Performance Testing Course : Report Analysis Technique

Author: Hien HOANG

Agenda

- Test result techniques
- Basic analysis.

WHIWH Rule

- WHIWH rule helps to lead bottleneck identification in correct direction which saves time.
- Usually performance testing window in the project plan is smaller than functional testing window and near to Go-Live date.

⇒ Being a performance tester you must have to complete the testing in that short span of time and not only completing the testing but need to identify bottlenecks as well.

WHIWH Rule

- W : ***W****hat errors come*
- H: ***H****ow many errors come*
- I: ***I****n which condition errors come*
- W: ***W****hen errors come*
- H: ***H****ow long errors persist*

What errors come

During the performance test you may get different types of error. To move analysis in the correct direction, first you need to categorize those errors.

a. *Scripting errors:*

- Missed to replace old URLs in some places: This causes failure of a particular transaction.
- Lack of test data: Set "Recycle at EOF" = false leading either failure of the transaction or user exit.

b. *LG side errors:*

- LG failed during the test: Make sure given no. of LGs can handle user threads during the test.

c. *Client-side errors:*

- HTTP 4XX: It indicates that something wrong is sent to the server, which server did not understand.

d. *Server-side errors:*

- HTTP 5XX: Such status codes fall under the server side issue.
- Slow response from server: Server is unable to handle large no of requests => It impacts on transaction response time which may lead "Timeout".

e. *Network error:*

How many errors come

% of error has its own importance in performance testing. Some systems have error tolerance limit like 3-5%, although some banking system have 0% tolerance. In that case, performance test result needs to analyse thoroughly. There should not be any SLA breach. It is advisable that do not consider near about the result when you have 0% error tolerance.

In which condition errors come

You need to carefully analyse the condition in which errors come whether they are occurring during ramp-up, steady state or in a pro-long test.
Generally,

- Errors during ramp-up period are due to the incapability of server to handle gradually increasing load.
- Errors during steady state are due to queue pile-up, server unavailable, network bandwidth issue.
- Errors in pro-long test (Endurance Test) are due to a memory leak

Please note that above are general error causes, it could be different in your test result, so better to analyse the result and then stamp on the cause.

Tips to resolve: Analyse the server log with the help of developer to identify the exact bottleneck.

When errors come

The exact time helps you to identify the cause of bottleneck while rendering in server logs. You can merge error graph with other graphs to see what are the impact on the test due to a particular error at a particular time. The developer expects a summarised report in which exact error timings are mentioned. This helps them to save their time and resolve the issue as quickly as possible.

How long errors persists

You need to check whether those errors last for a small duration or longer duration. The causes of small duration (one peak) error may be due to user exit, wrong test data for some of the entries, the server failed and recovered quickly (you can see a span of error), back-end server activity due to shared test environment etc. The causes of longer duration error may be server failure, DB failure, queue pile-up, response timeout, LG failure or network issue in communicating with LG.

ComCorDEEPT Rule

"***ComCorDEEPT***" Rule is very helpful to carry out performance test result analysis in a proper direction to identify accurate bottlenecks in the short time. This rule comprises 7 analysis methods, which help to understand the graphs more deeply, identify the correct bottleneck and decision making on the test result. These methods totally depend on graphs and their analysis, which will help you to conclude the test result after finding actual bottlenecks. These methods are:

- 1. Compare (Com)
- 2. Correlate (Cor)
- 3. Drilling (D)
- 4. Eliminate (E)
- 5. Extrapolate (E)
- 6. Pattern (P)
- 7. Trends (T)

Compare

This is the first and foremost method of Performance Test Analysis. As per Performance Test Life Cycle, you gather requirement from the client and agreed on SLA (Service Level Agreement). SLA is a mutual agreement between Client and you (Vendor) on Performance Metrics like No. of Users, Response Time, TPS, etc. which you capture in Performance Test Plan and compare with result metrics. This is a very simple method to conclude a result as pass or fail.

If results are under the SLA acceptance limit, then you can easily give GREEN sign-off (with some recommendation, if required) else you will ask the developer to tune the application so that the SLA can be met.

Compare(continued)

How will this method work when SLA are not predefined?

In such case, you have to apply the baseline and benchmark approach. First, execute a test on old (existing) code version and set baseline metrics, then deploy the latest code and execute the test with the same load and compare the result with baseline test result. Likewise, you can apply to compare method in the absence of predefined SLA and provide sign-off as I mentioned above.

How does this method help in analyzing the bottleneck?

The comparison of client and server performance metrics helps to understand the acceptable thresholds for the system and its resources and figure out what is significantly different. The places where the differences appear are the areas to look for bottlenecks.

Correlate

In Apache JMeter, you can open result file (.csv/.log/.jtl) in composite graph listener to use this feature.

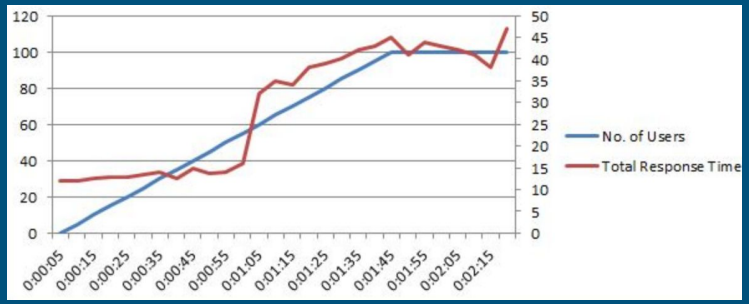
Graph correlation is about establishing the relationship between performance metrics by comparing the data. The software system is always complex with multi-tier architecture, different technologies and interfaces with internal or external systems. So the easy way to do correlation is to compare the end user performance with the server side metrics. It is like comparing the trend from one set of metrics like response time with other metrics like web server CPU utilization.

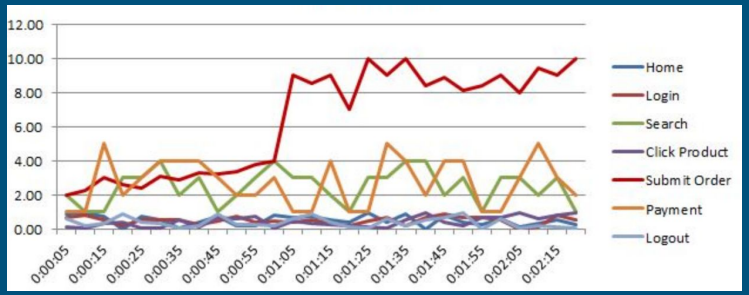
Drill down

This method is used to concentrate on one bottleneck at a time and narrow down the focus to a component by drilling down till you get exact root cause. It is pretty much like starting your investigation at one point and drilling down further to see what exactly is the problem. This method can be applied to any type of components or any metrics. This method also involves comparison and correlation methods.

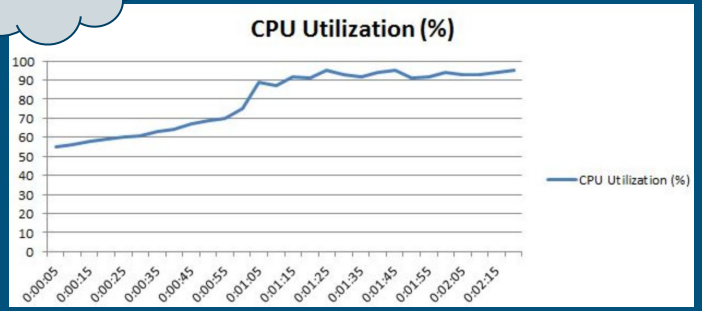
Drill down

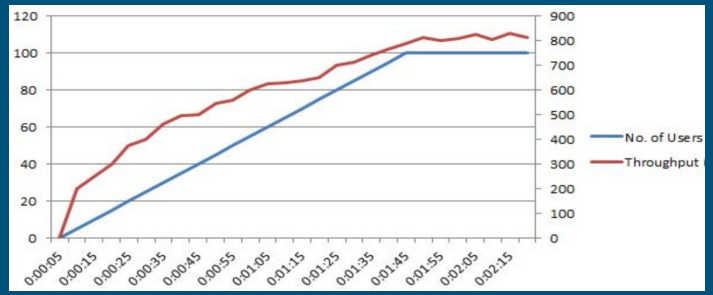
Example:





Voila!





Eliminate

This method leads to remove certain components from the list of culprits and focus on others to identify the bottleneck. But why? It has been observed that around 20%-30% newly build systems/applications have many performance issues. Some are really critical and some are having least priority to resolve.

This method is evolved because developers cannot concentrate on all the defects at the same time to resolve them and provide proper fixes in short timelines. Some of the fixes are again re-generated the old defects. To narrow down the percentage of bottlenecks with a quick and quality solution we use Eliminate method.

This is a method where you can use your cognitive skills and as innovative as possible to decide which bottleneck is critical and which can be parked for some time (until next release).

Extrapolate

Performance test result extrapolation is required when an application is tested for a small number of users (as per test server capacity) and need to extend the result as per production servers. The extrapolated results are also used to predict the scaling-up of the environment for a larger number of users.

Many clients do not have 100% scaled performance test environment. They either ask to execute the test in 50% scaled environment or using one instance of the server. In such case, you need to extrapolate the result with some techniques and calculation.

Tips:

- Linear Extrapolation can be applied for Throughput, Hits per seconds, TPS, Java Heap Size etc.
- S-curves or Mixed mode technique is used to predict extrapolation for Response Time, Latency, CPU Utilization and Memory Utilization.

Pattern

It is a method of understanding the pattern of the graph, analysing the graph and then conclude the result. This method is divided into two parts:

1. Analyse Pattern and Conclude

2. Compare Pattern and Conclude

Analyse Pattern and Conclude

Best example to understand this method is SOAK TEST(or ENDURANCE TEST).

Generally, Endurance test is executed for 6-8 hours, but do you think any application runs only 6-8 hours in production and then we switch-off the application?

So, how 6 to 8 hours test help us to predict whether an application has any bottleneck or not?

Compare Pattern and Conclude

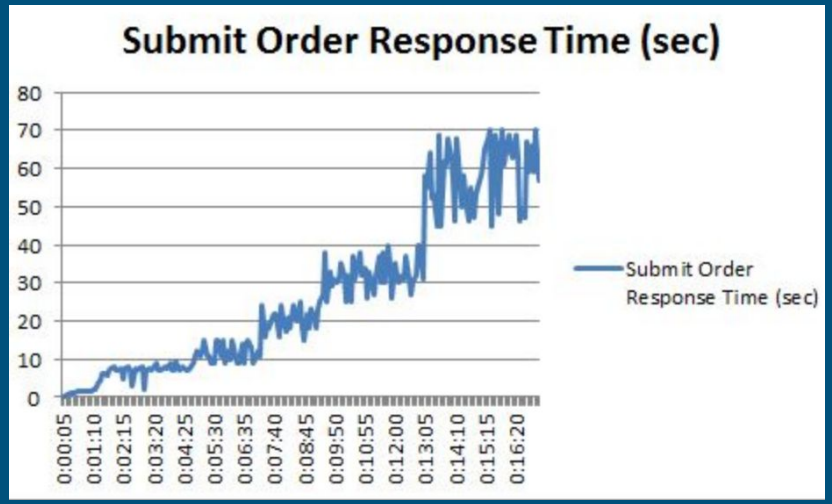
We compare the performance issues in the system under analysis with commonly seen issues and its causes in other systems.

There are so many common performance issues like one instance of the server is not configured properly in the load balanced environment resulting in regular spikes in the response time, the huge spike in response time in a stable system due to web server cache refreshes and even gradual increase in response time due to memory leaks.

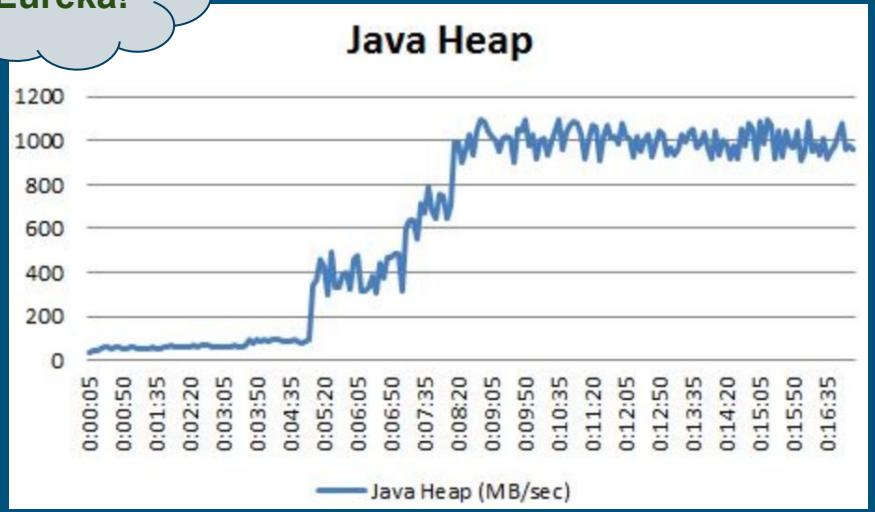
The basic is to compare the performance issues with commonly seen issues and its root causes. In this way, it is easy and fast to jump on and focus on the commonly seen components.

Compare Pattern and Conclude

Example:



Eureka!

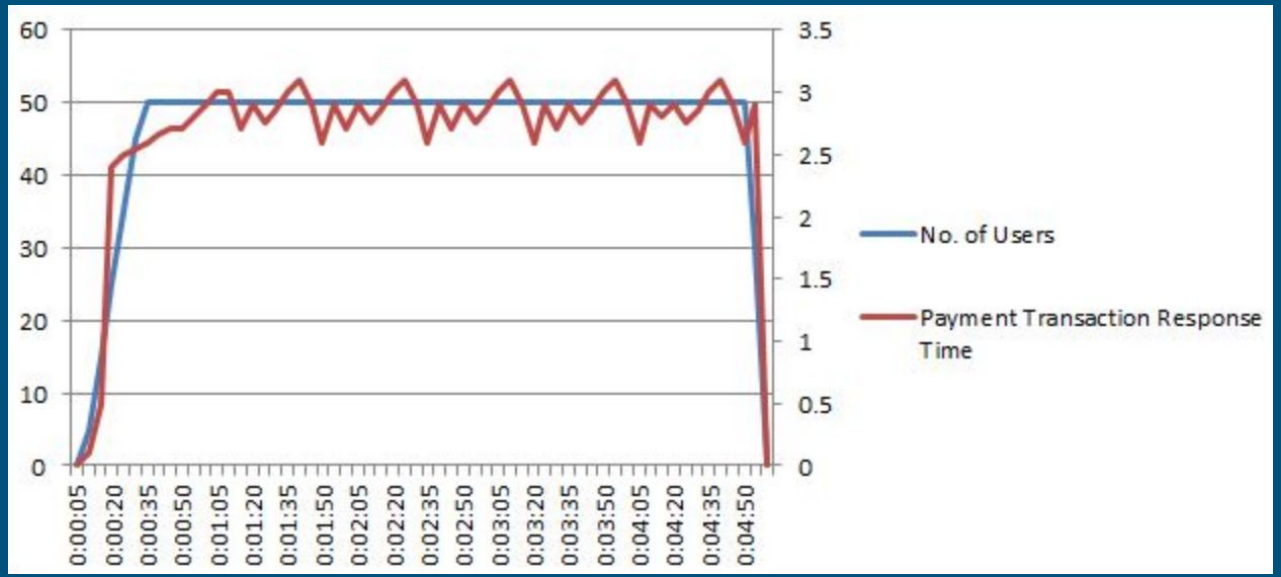


Trends

As per the definition of Trend, it is “*a general direction in which something is developing or changing*”. Performance trends can be of different types like consistent, fluctuate, increase and decrease. Trend analysis can be done on client, server and network metrics.

Trends - Some examples

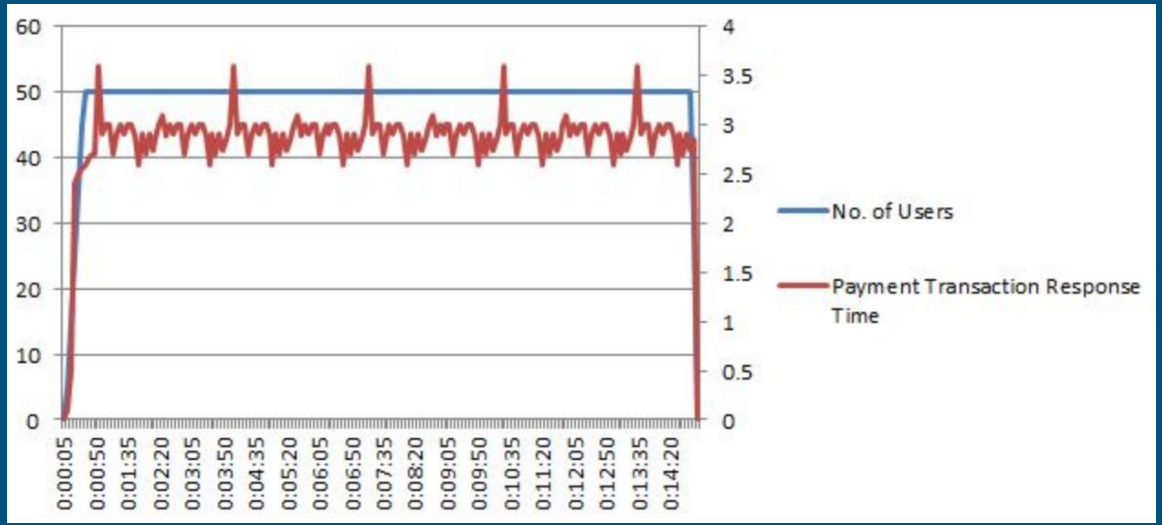
- Consistent trend:



Trends - Some examples

- Consistent with Fluctuation (not Spike):

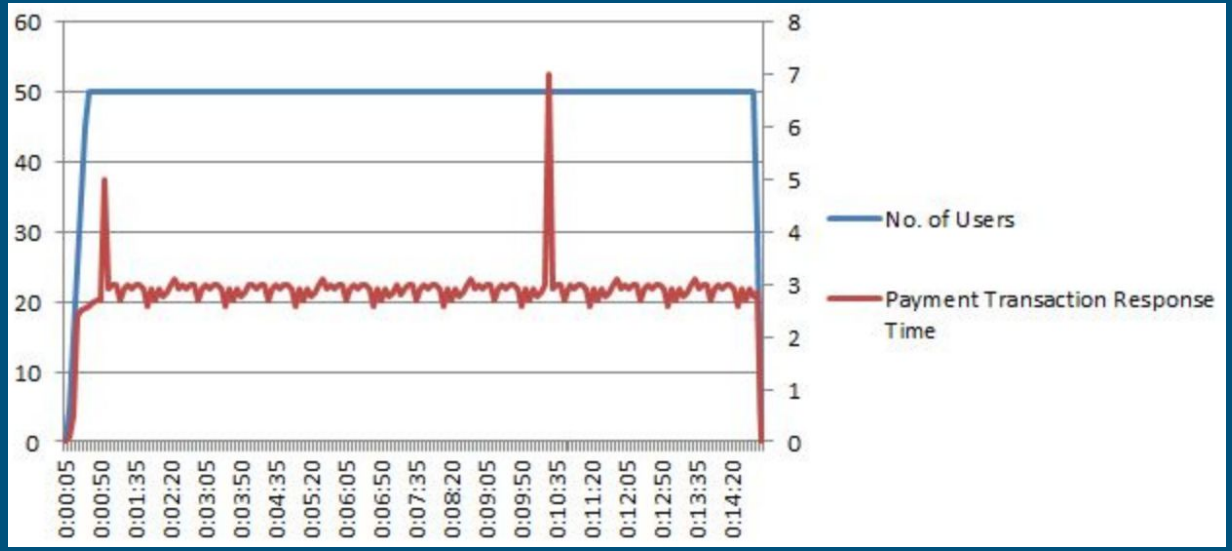
In such case, you can see the fluctuations over a long duration, but the graph is very much consistent (<0.2 standard deviations)



Trends - Some examples

- Consistent with spike:

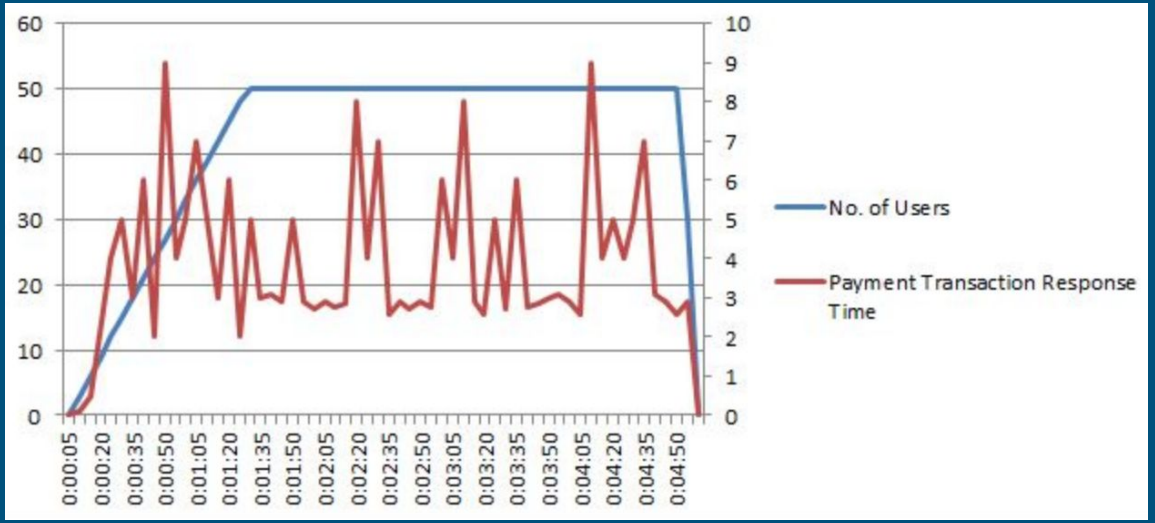
The spikes can be due to inadequate resources in the system like threads. Some of the suggestions to analyze the bottleneck are to look at the server resources like threads, CPU Utilization, ...



Trends - Some examples

- Regular Peaks:

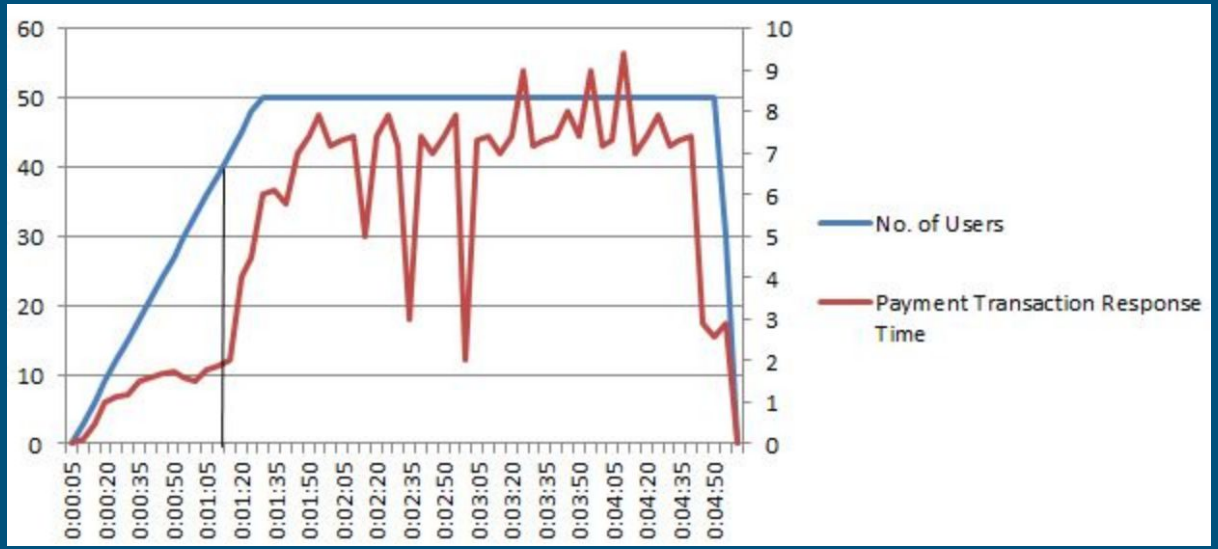
The spikes can be due to inadequate resources in the system like threads. Some of the suggestions to analyze the bottleneck are to look at the server resources like threads, CPU Utilization, ...



Trends - Some examples

- Increasing trends:

The spikes can be due to inadequate resources in the system like threads. Some of the suggestions to analyze the bottleneck are to look at the server resources like threads, CPU Utilization, ...



Basic graph

1. No of Users Graph:

Make sure the number of users are ramped-up, steadied and ramped-down as per the defined workload scenario

Users should not get completely failed and exit in the middle of the test

If failed/exit then what is the reason?

i. End of parameters in parameter files due to disabled re-cycle
ii. LG become out of memory
iii. Suspension of test user account

Basic graph

2. Response time graph:

Response time graph gives you a clear picture of overall time including requesting a page, processing the data and loading a page. You should know the response time SLA before analyzing the result. In case you do not have response time SLA then present the result to your business analyst (BA) and confirm whether response time is under the acceptable limit or not.

Generally, the response time for a normal user load website/application is 5 seconds although for a web-service it is less than 1 second or even in milliseconds still it is recommended not to predict anything without BA confirmation.

Basic graph

2. Response time graph(continued):

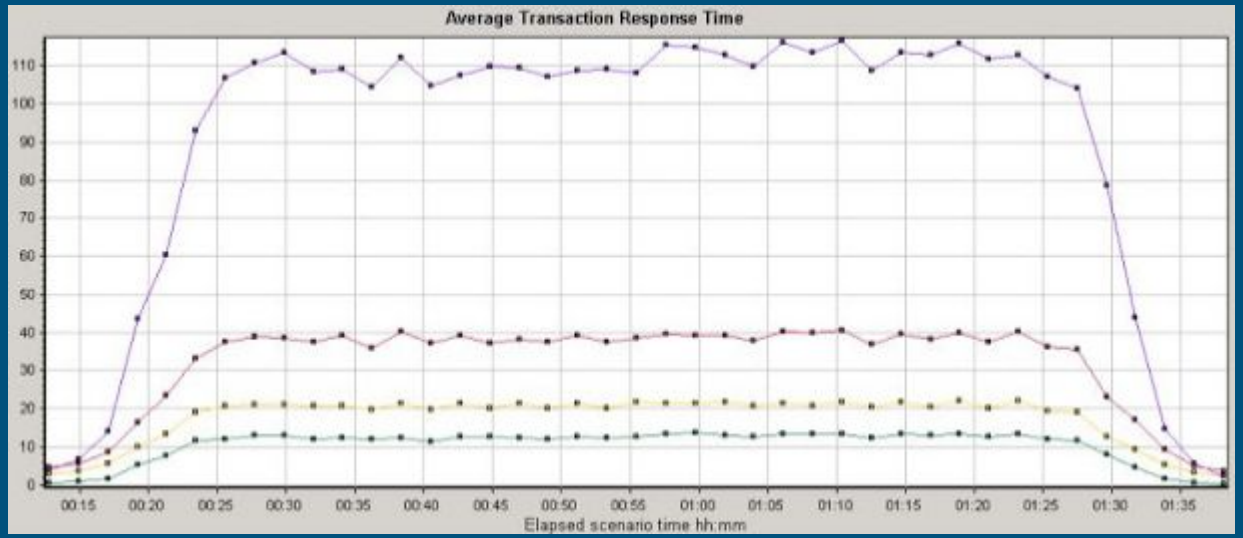
Tips:

- Response time may increase on increasing the number of users. If not, then the application has a good capability to handle a certain load.
- Response time may increase on increasing the number of users. If not, then the application has a good capability to handle a certain load.
- Response time may increase on increasing the number of users. If not, then the application has a good capability to handle a certain load.
- Try to analyse 95^{th}, 90^{th} or 80^{th} percentile response time.

Basic graph

2. Response time graph(continued):

Examples:



Basic graph

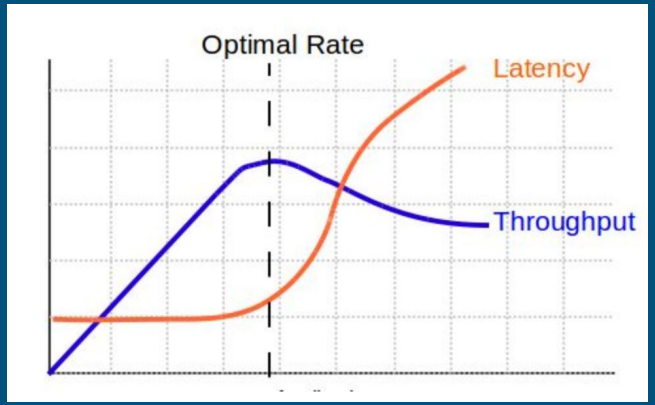
3. Throughput:

Latency and throughput are more related to Network.

If there is a sudden increase in latency then this implies that the alleged bottleneck has no relation to issues caused by the web-server itself. Thus we can consider that the performance issues are caused by low network capacity.

If latency increases and Hits per second or Throughput decreases

or shows flat graph then this is the network bandwidth issue.



Basic graph

4. CPU Utilization:

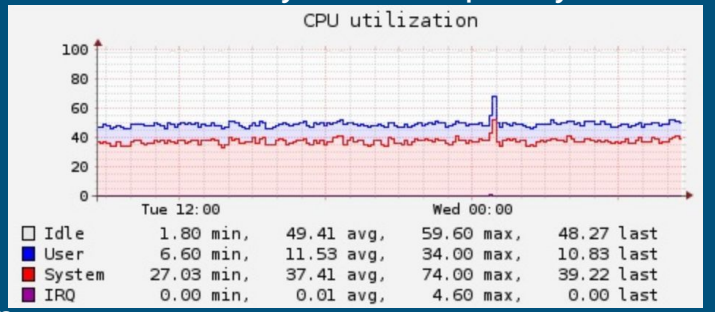
It is better to have:

- As low as the application's CPU utilization, it shows its stability
- Note down at what point CPU reached 100% or breached SLA and investigate by analysing response time, no. of users and throughput graphs
- Check how many times you got HTTP 3XX, 4XX and specially 5XX errors. You can get this count from HTTP status code/Response code graph.

Basic graph

4. CPU Utilization:

Ideally, CPU utilization should be less than 60%, although the utilization percentage totally depends on giving SLA. As a performance tester you could mark less than 60% CPU utilization as GREEN, 60%-80% AMBER and more than 80% as RED but it is generic. It would be different based on system complexity.



Tips:

a. CPU Utilization must be constant (+/- 2% tolerance)

during steady state.

b. There must be gradually increased in utilization % during ramp-up.

c. Continue Peaks in utilization % graph leads the investigation of the bottleneck.

Basic graph

4. Memory Utilization:

- As low as the application's memory utilization it shows its capacity to handle more load
- To check memory leak in the application perform Soak (Endurance) Test and monitor the memory % 3-4 hours after test completion. Why? Just to check whether memory become normal (Pre-Test Memory Utilization % =~ Post-Test Memory Utilization %)