



# Analyzing Response Time Bottlenecks in Web Applications

**By Sanjay G Menon  
Symphony Services Corp (India) Pvt. Ltd**



## Table of Contents

<b>TABLE OF CONTENTS</b> .....	<b>2</b>
1. ABSTRACT.....	<b>3</b>
2. AUDIENCE.....	<b>3</b>
3. BACKGROUND AND PROBLEM STATEMENT .....	<b>3</b>
4. POSSIBLE SOLUTIONS OR TRICKS OF THE TRADE .....	<b>5</b>
4.1. CASE STUDIES.....	<b>7</b>
4.1.1. <i>Single User Scenarios</i> .....	<b>7</b>
4.1.2. <i>Multiple User Scenarios</i> .....	<b>9</b>
5. FUTURE DIRECTION AND CONCLUSION .....	<b>9</b>
6. AUTHOR PROFILE .....	<b>9</b>

## 1. Abstract

A common problem amongst the millions of web users around the world is the performance of web sites. Performance in common man's terminology can be defined as the time experienced by the user for an action performed. Action could be as simple as a log in to an application or as complex as creating a complex report (one that has a lot of data dependencies). All the user sees on doing a particular action is an hour glass rotating or a message stating that "Processing is in progress; Please wait".

What is Response Time? Response time in performance terminology is the time taken by the server to respond to the request. In end user terminology, Response time is the time for the user to view the data requested on the page. The industry standard for viewing data on a web page is 3 seconds. More often than not users would stop or discontinue the activity on the browser if it takes more than 3s.

Why do we see such high response times in our applications? This is the question that users tend to ask performance engineers. In the subsequent sections of this paper we would discuss and analyze some of the probable causes of high response times which translate to low performing applications.

## 2. Audience

Many number of times most of the application developers and users would have faced performance issues in their applications. This document talks about some of the issues the author has faced or seen and how they were analyzed. So anybody who has faced or is facing high response time issues can refer to this document. This paper does not claim to be the bible for all high response time performance issues but can be used as a reference.

## 3. Background and Problem Statement

Overall performance of an application can be improved only by looking at all the below objectives at the same time:

- Response Time
- Throughput
- CPU utilization
- Memory consumed
- User load

Even though all the objectives above are interdependent, Response Time is something which the user experiences and is glaring.

Applications around the world are observed to have response time issues. Some applications would have not gone through a performance testing cycle due to which the response time of the whole application is bad for a single user as well. Some applications would experience response time issues under load.

In the performance testing cycle for any application the user scenarios need to be studied. The scenarios which are the most important or rather the frequently used are usually tested. The user load is studied. Load can be characterized into peak and normal load. Peak load is when the maximum number of users accesses the application and normal load is when there is not as much stress on the system. The architecture as well as the network topology needs to be understood. The network topology is needed so as to identify whether all the tiers are in the same machine or a different machines, whether the different machines are in different networks and is there latency between the machines etc. The hardware configurations of the machines are also studied. In the performance lab the engineers use this



information to study (benchmark), simulate and then try to optimize the performance of the applications. Based on the performance of the application the performance engineers can suggest the performance on the application. This information can also be used for capacity planning for the organization.

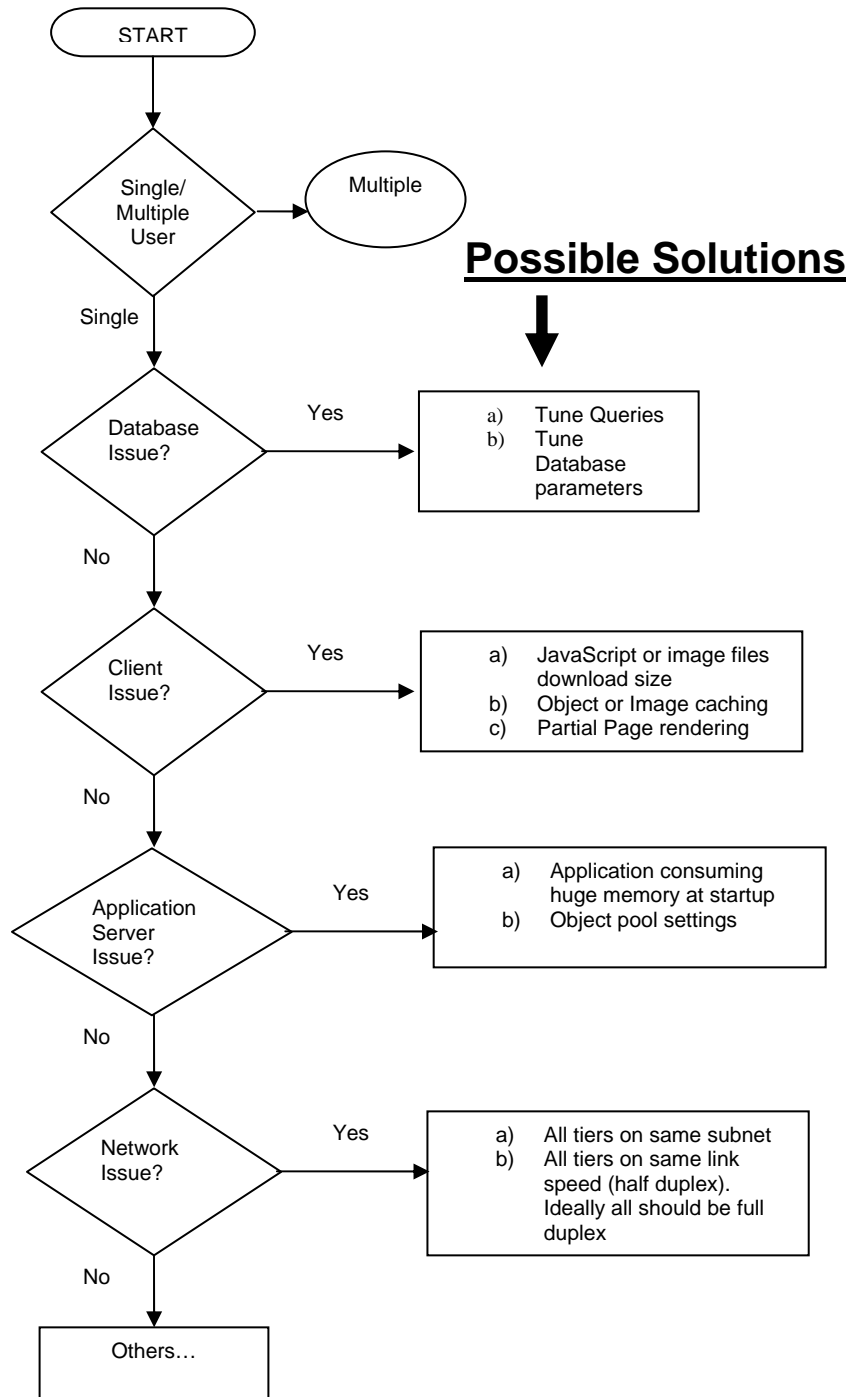
If a web application or a scenario in a web application or an action in a scenario is observed to have high response times then how does the performance engineer or in some cases the end user know the cause of this issue?

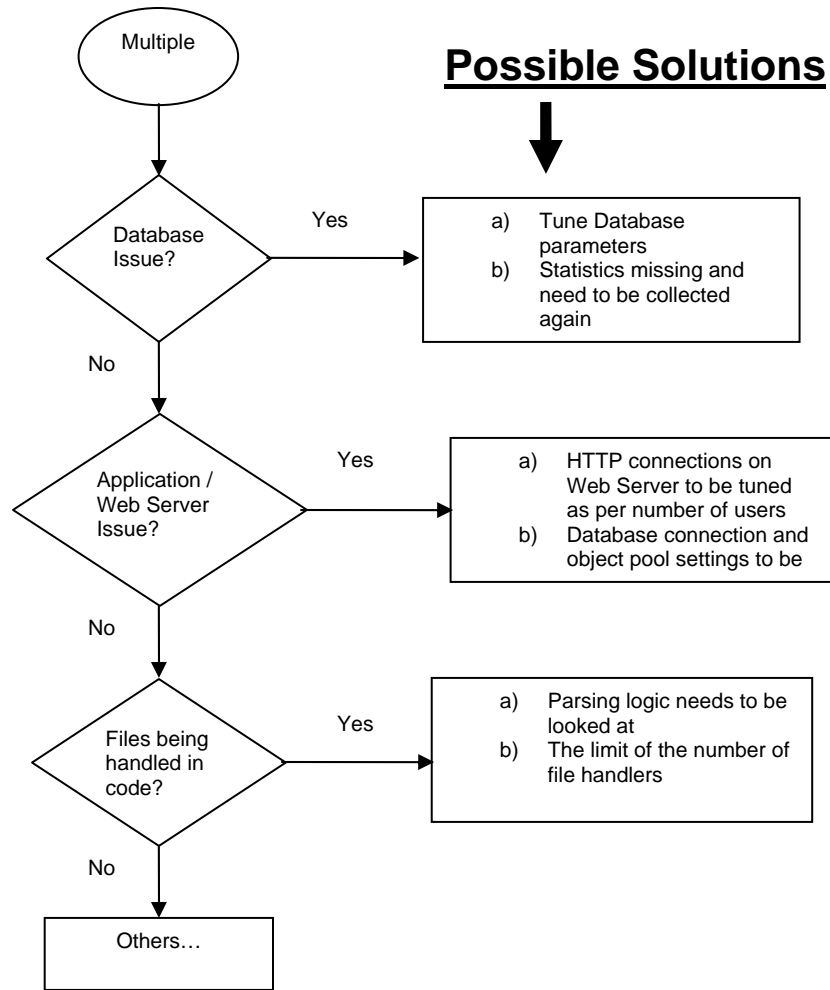
All web applications are N-Tiered. N-Tiered applications are applications having a logical split between the client, server and database. The whole business logic could be at the application server level. How does the performance engineer or end user know which tier is the actual cause of the issue?

The problem here is not only finding out the cause of the high response times but also to find out the location of the problem. The problem could be at the web server or at the application server or at the database layer, application itself or network etc. All you know it could be at the browser level. We also have to keep in mind that the other performance objectives like CPU, memory utilizations etc do not degrade in trying to solve the response time issue.

## 4. Possible Solutions or Tricks of the Trade

Some of the tricks of the trade which could help performance engineers to drill down into response time issues-





## 4.1. Case Studies

### 4.1.1. Single User Scenarios

#### ***User Search operation***

Application Source: An application stores the user related data into a central directory (database). An end user can create, search or delete users. The search operation was taking nearly 120 seconds to retrieve the data even for a single user. One of the criteria was that the search operation should not be more than 5s for 80 users.

Solution: The performance team took database snapshots and found that there are a couple of issues:

- The database statistics that were collected were old
- One of the key tables was missing a index

Result: The response time of the search action came down to less than 1 second for a single user.

#### ***Navigation to a list page***

Application Source: A CRM application had one of the scenarios wherein upon navigating one of the pages showed a list of 1 lakh "opportunities". Now this used to take more than 3 minutes to show the list for a single user.

Solution: The performance team suggested that the SQL fired to retrieve the data have an optimizer hint of First\_Rows(number of rows). The number of rows was set as 10 such that the first 10 rows are retrieved immediately.

Result: The response time of the application came down to less than 1 second for a single user.

### ***Partial Page Rendering***

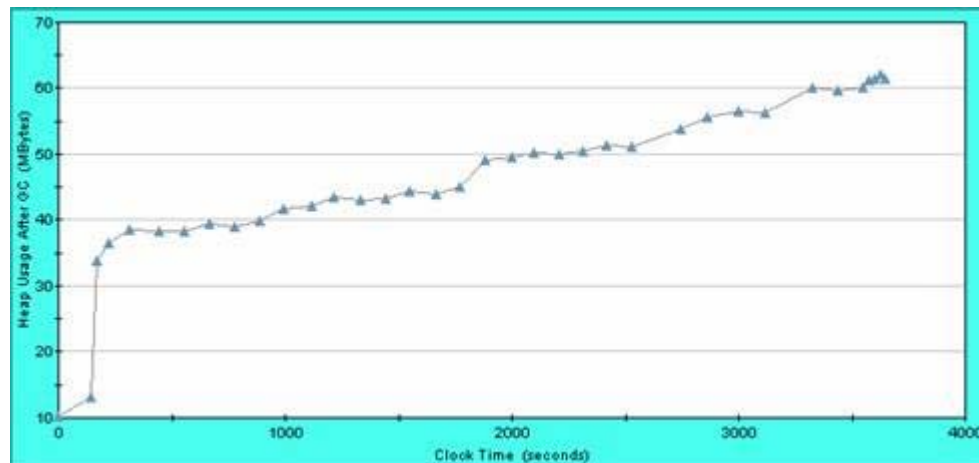
Application Source: A Rich Client CRM application had response time issue. There were multiple frames which retrieved the data from different objects.

Solution: The performance team suggested partial page rendering logic to the code wherein instead of executing all the queries to retrieve data on the page, only the required data could be retrieved.

Result: The End user response time of the application came down to less than 1 second for a single user.

### ***Memory Leak with time***

Application Source: A simple CRM application was observed to have response time increase as the user keeps navigating through multiple iterations (single user case)



Solution: The performance team observed that the heap usage was increasing at ~1K per iteration. As soon as the max size limit was reached JVM crashed.

Result: The response time of the application came down to less than 1 second for a single user by fixing the code for the memory leak.

### ***Network Issue***

Application Source: A CRM application was observed to have very high response times for a single user. This was observed for any action in a scenario.

Solution: The Database tier was on a different subnet with respect to the application and web tiers.

Result: Performed the operation by connecting all machines in the same subnet and observed sub second response times

#### 4.1.2. Multiple User Scenarios

- **Workflow related services**

Application Source: A simple workflow application which after creating tasks would go ahead to query and then approve the tasks. The response time was very high for multiple users.

Solution: The performance team took database snapshots and found that there was a particular query which was executed 20000 times within a span of 15 minutes which was found to be unnecessary. They found that they could achieve the same with only 3000 executions.

Result: The response times drastically reduced.

- **Portal Application**

Application Source: A huge portal application had response time issue with increasing number of users. The users did not know as to where the issue was coming from since this response time increase appeared only after an hour.

Solution: The performance team observed that there was a timeout of 60 minutes specified in one of application server machines after which the requests got into a queue and the response times increased drastically.

Result: The timeout issue was increased to a high value as well as the session information was retained in objects to avoid failure of the application.

## 5. Future Direction and Conclusion

All applications should go through rigorous performance testing right from the start of the development cycle. This would address the issue of having to solve architectural issues later on in the ball game. Also the data collected would be important for reference.

Performance is the key to any application and as mentioned previously no user likes waiting. Time is money. So any action that takes a long time in an application should be looked at. If necessary application configurations need to be tuned or queries need to be tweaked to achieve faster response times.

## 6. Author Profile

The author has been with the Performance engineering team at Symphony Services for the past four and half years. He has more than 10 years of IT experience of which more than 5 years have been spent in performance engineering of different applications.

All the case studies mentioned here are real experiences observed by the author in the performance cycle of the different applications.