TemplateLoad Testing Summary Report

Date: 25.02.2018
Prepared: Anton Serputko

Summary

Load testing of Solution_name solution was divided into several testing phases. The main **goals** were:

- **Implement** load testing **framework** for current load/capacity testing and future regression performance testing
- Determine **capacity** of **app** servers
- Determine **performance** of **app** servers under the load
- Determine **capacity** and performance of **smoke** functionality
- **Optimize** cluster **configuration** to handle more concurrent clients

Configuration: cluster was deployed on **t2.medium** aws instances.

Load testing **framework** is published on **github** repository:
All necessary **info** about framework usage and tests execution could be found in **Readme**.

General results

App server performance:

Operation_name processing **throughput**:

Average- **423 per second**

Max- 432 fps per second

Operation_name **processing time** under load:

90% of requests are processed **under 100ms**

Performance **degradation** starts **after ~53 requests per second** load

Flow_name flow performance:

Average **throughput** on current configuration is **~423 requests per second**

Under the load of **50 concurrent users** 5% of requests are executed **60+ seconds**. That causes **Gateway_timeout errors**.

More information could be found below.

App server results

For testing app server was used next scenario:

- Load generator **sends requests** to server with smoke scenario.

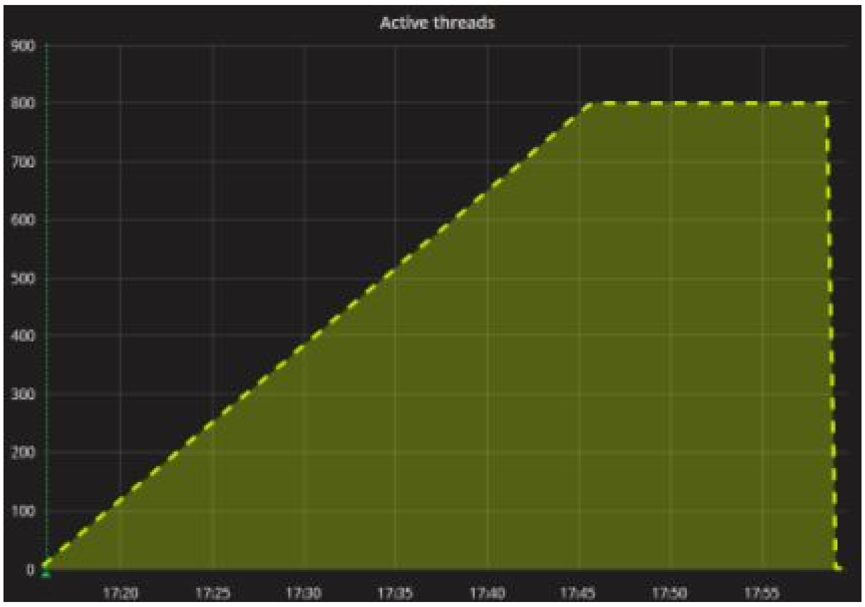
Summary:

Scenario provides load with total **planned** number of concurrent **threads**(virtual users) - **800**. Number of users was increasing linearly for first 30 mins of test(rampup), after that load generator was holding 800 vu for 15 mins.

Load model:

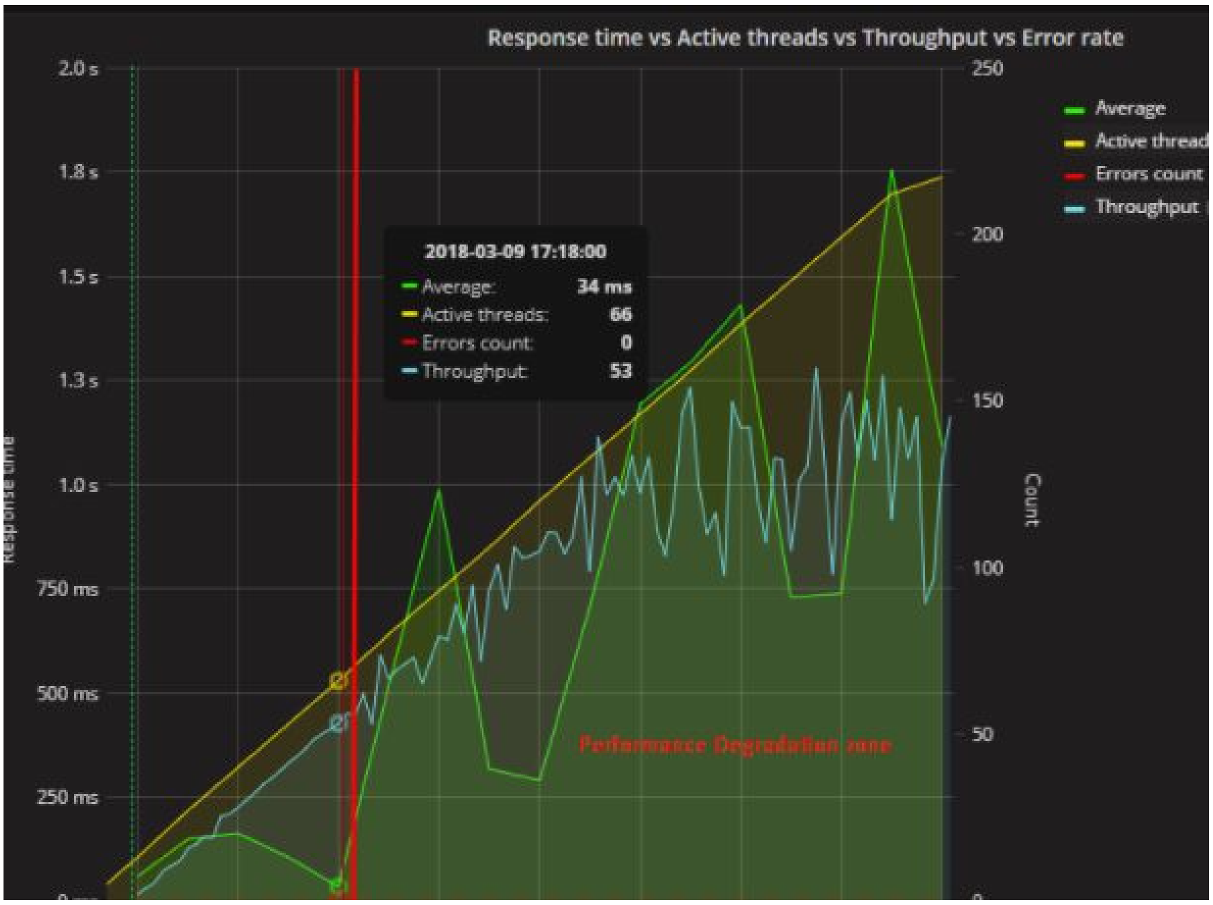
Number of virtual users: 800
Rampup period: 1800s(30m)
Test duration: 2600s(43m)

Number of active threads trend



Performance degradation starts **after** 66 active threads(**53 requests per sec**). This means that if users will concurrently send less than 53 requests to app server it will handle that load without increasing request processing time. In other words server has enough resources to handle such load.

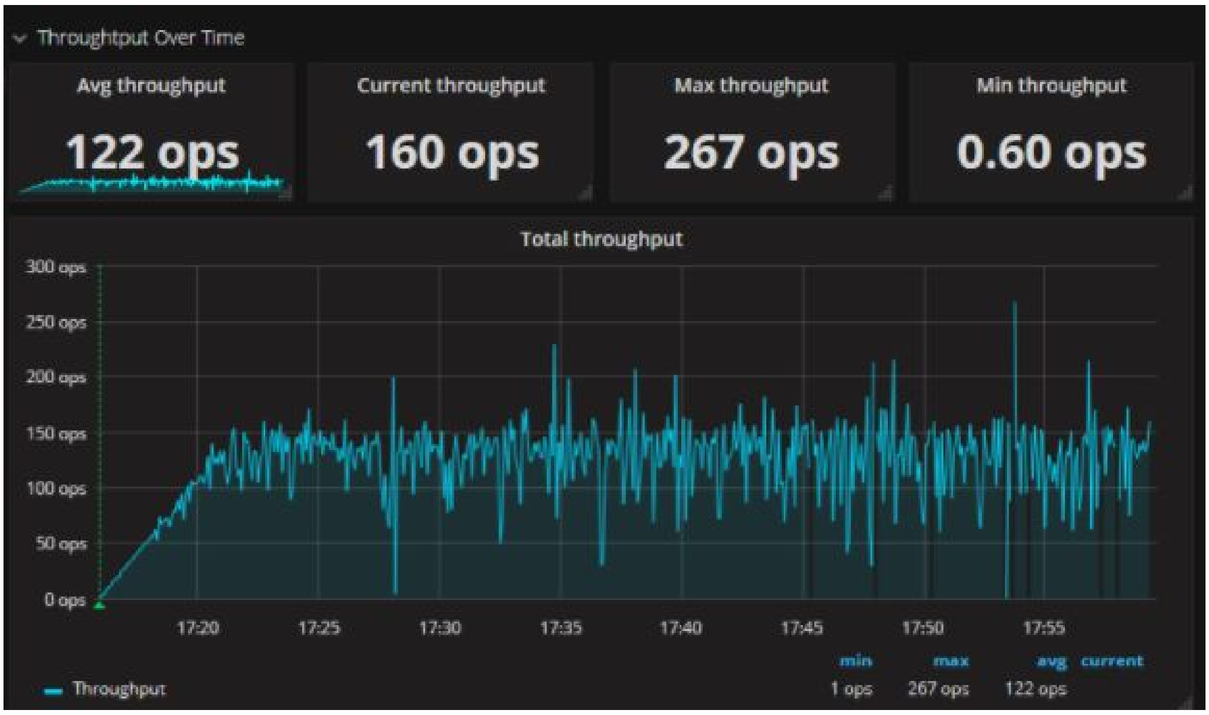
Performance degradation point



Current configuration of system could provide next **throughput**:

	Throughput in rps
Average	122
Max	267

Throughput trends

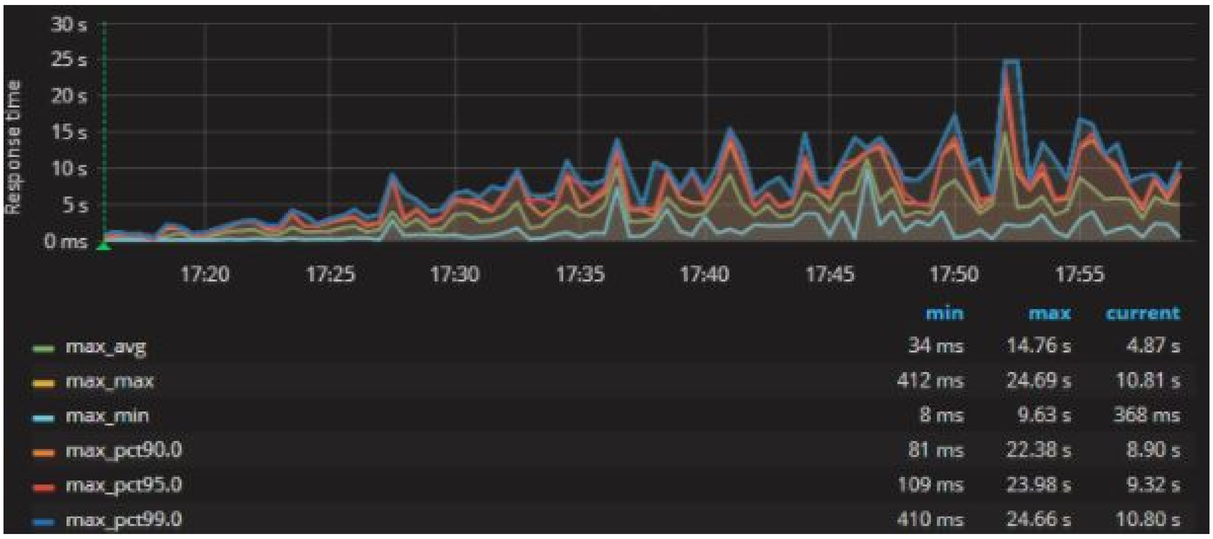


processing time on app server

Under load app server in current configuration of **cluster processes 90% of requests under 5s**, 4% of emails from 12s to 16s and 1% of request are processed more than 16s

	Average	90-th percentile	95-th percentile	99-th percentile	Min	Max
request	3s	5s	12s	16s	123ms	39s

Request processing time trends

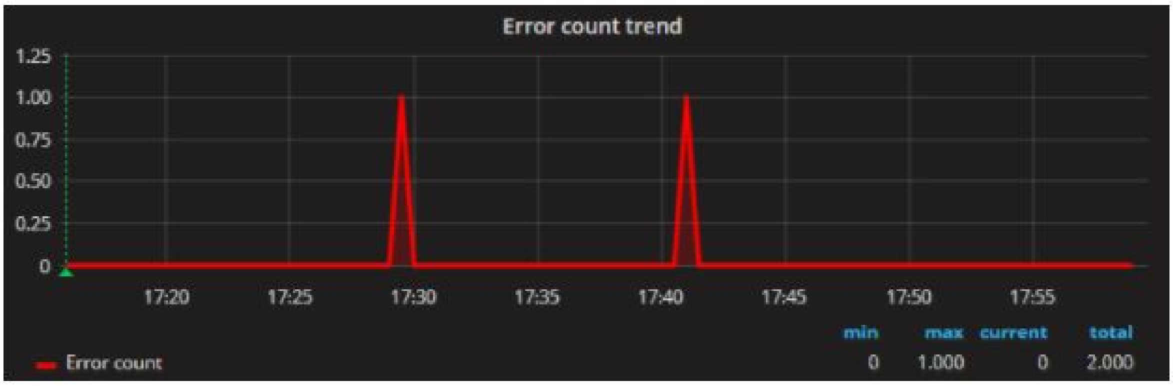


	min	max	current
max_avg	34 ms	14.76 s	4.87 s
max_max	412 ms	24.69 s	10.81 s
max_min	8 ms	9.63 s	368 ms
max_pct90.0	81 ms	22.38 s	8.90 s
max_pct95.0	109 ms	23.98 s	9.32 s
max_pct99.0	410 ms	24.66 s	10.80 s

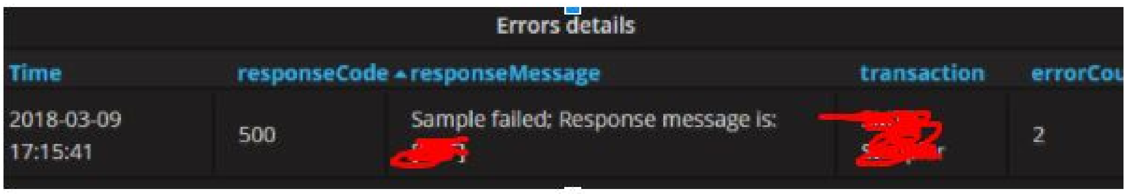
Note: Do not pay attention to series label prefix ‘max’. For example max_avg means just avg, etc

While running test there were few [Internal server error] errors returned from app Server.

Error count trend



	min	max	current	total
Error count	0	1.000	0	2.000

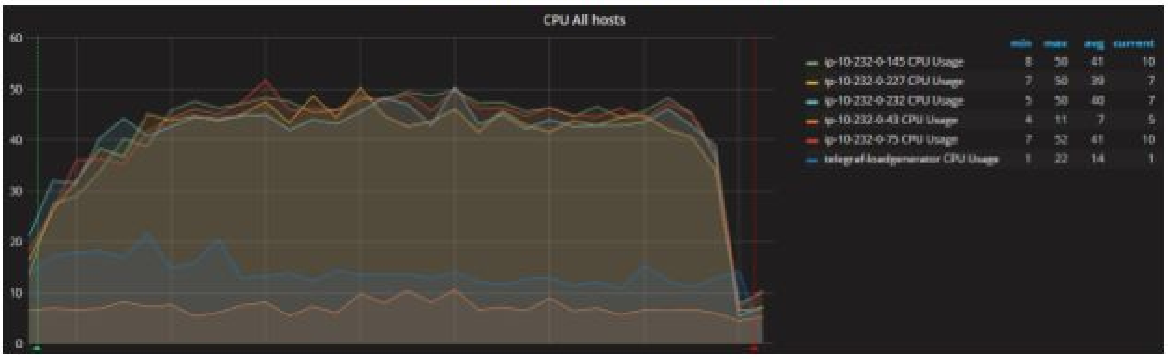


Errors details

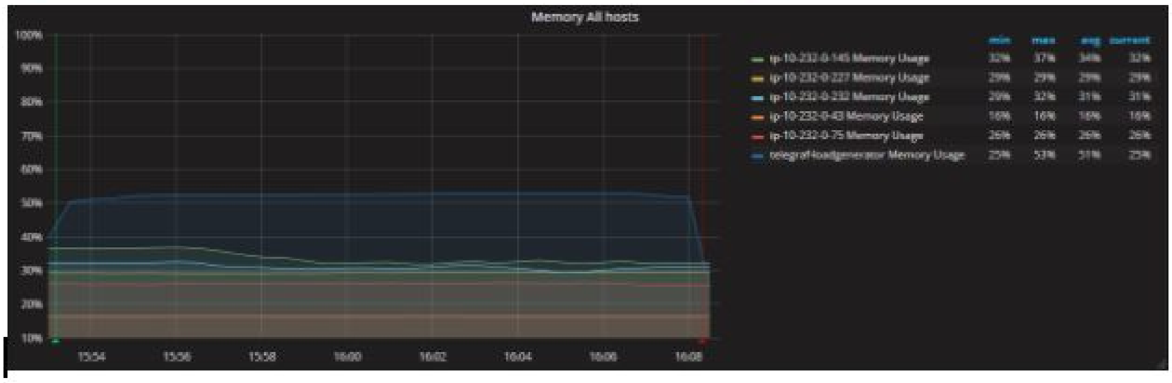
Time	responseCode	responseMessage	transaction	errorCou
2018-03-09 17:15:41	500	Sample failed; Response message is: [redacted]	[redacted]	2

CPU/RAM utilization on aws

[illegible]



	min	max	avg	current
ip-10-232-0-145 CPU Usage	8	50	41	10
ip-10-232-0-227 CPU Usage	7	50	39	7
ip-10-232-0-232 CPU Usage	5	50	40	7
ip-10-232-0-43 CPU Usage	4	11	7	5
ip-10-232-0-75 CPU Usage	7	52	41	10
telegraf-loadgenerator CPU Usage	1	22	14	1



	min	max	avg	current
ip-10-232-0-145 Memory Usage	32%	37%	34%	32%
ip-10-232-0-227 Memory Usage	29%	29%	29%	29%
ip-10-232-0-232 Memory Usage	29%	32%	31%	31%
ip-10-232-0-43 Memory Usage	16%	16%	16%	16%
ip-10-232-0-75 Memory Usage	26%	26%	26%	26%
telegraf-loadgenerator Memory Usage	25%	53%	51%	25%