Harvard India Policy Insights

Load and stress test report

OVERVIEW

In this report we present test results of load and stress tests made in the Backend component of the application, which have been carried out by using JMeter.

We used QA environment for these tests, which has half of the resources of the base infrastructure suggested for production.

SCENARIOS

Average use of services / testing / development

Where it simulates an average of 10 users making requests to the server in 1 second intervals, which should respond quickly and not require a large use of resources. Determining the expected behavior in a development environment.

Useproductive

The normal workload that is expected to have once it is in a productive environment. 50 concurrent users are scheduled. In order to measure response times and determine possible improvement options.

Saturation

Set an overkill load to see at what point it breaks, slows down, or stops responding to requests.

AVERAGE USE OF SERVICES / TESTING / DEVELOPMENT

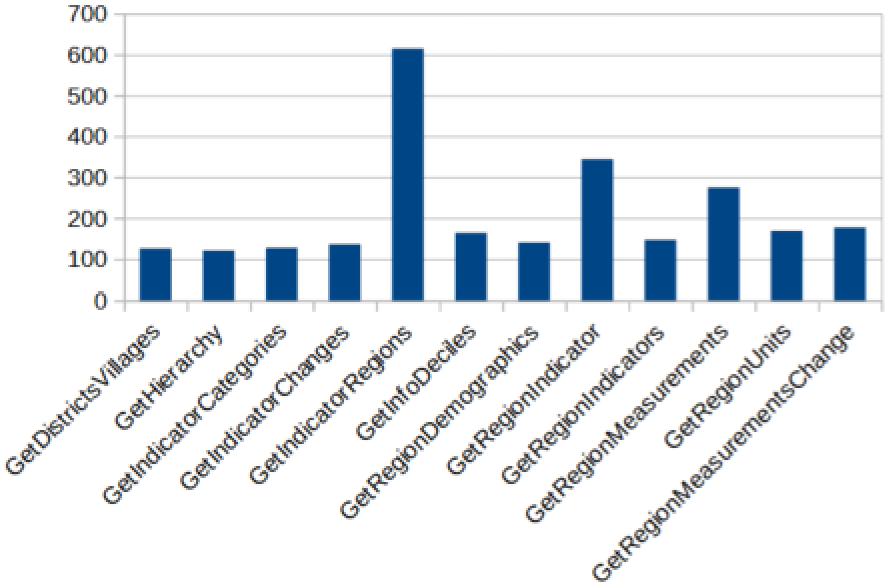
The tests were carried out with static requests, taking into account that a test and development environment was simulated, with less load and stress than the other scenarios.

Summary of Results

Test threads (seconds - repetitions)	10 (1 - 1)
HTTP Requests	12
Test duration	6 s
Average time per transaction	211,75 ms
total transactions	120
failed transactions	0
correct transactions	120

response time graph

In the graph it can be seen, reading from left to right, that the most expensive requests are ***GetIndicatorRegions*** and ***GetRegionIndicator,*** the first with an average of 615 milliseconds and the second with 344 milliseconds, in response times.



Datagraphed

Request	**Time average (ms)**
GetDistrictsVillages	126
GetHierarchy	121
GetIndicatorCategories	127
GetIndicatorChanges	136
GetIndicatorRegions	615
GetInfoDeciles	164
GetRegionDemographics	140
GetRegionIndicator	344
GetRegionIndicators	147
GetRegionMeasurements	275
GetRegionUnits	169
GetRegionMeasurementsChange	177
Total	**211.75**

Transaction distribution

In the following distribution of intervals per second, the requests are grouped by average response time, through which it can be observed that the transactions have been executed in less than 1 second.

Average response time	**Requests**
0-1 sec	GetDistrictsVillages GetHierarchy GetIndicatorCategories GetIndicatorChanges GetIndicatorRegions GetInfoDeciles GetRegionDemographics

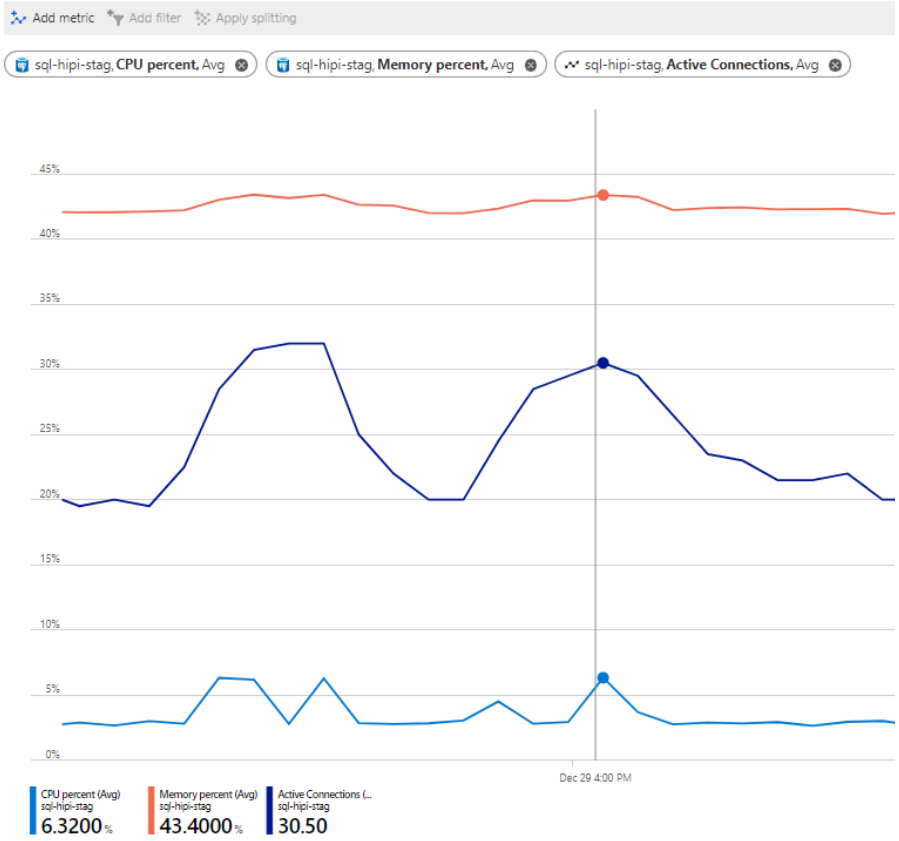
	GetRegionIndicator GetRegionIndicators GetRegionMeasurements GetRegionUnits GetRegionMeasurementsChange
1-2 seconds	
2-3 seconds	
more than 3 seconds	

Impact on Azure DataBase

At the beginning In the test plan there were other active connections to Azure, but based on the average number of connections during the test run, the impact of those additional connections is irrelevant.

In the following graph you can see:

1. The number of connections at the time the test started went from 28.5 to 30.5.
2. CPU usage went from 2.91% to the highest peak during the run, 6.32%.
3. Memory usage went from 42.96% to the highest peak during the run of 43.4%.



PRODUCTIVE USE

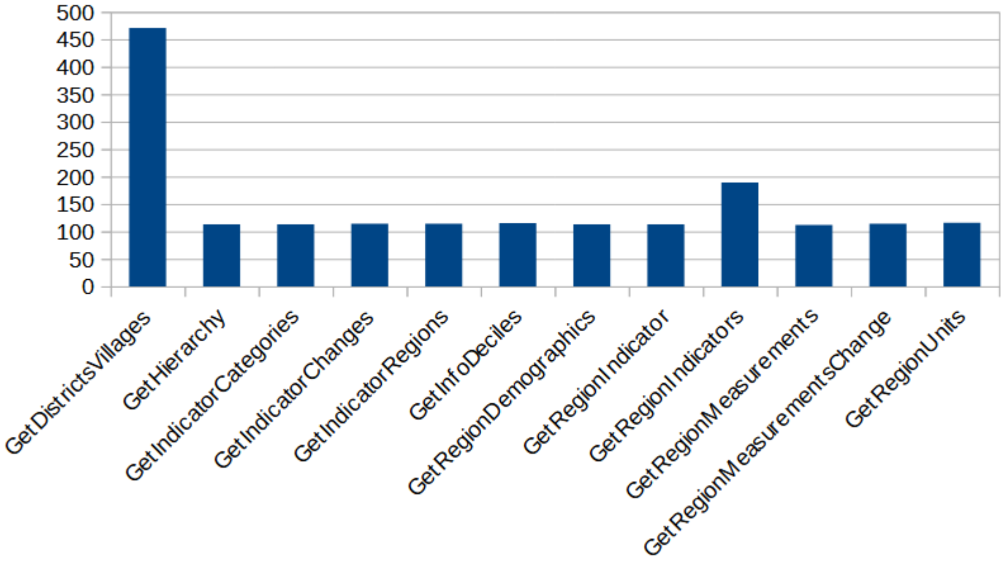
The tests were carried out with requests with dynamic variables, taking into account that a productive environment was simulated, in a scenario where any data, correct or incorrect, can be sent in the HTTP request.

Summary of Results

Test threads (seconds - repetitions)	50 (1 - 1)
HTTP Requests	12
Test duration	2 s
Average time per transaction	149,7 ms
total transactions	600
failed transactions	0
correct transactions	600

response time graph

In the graph it can be seen, reading from left to right, that the most expensive requests are***GetDistrictVillages*** and***GetRegionIndicators,*** the first with an average of 471 ms and the second with 189 ms, in response times.



Datagraphed

Request	Time average (ms)
GetDistrictsVillages	471
GetHierarchy	113
GetIndicatorCategories	113
GetIndicatorChanges	114
GetIndicatorRegions	114
GetInfoDeciles	115
GetRegionDemographics	113
GetRegionIndicator	113
GetRegionIndicators	189
GetRegionMeasurements	112
GetRegionUnits	114
GetRegionMeasurementsChange	116
Total	**149.75**

Transaction distribution

In the following distribution of intervals per second, it can be seen that none of the endpoints took more than 1 second to respond.

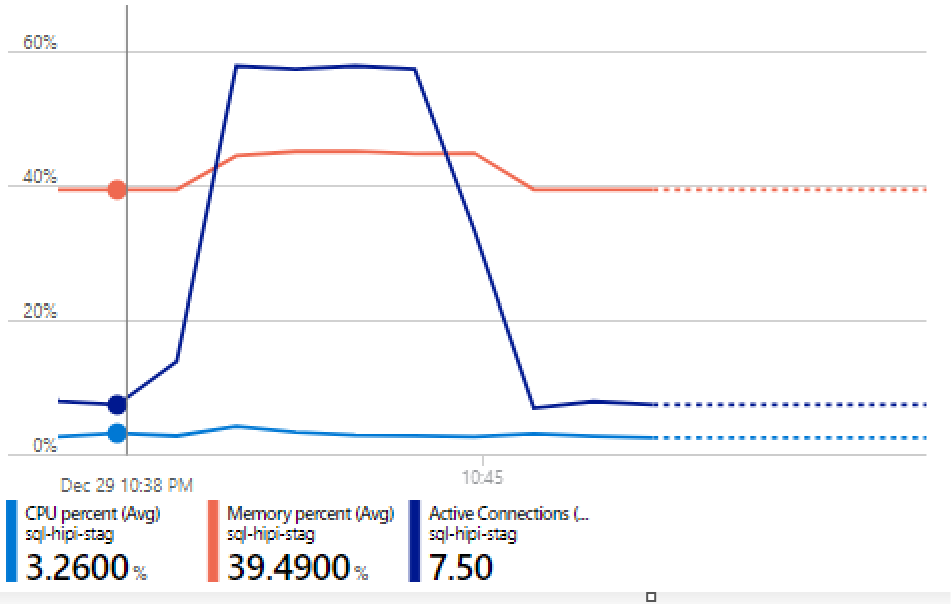
Average response time	Requests
0-1 sec	GetDistrictsVillages GetHierarchy GetIndicatorCategories GetIndicatorChanges GetIndicatorRegions GetInfoDeciles

	GetRegionDemographics GetRegionIndicator GetRegionIndicators GetRegionMeasurements GetRegionUnits GetRegionMeasurementsChange
1-2 seconds	
2-3 seconds	
more than 3 seconds	

Impact on Azure DataBase

In the following graph you can see:

4. The number of connections at the time the test started went from 7.5 to 58.
5. CPU usage went from 3.26% to the highest peak during the run, 4.32%.
6. Memory usage went from 39.49% to the highest peak during the run of 45.24%.



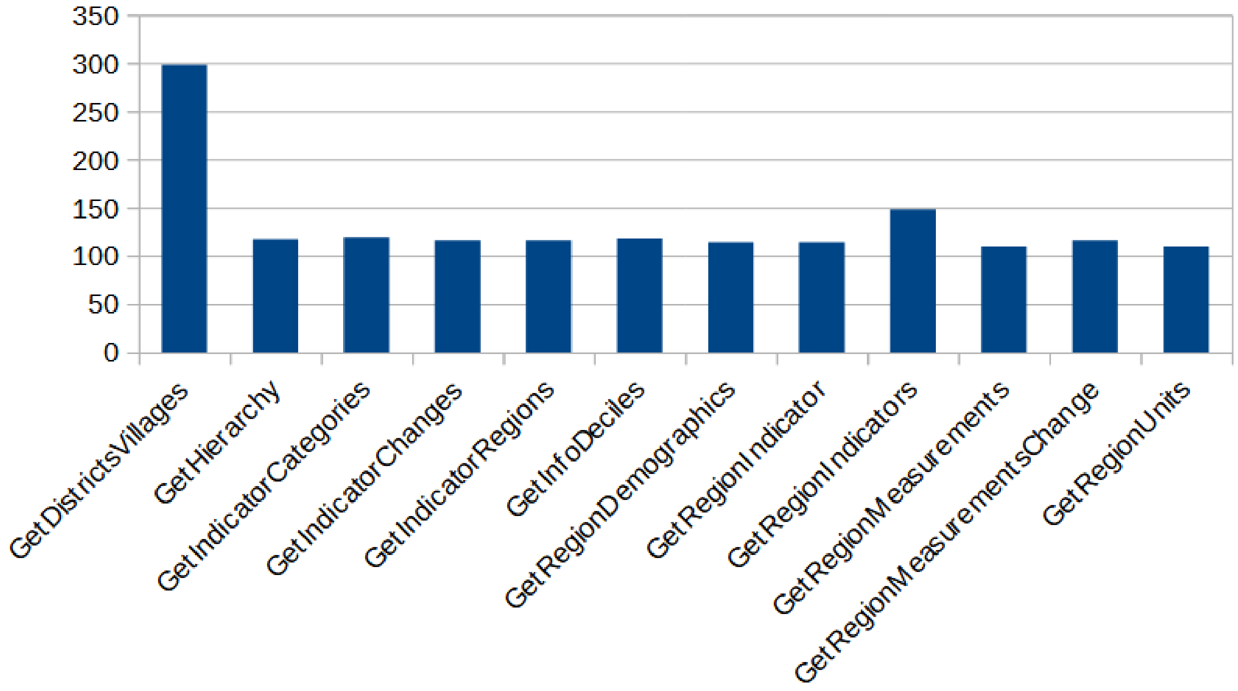
SATURATION

The tests were performed with requests with dynamic variables. In a scenario where any data, correct or incorrect, can be sent in the HTTP request.

Summary of Results

Test threads (seconds - repetitions)	100 (1 - 2)
HTTP Requests	12
Test duration without error	4 s
Test duration	4s
Average time per transaction without error	133 ms
Average time per transaction	133 ms
total transactions	2400
failed transactions	0
correct transactions	2400

response time graph



Datagraphed

Request	Time average (ms)
GetDistrictsVillages	298
GetHierarchy	117
GetIndicatorCategories	119
GetIndicatorChanges	116
GetIndicatorRegions	116
GetInfoDeciles	118
GetRegionDemographics	114
GetRegionIndicator	114
GetRegionIndicators	148
GetRegionMeasurements	110
GetRegionUnits	116
GetRegionMeasurementsChange	110
Total	**133**

Transaction Distribution

In the following distribution of intervals per second, no transaction exceeded one second to return results.

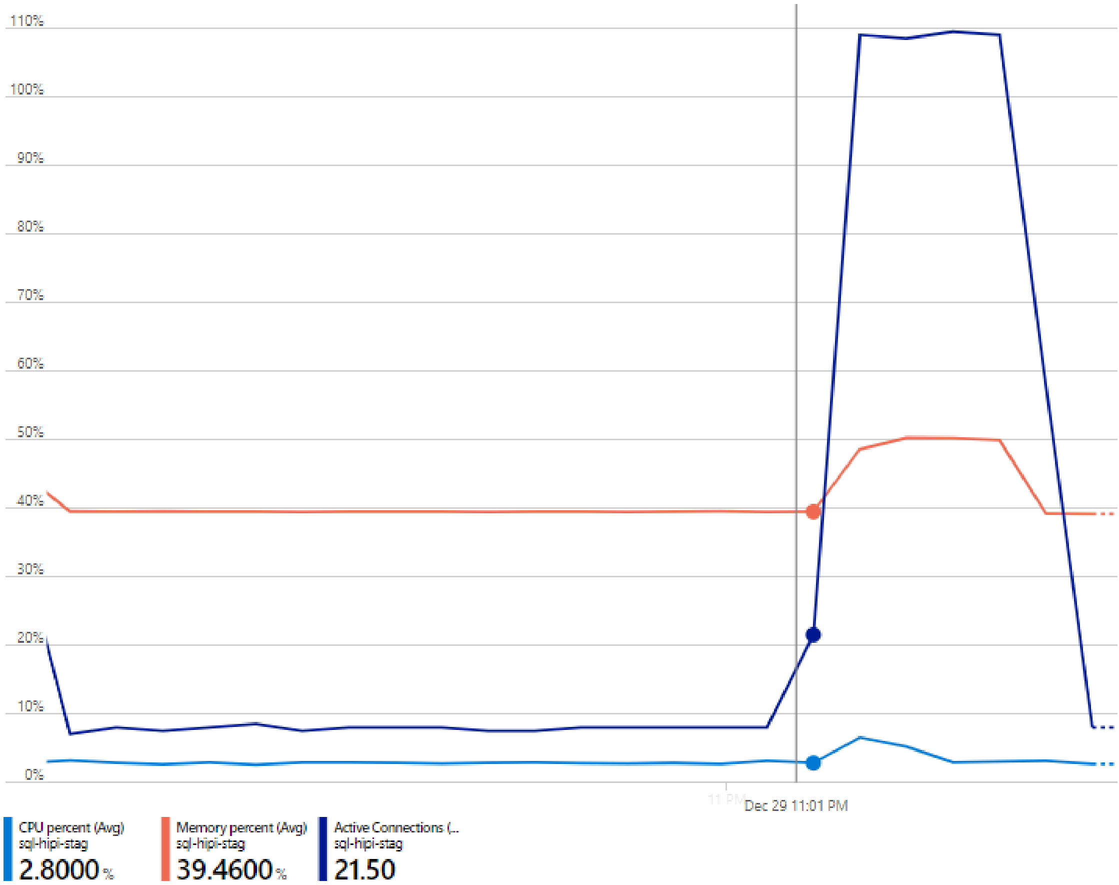
Average response time	Requests
0-1 sec	GetDistrictsVillages GetHierarchy GetIndicatorCategories GetIndicatorChanges GetIndicatorRegions GetInfoDeciles

	GetRegionDemographics GetRegionIndicator GetRegionIndicators GetRegionMeasurements GetRegionUnits GetRegionMeasurementsChange
1-2 seconds	
2-3 seconds	
more than 3 seconds	

Impact on Azure DataBase

In the following graph you can see:

7. The number of connections at the time the test started went from 8 to 109.5.
8. CPU usage went from 2.8% to the highest peak during the run, 6.5%.
9. Memory usage went from 39.46% to the highest peak during the run of 50.24%.



Harvard India Policy Insights

Load and stress test report

CONCLUSIONS

- The Response times between each of the scenarios are similar.
- The API is prepared for significant and prolonged load.