

**Performance Tuning of  
Software Product  
By  
Rajib Roy**

## Prologue

Speed, Quality and Capacity are the critical components to success in the product based business environment today and tomorrow. Software Products those control the business, process and manufacturing, needs to provide all the three requirements successfully. To ensure success all product based projects need to deploy strategic performance testing and tuning process.

Performance is a "must have" feature. No matter how rich the product is functionally, if it fails to meet the performance expectations of customer the product will be branded a failure.

The methodology described in this doc can be followed for any software product performance tuning. Without a proper performance testing it's not possible to find out the bottleneck for fine tune. And, successful completion of tuning of a product the reliability, capacity, and performance can be determined. Tuning is an activity, which is performed after the performance testing. Identifying the bottlenecks, pinpoint the transactions associated with the bottleneck, and then product need to undergo a fine tune process for better performance.

## Performance Tuning

The ultimate goal of performance tuning is to experience an excellent end-user experience. Performance tuning, measures an application and its operating environment against specific response criteria when many people use them simultaneously.

- **The Need for Performance Tuning**

Most of the web based product or application offers services to a huge end users and that's where performance of the product plays a major role. The main entity for performance is load on server, response time, throughput, database query response time; scalability etc. including some other measure of transaction completion within a specific time, error occurring per unit time etc plays major roles too.

Application architectural design decisions may be greatly influenced by the importance placed by the customer on one or more specific requirements. Incorrect design decisions, made at the outset of a project as a result of invalid assumptions, may become impossible to remedy downstream. (Remember: What we 'ASSUME' can often make an 'ASS' out of 'U' and 'ME'.)

- **Set Performance Testing Objectives**

It is useful during performance testing to start by setting clear objectives. More often than not, performance tests will seek to achieve one or more of the following objectives:

- Identify system bottlenecks.
- Verify current system capacity.
- Verify scalability of the system.
- Determine optimal hardware/software configuration for product.

Dealing with the identification of system bottlenecks is a good place to start. The scalability and capacity of system will often be directly constrained by a bottleneck, although identifying and removing a bottleneck often leads to the discovery of yet another bottleneck, so be prepared for the long haul.

- **Determine Customer Requirements Early**

It is extremely important that fully understand customer's intentions and requirements as early as possible regarding software performance i.e. the operating environment (both hardware and software) in which the product will be deployed and the manner in which it will be used.

To begin to identify customer's requirements a performance tester must determine:

- Transaction mix.
- Usage patterns.
- Data volumes.
- Maximum allowable response times.
- Minimum transaction throughput rates.

**Note:** Determining the above information is particularly important when a customer is seeking an advice on purchasing suitable hardware and software specifications in order to best deploy of a product.

- **Determine the Transaction Mix**

The "transaction mix" that the product must cope with is determined by the number of functions that product implements and the way in which those functions are executed by users as part of the activities they each perform in relation to their individual roles. Try to identify key user groups and list the activities associated with each user role.

- **Determine Usage Patterns**

To accurately simulate system usage it is important to understand the intended usage patterns for the product. By studying user roles and quantifying the frequency and concurrency of user activities it becomes possible to begin to predict user behavior and usage patterns that can be simulated in the test environment. Performance test cases must simulate real usage patterns to be meaningful.

Try to determine different levels of usage occurring over time such as normal usage and peak usage. From this information Performance test engineer can estimate double peak usage levels and design appropriate system stress tests to push product to limits that it might not normally achieve.

**TIP:** The number of virtual user licenses required for automated testing tools to test software product need not necessarily agree with the intended user base size. On most occasions, automated regression testing tools will out-perform individual user activities which need not run concurrently and can therefore be used to simulate the load of a much larger number of 'real' users.

- **Data Volumes**

If the application creates data then it's require to consider the impact of increasing data volumes over time on application performance. Forecast data volume sizes based on usage patterns and then create test data volumes as appropriate to simulate future data volume scenarios.

Creating or obtaining large data volumes can be a problem. Large volumes of test data can take considerable time to create as well as introduce unexpected hardware requirements during development.

- **Design Tests Carefully**

To design credible tests requires an intimate understanding of the system transaction mix i.e. user activities and behaviors. Even when armed with this knowledge, time and costs will ensure it will not be possible to test every conceivable scenario and so tests need to be considered carefully.

All tests must simulate real user activities under a variety of circumstances providing sufficient data to allow plotting of meaningful graphs. Unfortunately this is more easily said than done so expect to spend a reasonable amount of time designing tests and their associated pass / fail criteria.

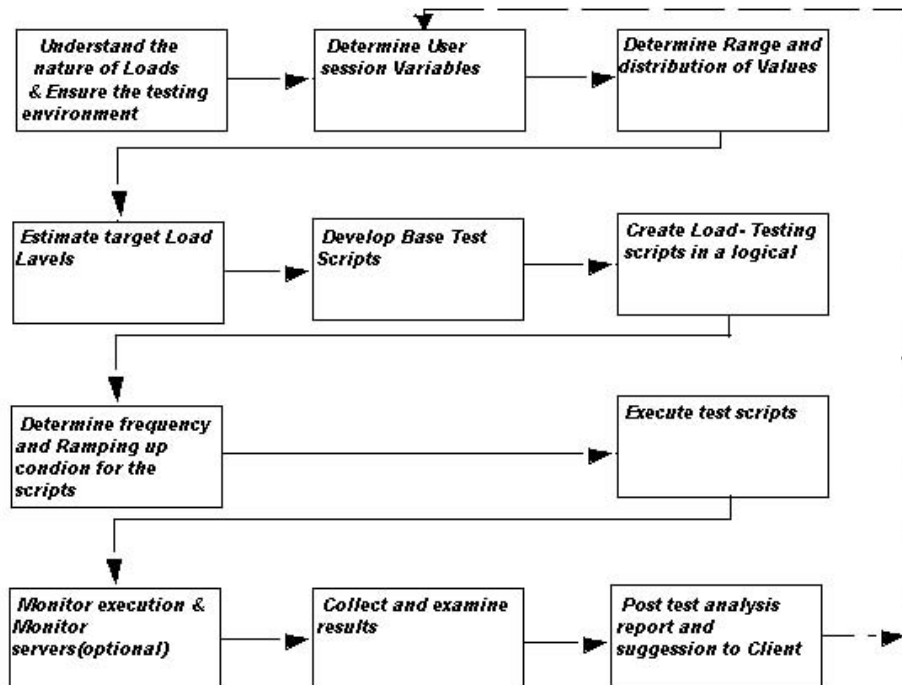
- **Keep Tight Control of the Test Environment**

For performance tests to be meaningful, test environment must be kept under strict control with no unauthorized changes being made (such as the installation of service packs or tweaking of configuration settings etc.) which might falsely influence test results leading to possible incorrect conclusions.

Take steps to ensure that the test environment is isolated from variance such as network traffic or scheduled tasks to ensure that test results are repeatable. Only variations which are choose to make as part of test strategy must be allowed. (Beware memory leaks which can prevent repeatable test results!)

The test environment (hardware and software) should mimic the intended deployment environment of the customer as closely as possible. The manner in which install and configure test software builds of the product should also mimic the way customer intends to install and configure product.

- **Follow a Testing Process**



- **Collect Metrics for tuning**

During performance testing, be sure to take and record precise measurements in a controlled environment. Treat each test as a controlled experiment. Don't just measure response times, the number of users or transaction throughput rates. Take note of system performance counters such as processor usage, memory usage, network traffic, disk input/output etc. as these can provide developers with valuable clues regarding the cause of a bottleneck or failure.

Performance rarely degrades gracefully. More often than not performance degrades drastically when circumstances suddenly change. Under such circumstances, the more operational data have at fingertips; the sooner the problems are diagnose which cause sudden performance degradation.

Performance counter information can also be used to derive the possible impact of vertical scaling on performance i.e. how upgrades to an individual computer such as adding additional memory, processors or faster disks might improve performance.

- **What to Tune and how?**

A performance tuning might require at any portion of the product, using the automated testing tools we will gather different post test data using remote monitoring agent into those resources.

While executing the Performance test scripts, it's very much essential to monitor online different graphs and data related to application and server resources. The monitoring agent of the tool gathered online information from resources and show it online. The online monitor gives immediate information about the resource usage (web server, app server, network, DB server etc.) and impact on application while executing the scripts. Different terms are monitored online while executing the scripts; they are Response Time, Hits per second, Throughput, Transaction per Second, Errors per Second, Passed/Failed transactions per second, CPU usage, etc.

The online information can also tell that it is reliable to carry out the testing or not. The online monitor helps to find out the immediate bottleneck of the application & resources.

- **The Main Concern of tuning**

A performance bottleneck can be anything which is attached with the integration of the whole system of the product. The page loading time on browser, web server, network, app server, or database server, the coding standard, software/hardware configuration, caching.

After collecting all the post test data, graph from the automated tool the performance test engineer need to analyze the data, graph and compare the data with the benchmarked value provided for the performance of the product and mark the unexpected values as bottle neck of the product and hence the component associated with it.

The page which are getting loaded if that page has some component (large size gif file) then it might be a small bottleneck for certain transactions.

Web server capacity and the actual load provided, not using multiple web servers with load balancer, caching of web server can be a bottle neck for the product performance. Deployment and configuration is the basic thing to look into.

Need to ensure the network bandwidth required for the no of users and for transactions, if the calculated bandwidth is less then the desired then network will also a bottleneck. Any network delay can be captured by network monitoring tool.

Application server logic can be a bottleneck, need to review coding standard and also the way it's uses the database connectivity and caching of the most frequently used SQL statements search results and accessing the result can be a bottleneck of the product. Deployment and configuration is the basic thing to look into.

Database server and the database it self can be a bottleneck, the Database design, proper usage of indexing, normalization, partitioning is bottleneck for many transactions. The SQL statement used for query, update etc need to profile. Any transaction is taking huge time in database need

to under go a proper review for its SQL statements and then need to send the SQL statement directly to the database using and DB tool (ex. toad) and check the time. If it's more then it need a modification. Concern developers can be noticed for the required change and tune this bottle neck.

- **Conclusion**

Performance tuning requires a different mindset and skill set to that of functional testing and is best started early in the development life cycle if at all possible.

Understanding customer requirements and expectations, as well as user activities and behaviors, is key to designing suitable tests and at the end fine tune the required areas.

System bottlenecks can rapidly become very technical in nature and consume considerable resource and effort to diagnose. Resolution may require considerable re-work and even re-design of any product.

The activity of tuning is a process need to continue regressively until the product reaches an acceptable limit of its desired performance.

Remember, "Proper V&V activity brings customer satisfaction. 'Cause, customers don't buy the product from a V&V GROUP they buy assurance of their requirements."