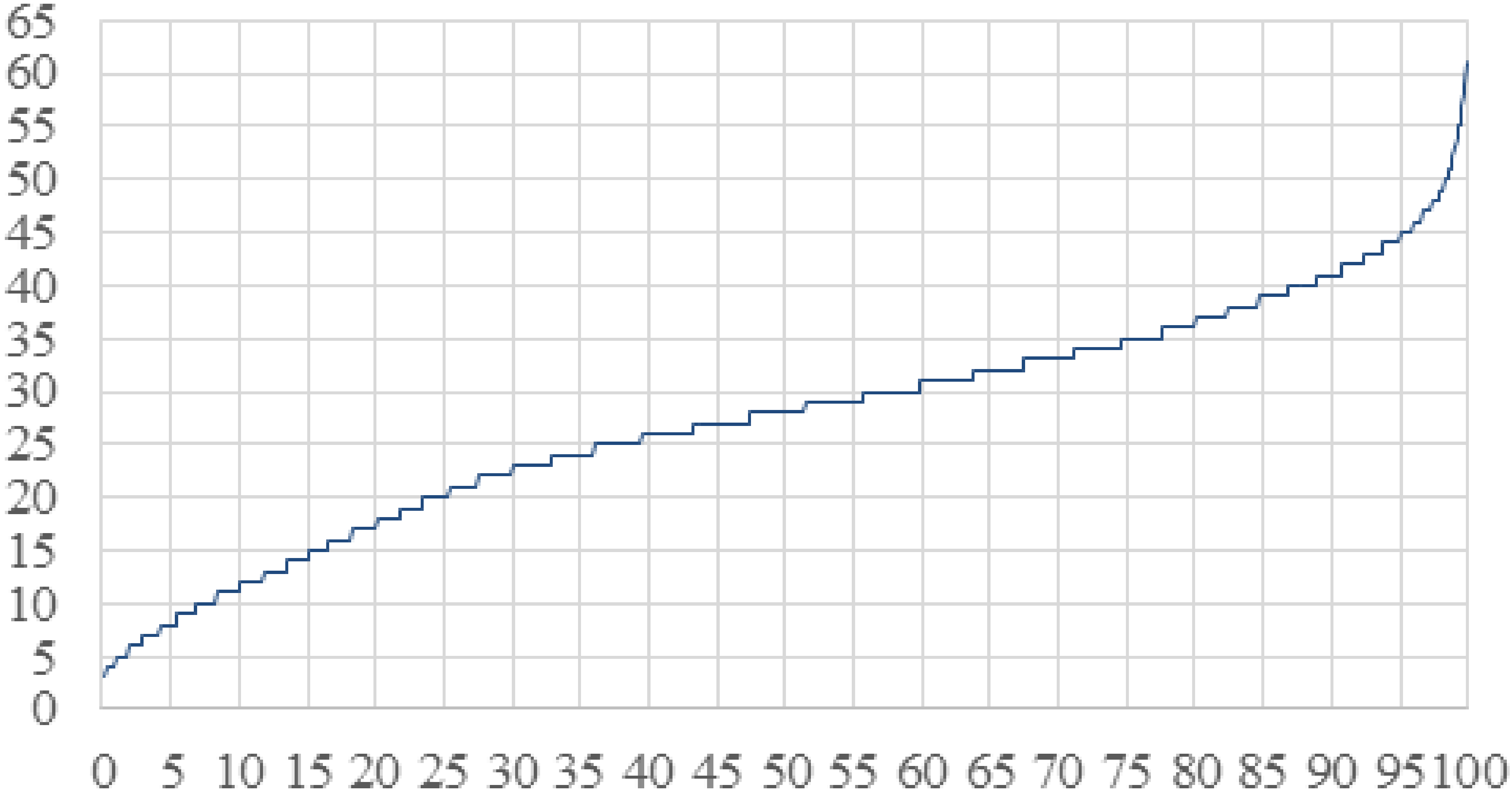
Keepalived-repmgr: Read Only with Data Execution - Response Time



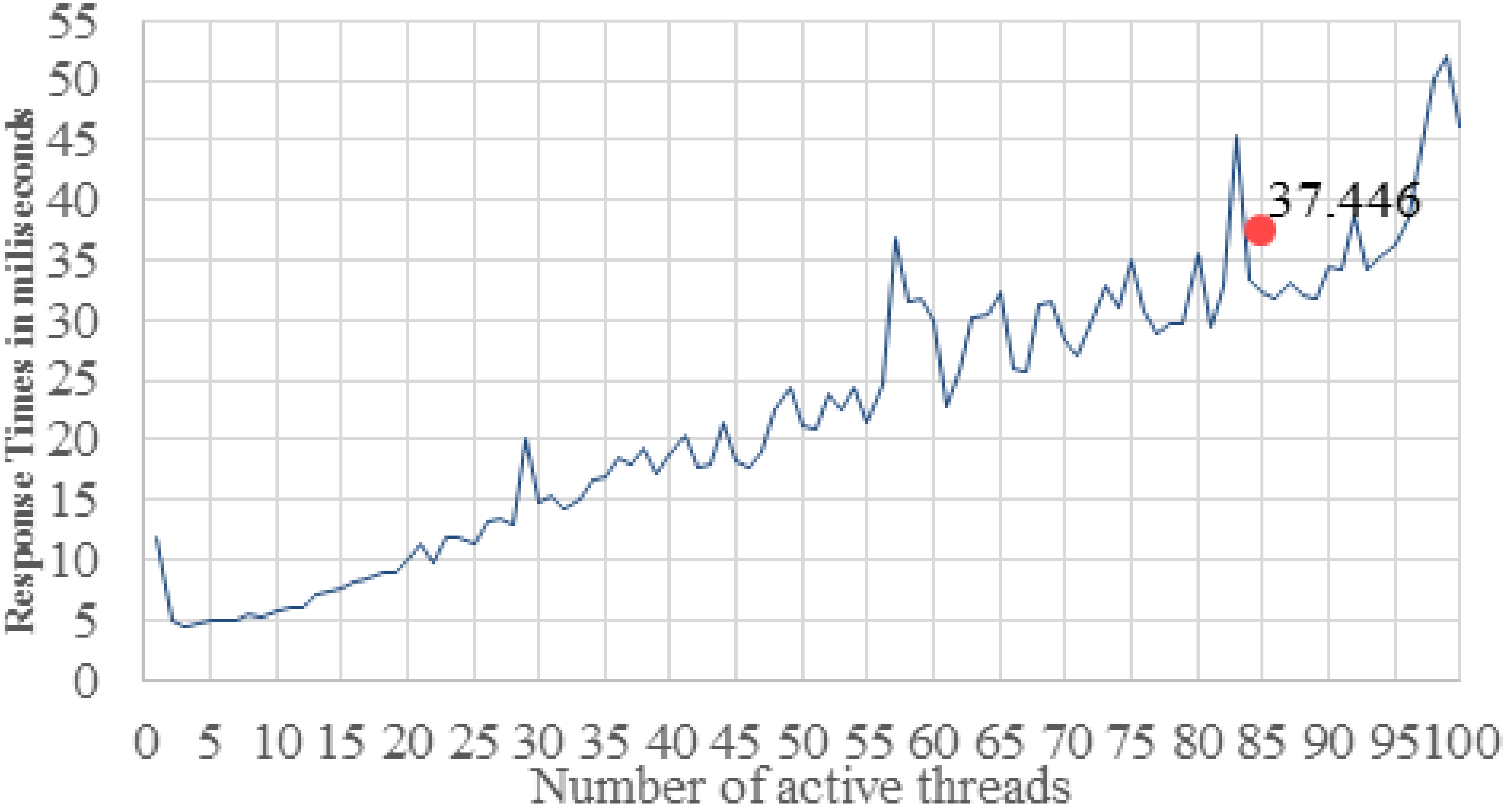
Keepalived-repmgr: Simple Write with Inserts and Updates - Bytes Throughput over Time



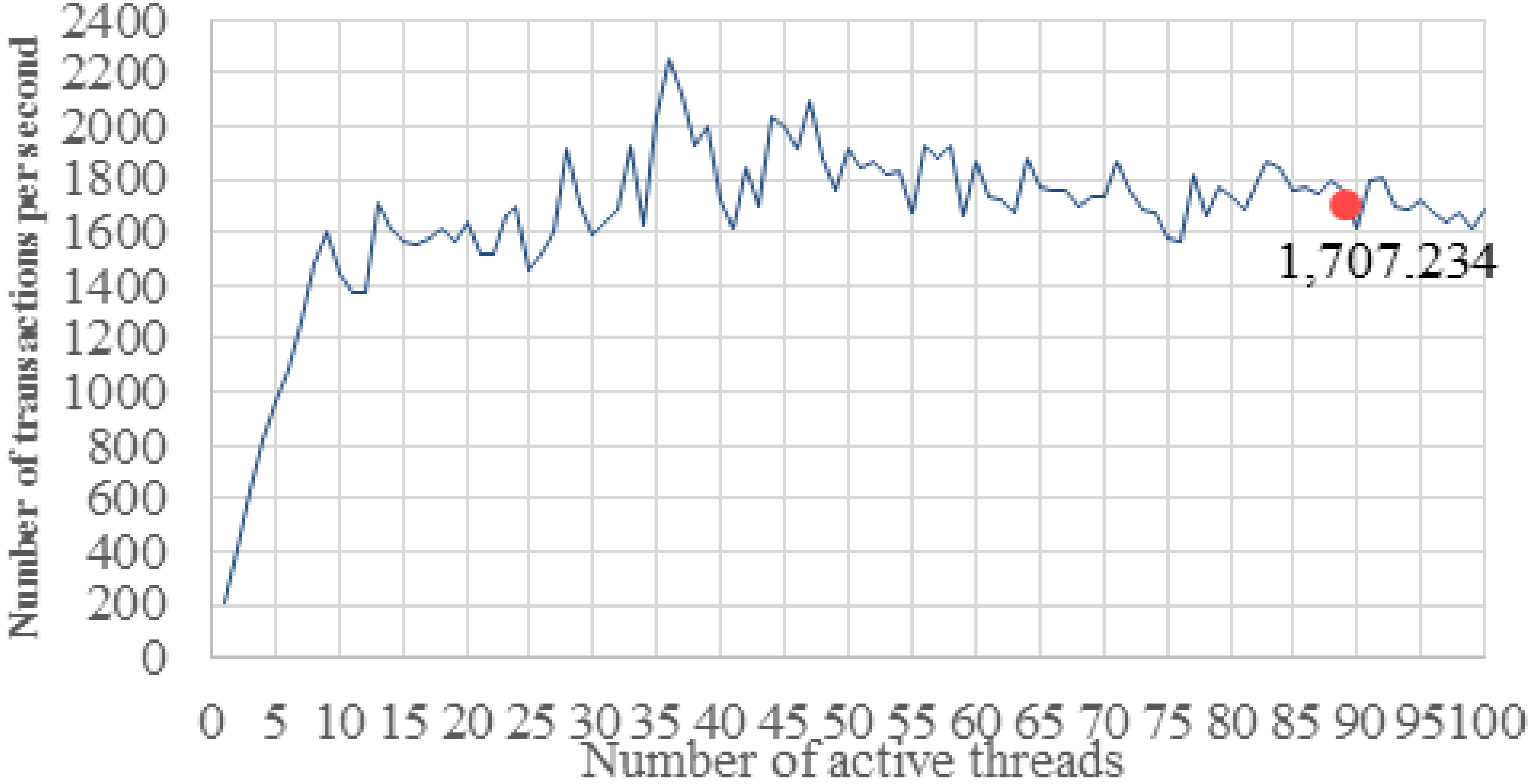
Keepalived-repmgr: Read Only without Data Execution - Response Times Percentiles



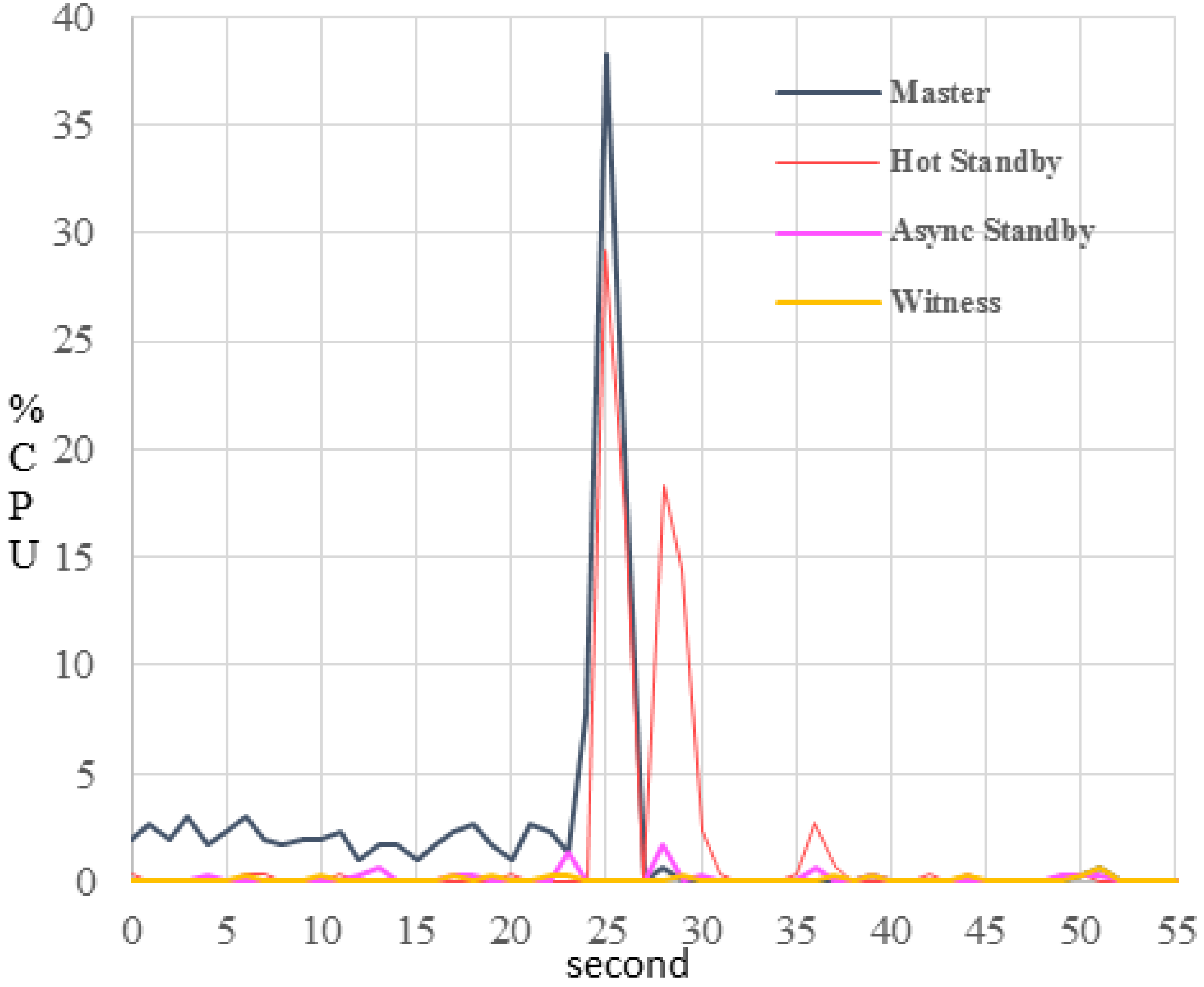
Keepalived-repmgr: Read Write with Selects and Deletes - Response Times vs Threads



Keepalived-repmgr: Simple Write with Inserts and Updates - Transaction Throughput vs Threads



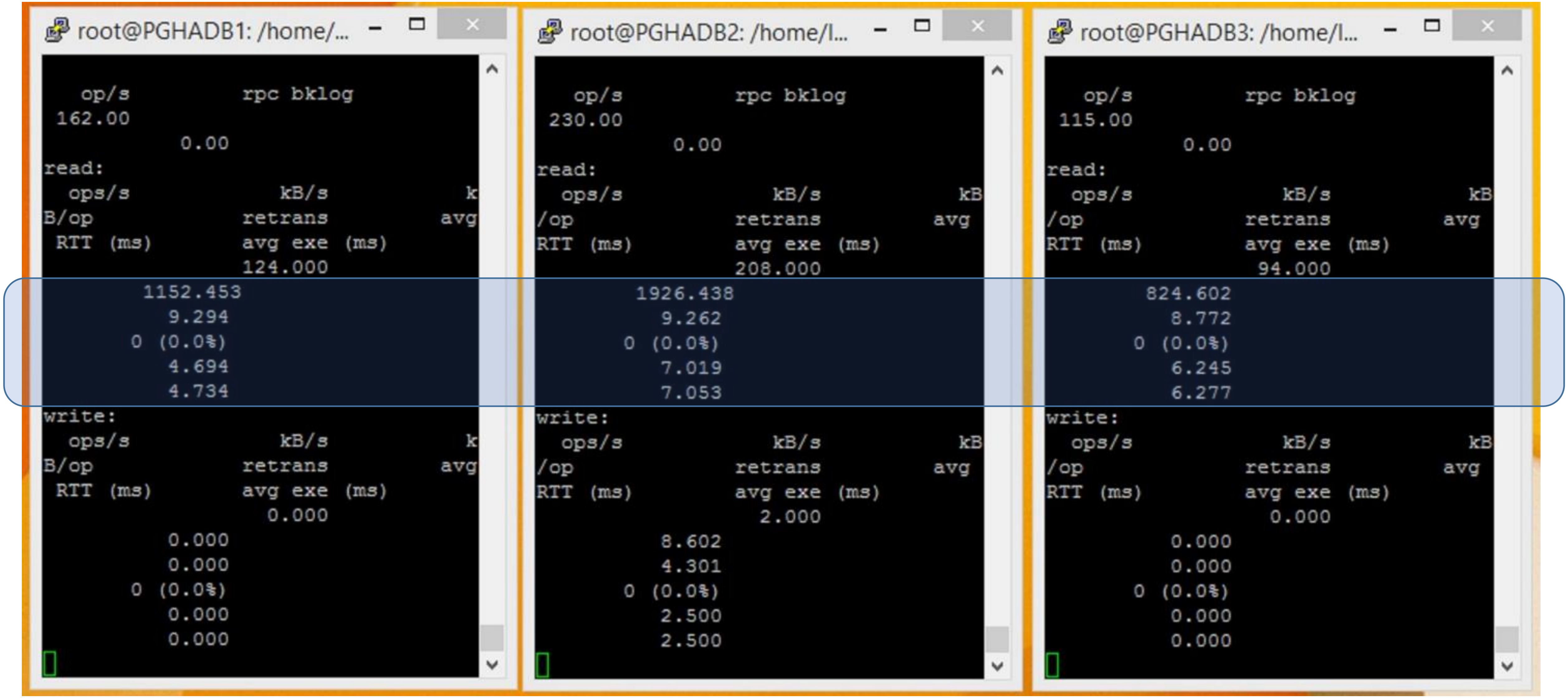
Keepalived-repmgr: Failover CPU usages



HAPROXY – PGBOUNCER THROUGHPUT

HTTP Request	Test duration	Avg. Response Time /sample	Throughput	KB/sec	Avg. Bytes /transaction	Avg. Latency /transaction
Read Only without data execution	423.901s	28.234s	2,359.041	662.477	287.565	28.228
Read Only with data execution	471.192s	28.215s	2,122.277	1,354.034	653.322	28.209
Simple Write with Inserts and Updates	702.484s	55.893s	1,423.520	1,895.078	1,363.212	55.886
Simple Write with Deletes	521.546s	36.755s	1,917.376	540.319	288.565	36.749
Read Write with Selects, Inserts and Updates	970.949s	77.564s	1,029.920	679.687	675.780	77.557
Read Write with Selects and Deletes	568.803s	42.116s	1,758.078	626.928	365.157	42.110

LOAD BALANCING



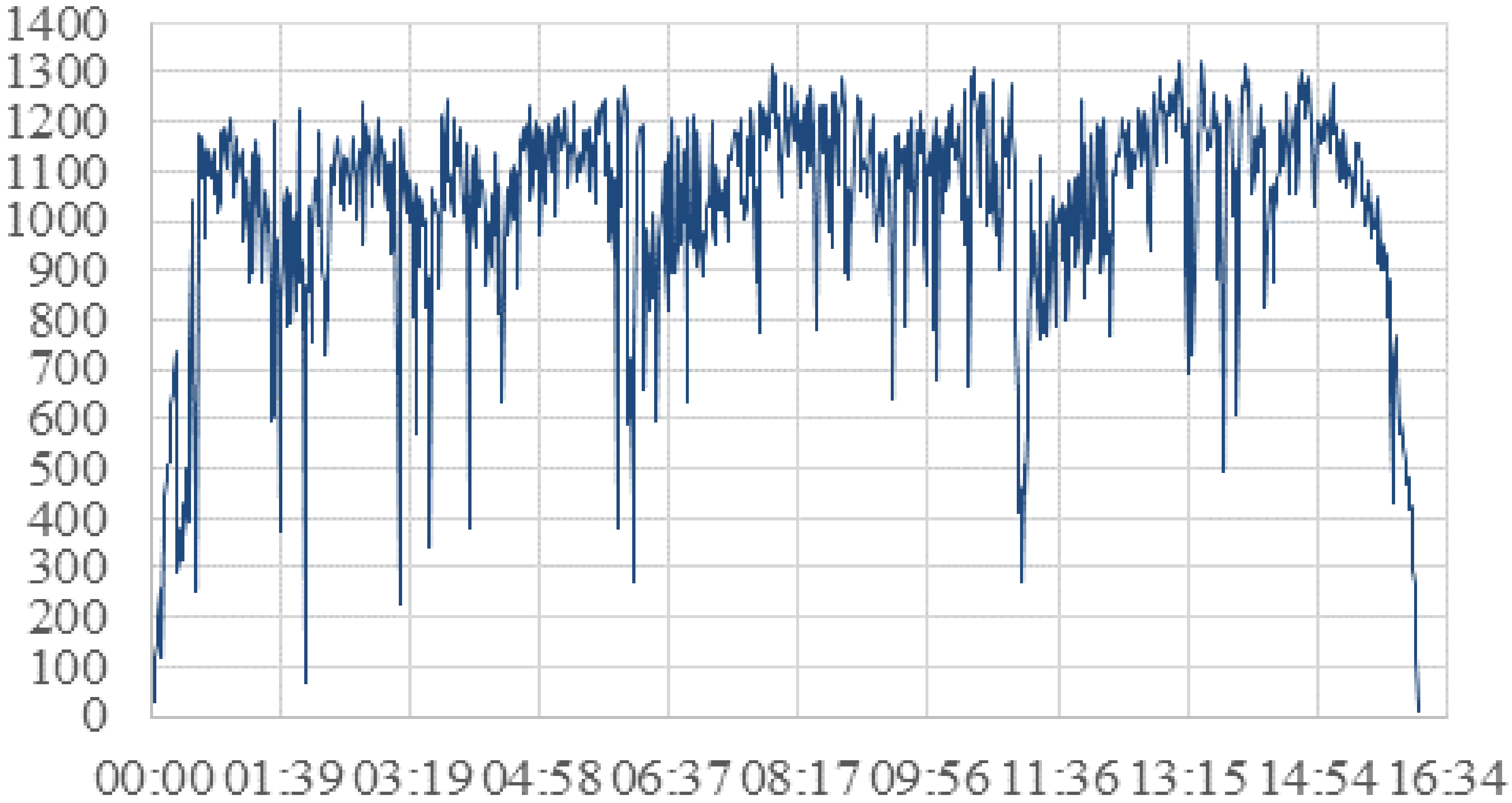
```
   op/s          rpc bklog
 162.00
           0.00
read:
  ops/s             kB/s          k
B/op            retrans         avg
 RTT  (ms)      avg exe  (ms)
                124.000
        1152.453
          9.294
       0 (0.0%)
          4.694
          4.734
write:
  ops/s             kB/s          k
B/op            retrans         avg
 RTT  (ms)      avg exe  (ms)
                  0.000
          0.000
          0.000
       0 (0.0%)
          0.000
          0.000
```

```
   op/s          rpc bklog
 230.00
           0.00
read:
  ops/s             kB/s          kB
/op             retrans         avg
RTT  (ms)       avg exe  (ms)
                208.000
        1926.438
          9.262
       0 (0.0%)
          7.019
          7.053
write:
  ops/s             kB/s          kB
/op             retrans         avg
RTT  (ms)       avg exe  (ms)
                  2.000
          8.602
          4.301
       0 (0.0%)
          2.500
          2.500
```

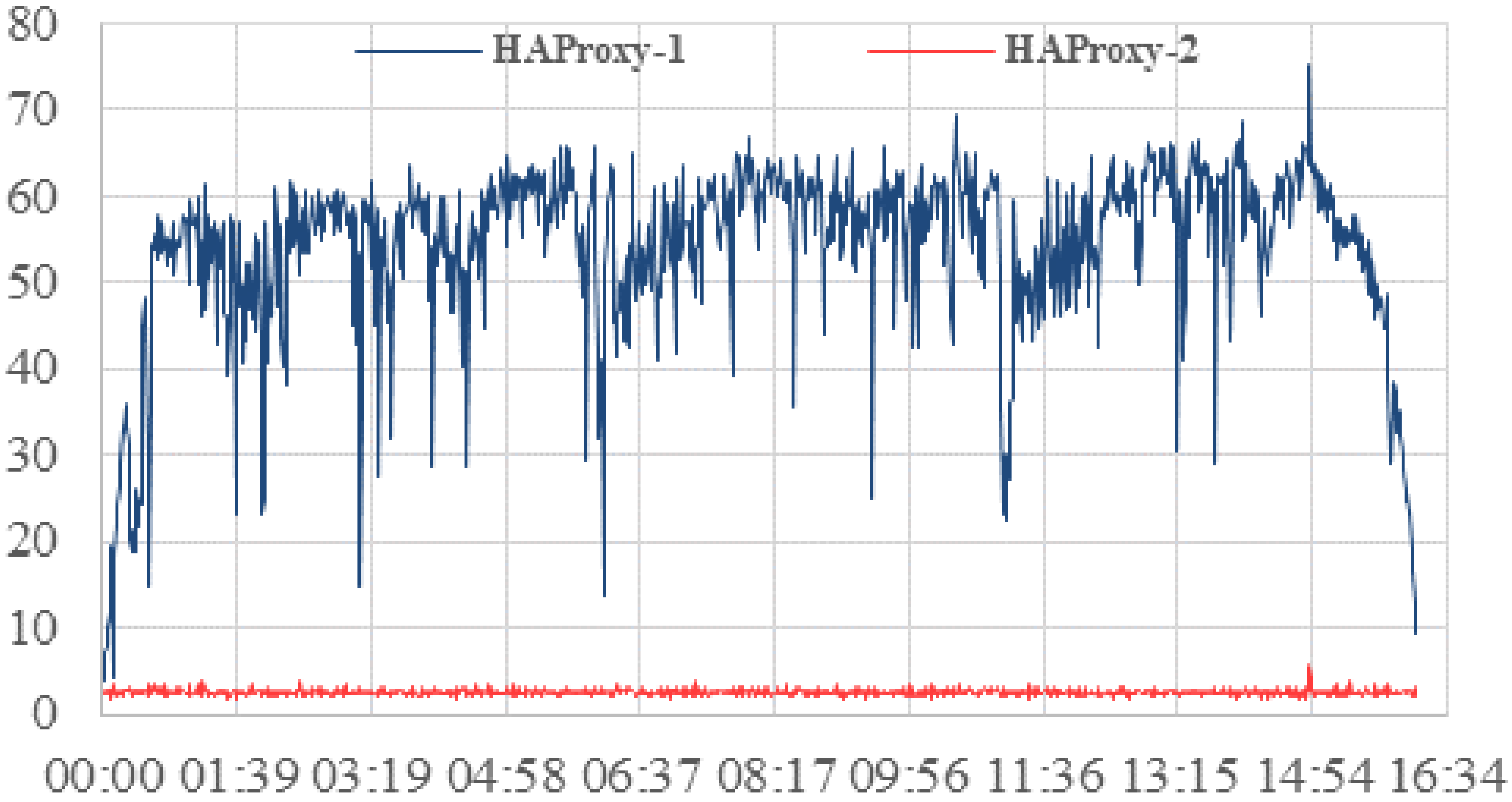
```
   op/s          rpc bklog
 115.00
           0.00
read:
  ops/s             kB/s          kB
/op             retrans         avg
RTT  (ms)       avg exe  (ms)
                 94.000
         824.602
          8.772
       0 (0.0%)
          6.245
          6.277
write:
  ops/s             kB/s          kB
/op             retrans         avg
RTT  (ms)       avg exe  (ms)
                  0.000
          0.000
          0.000
       0 (0.0%)
          0.000
          0.000
```

SYSSTAT is showing that the Reads are shared between Backend Master, Hot Standby and Async Standby

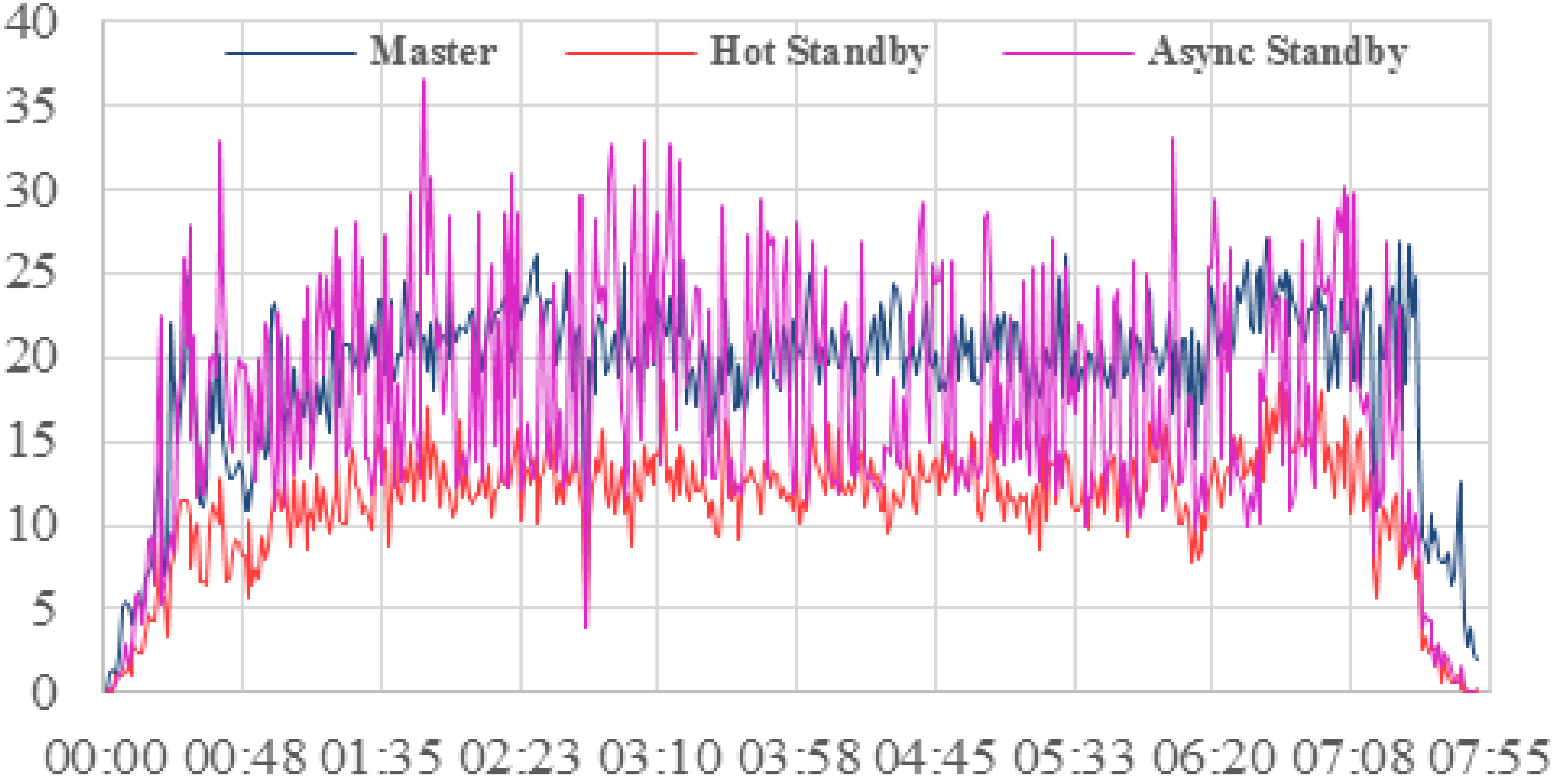
HAProxy-PgBouncer: Read Write with Selects, Inserts and Updates - Transactions per Second



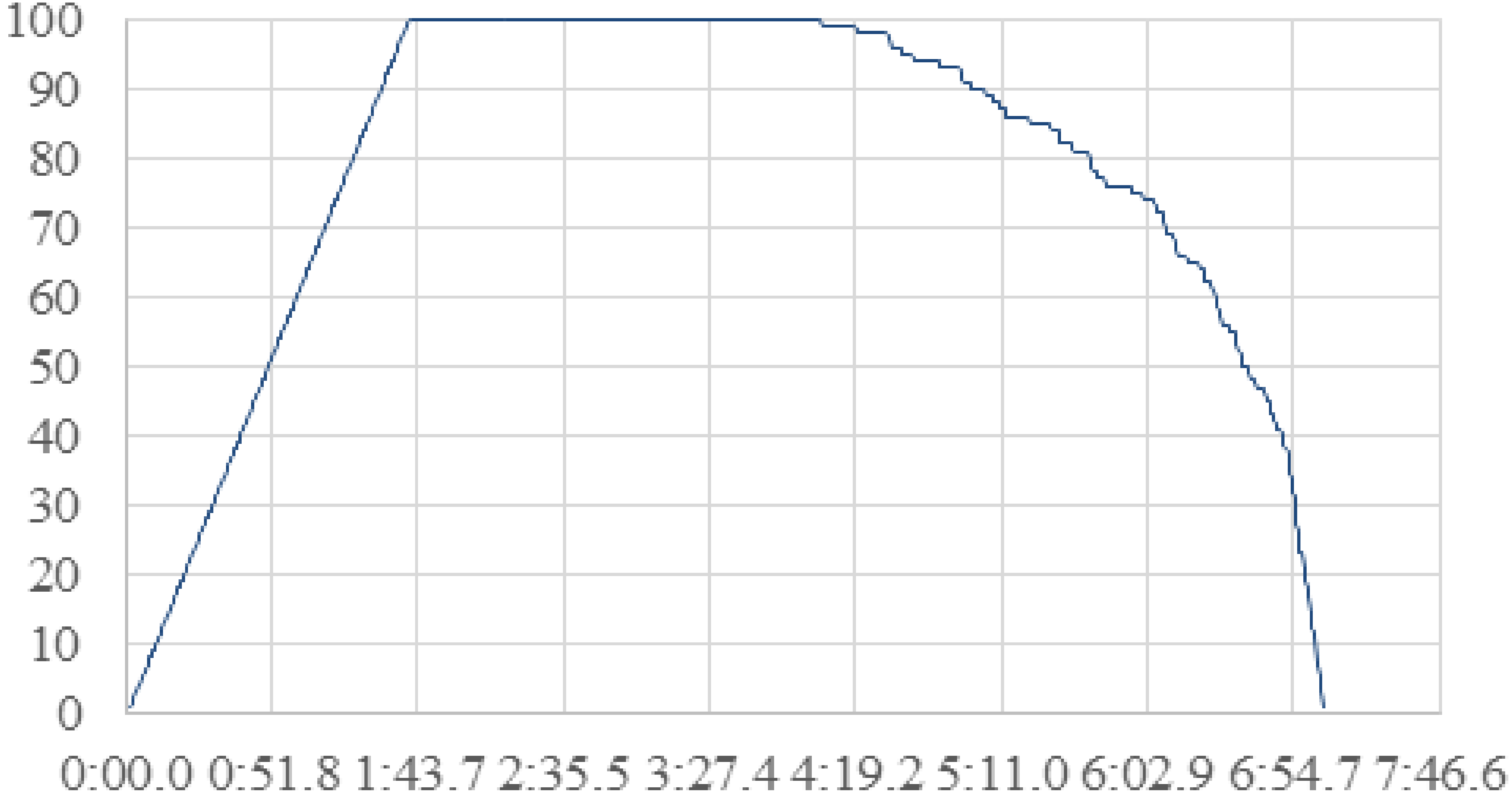
HAProxy-PgBouncer: Read Write with Selects, Inserts and Updates - Frontend CPU usages



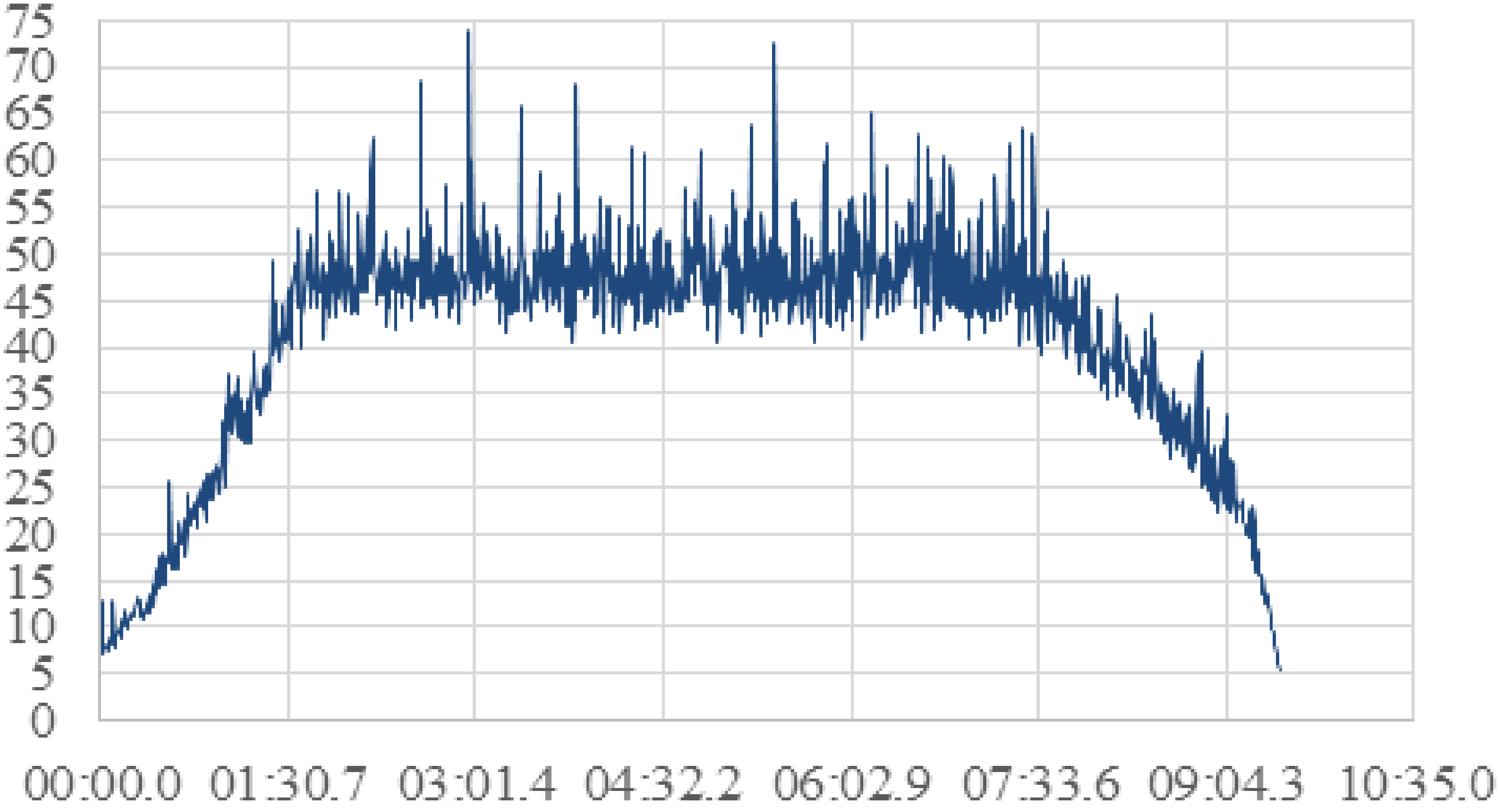
HAProxy-PgBouncer: Read Only with Data Execution - Backend database server CPU usages on Load Balancing



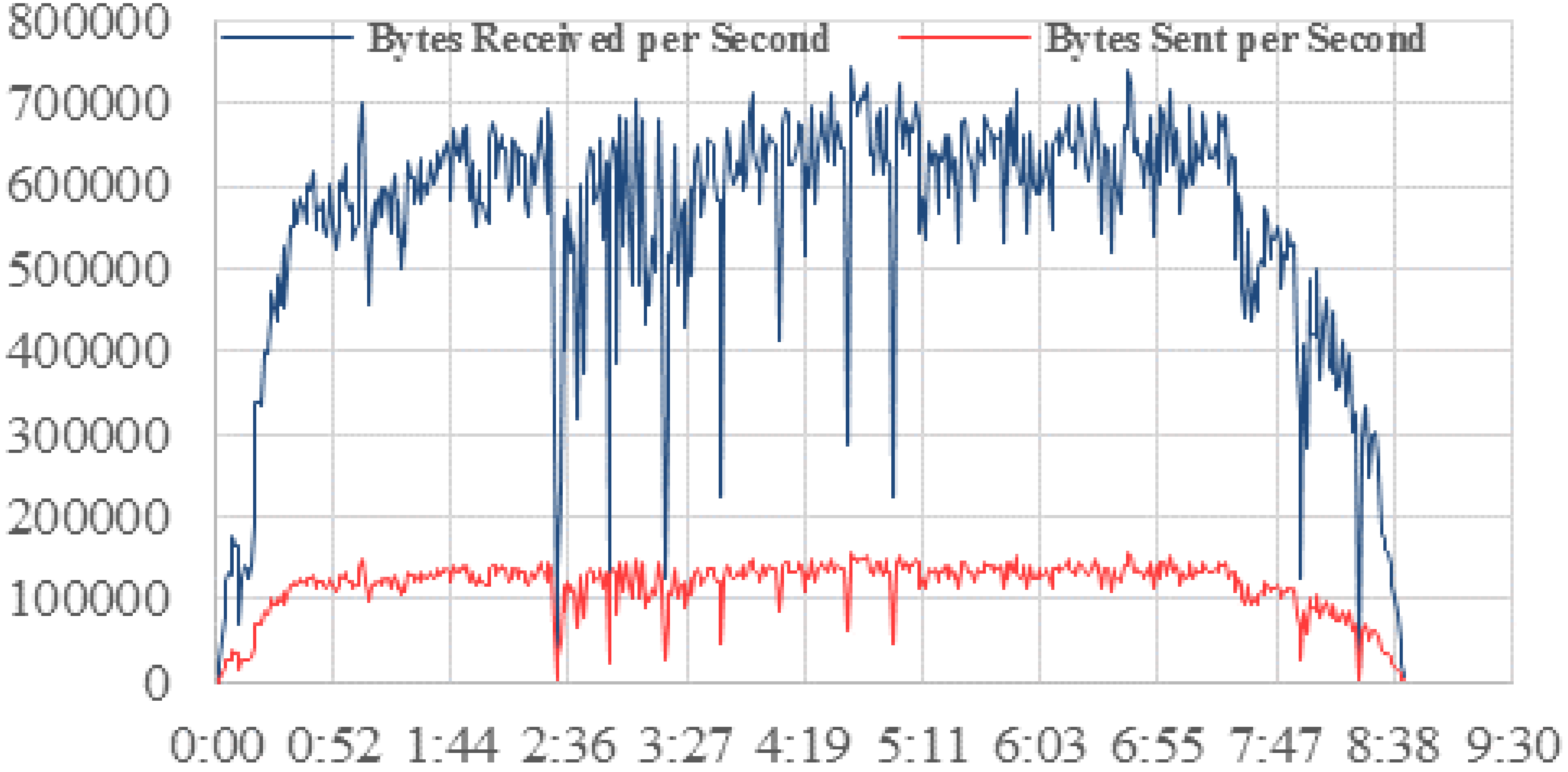
HAProxy-PgBouncer: Read Only without Data Execution - Active Threads



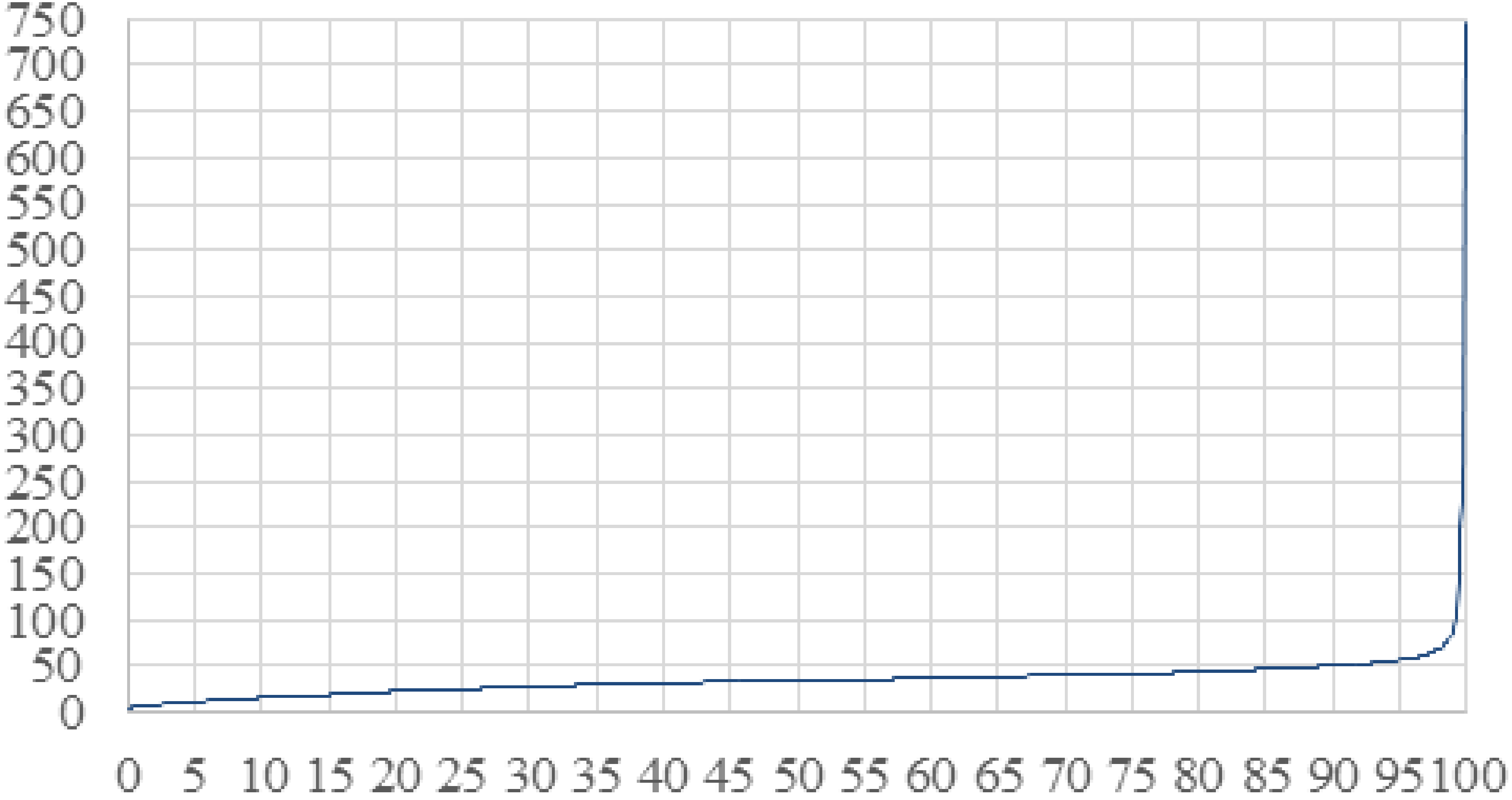
HAProxy-PgBouncer: Read Write with Selects and Deletes - Response Time



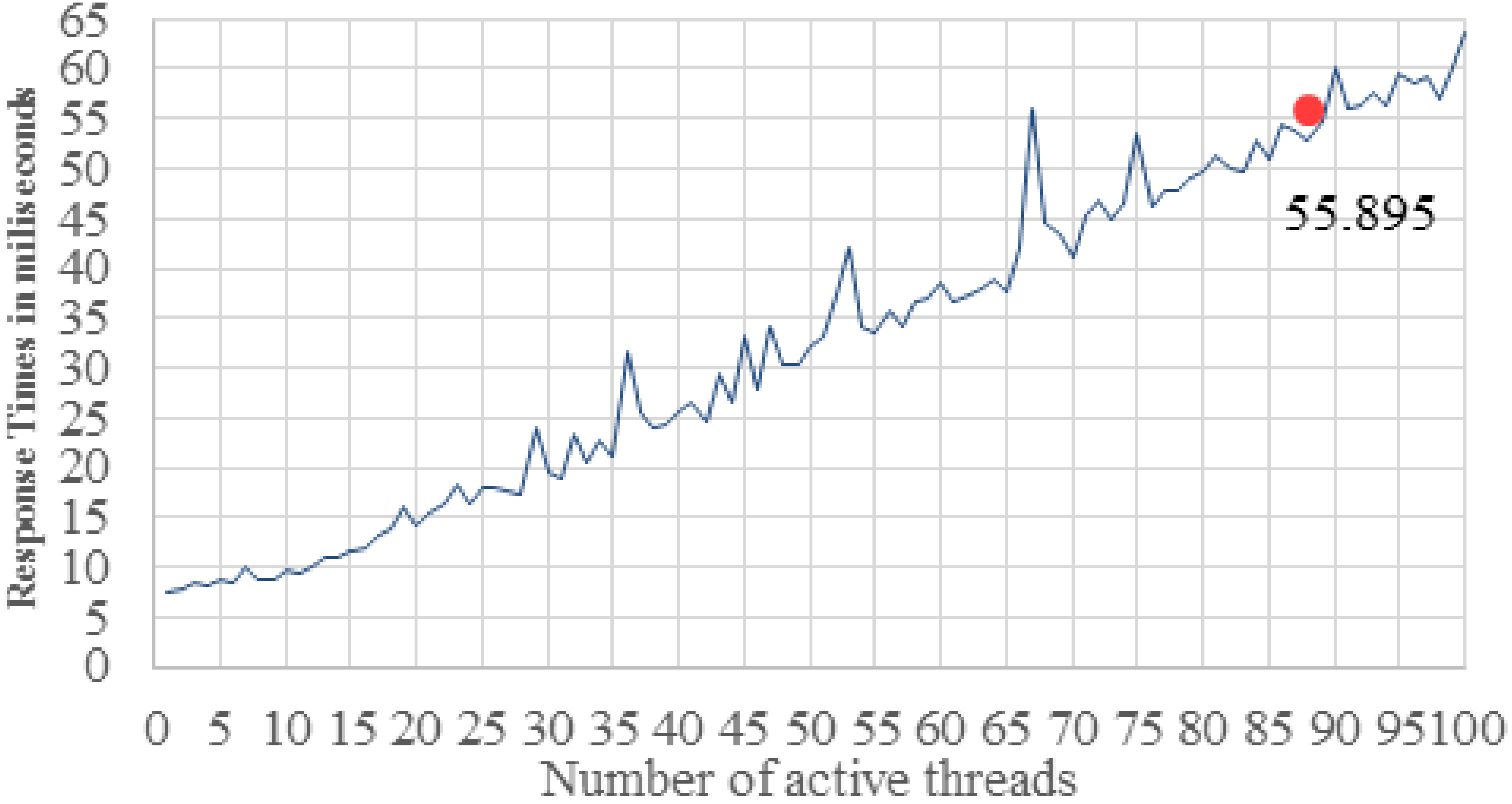
HAProxy-PgBouncer: Simple Write with Deletes - Bytes Throughput over Time



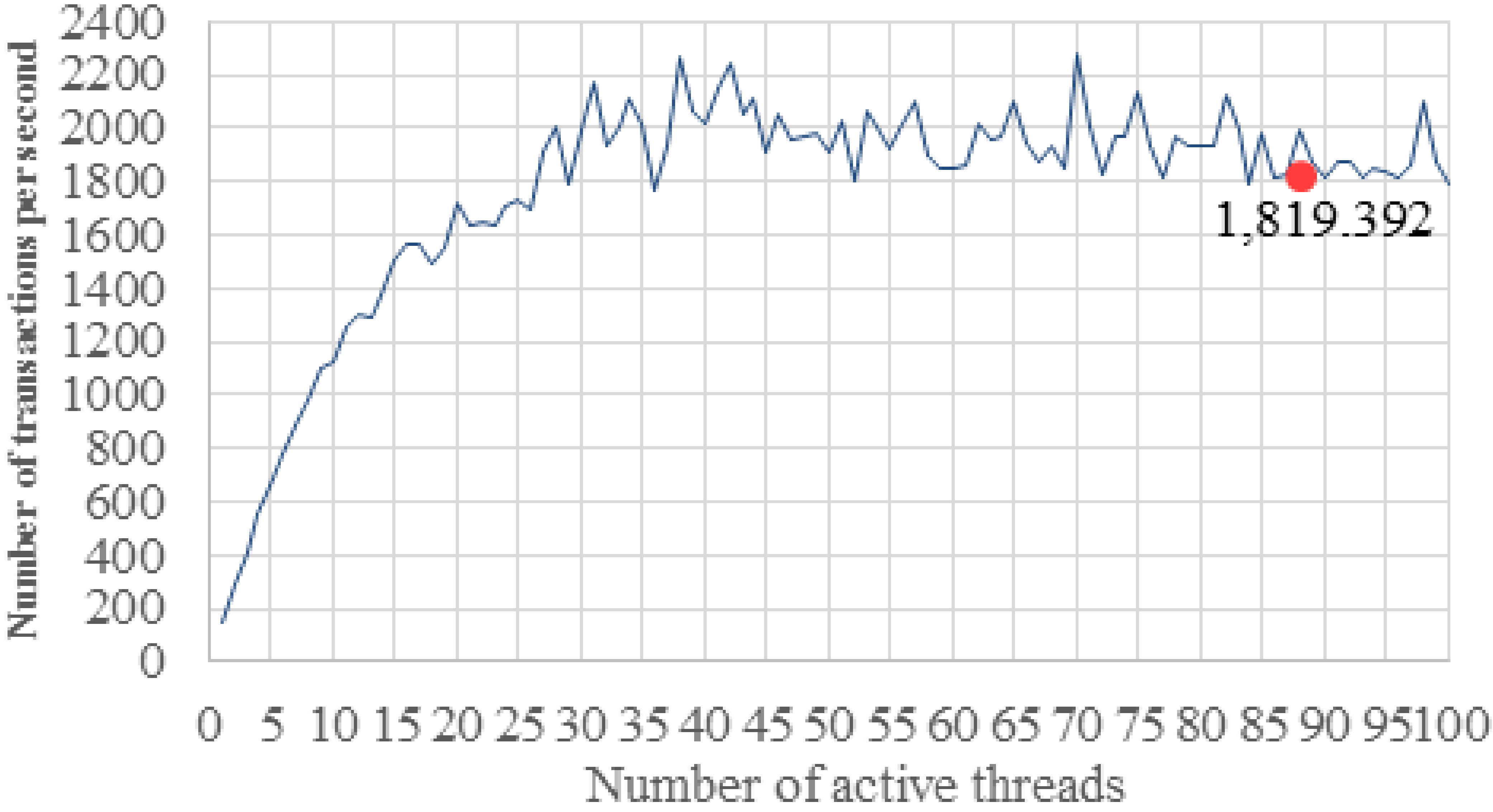
HAProxy-PgBouncer: Simple Write with Deletes - Response Times Percentiles



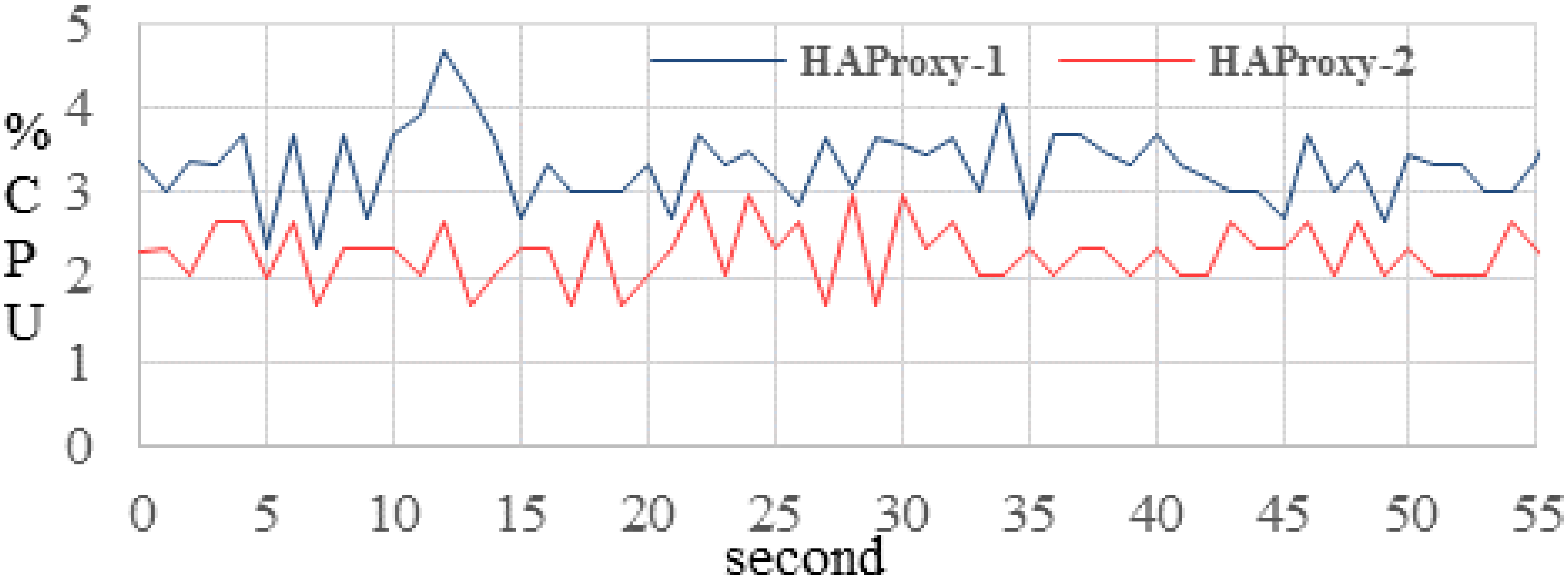
HAProxy-PgBouncer: Simple Write with Inserts and Updates - Response Times vs Threads



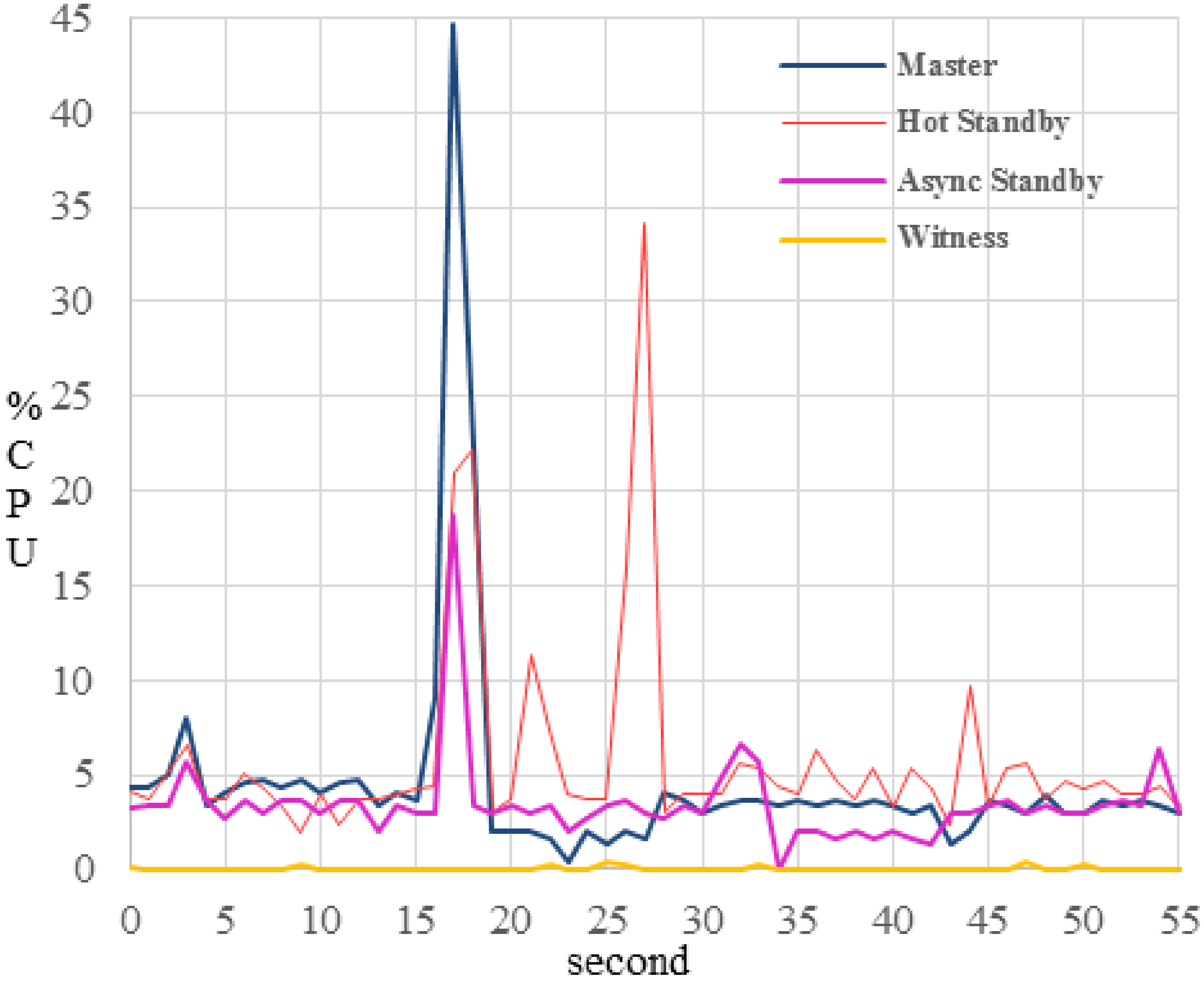
HAProxy-PgBouncer: Simple Write with Inserts and Updates - Transaction Throughput vs Threads



HAProxy-PgBouncer: Failover Front-end CPU usages



HAProxy-PgBouncer: Failover Back-end CPU usages



PERFORMANCE COMPARISON: KEEPALIVED-REPMGR vs. HAPROXY-PGBOUNCER

HTTP Request	Throughput improvement of HAProxy-PgBouncer
Read Only without data execution	9.454%
Read Only with data execution	1.609%
Simple Write with Inserts and Updates	7.266%
Simple Write with Deletes	2.220%
Read Write with Selects, Inserts and Updates	1.041%
Read Write with Selects and Deletes	0.346%

CONCLUSIONS

- HAProxy-PgBouncer cluster supplies good cross-containment approach
- The IPG, Database Group have achieved good purposes
  - ➢ Load Balancing
  - ➢ Farm Failover
  - ➢ Healthcheck
  - ➢ Auto-Failover
- Performance Analysis have been done by JMeter HTTP Requests combined with PhP using Fast CGI on Apache2
  - ➢ Read Only
  - ➢ Simple Write
  - ➢ Read Write
- HAProxy-PgBouncer improves the throughputs from 0.346% to 9.454% performance than keepalived-repmgr
- Keepalived-repmgr also does not offer cross-containment and load balancing abilities
- HAProxy-PgBouncer also provides two different methods to implement failovers: auto-failover and farm-failover