

White Paper



Recommendations for Performance Benchmarking

Shikhar Puri

Abstract

Performance benchmarking of applications is increasingly becoming essential before deployment. This paper covers recommendations and best practices that can be adopted when performance benchmarking of applications is being done.

Introduction

For an application, it is no longer sufficient to test functionality alone. Testing an application to ensure that it satisfies the desired performance parameters is fast becoming the norm. Performance testing and benchmarking of an application can be performed before deploying the product in the production environment or after the deployment. If the business is expected to grow, the organization should ascertain the application's ability to support the extended business volumes on existing infrastructure and the potential areas of concern or bottlenecks in terms of performance. It is essential to identify the areas where enhancements, tuning and upgrade are required.

Performance testing and benchmarking also provide proof of the performance of the product and set a baseline for further enhancement in the application.

The need to benchmark performance

The objective of performance benchmarking is to eliminate performance bottlenecks prevalent in applications and infrastructure components by identifying and tuning them. This might involve conducting performance tests iteratively on the application against representative production workload and anticipatory data volumes as expected in production after a significant period of time.

Performance benchmarking methodology

The performance testing lifecycle goes on in parallel to the software development lifecycle and starts with the requirement gathering exercise for the software application. The performance testing methodology consists the following phases:

Planning phase

The objective of the planning phase is to evolve the test plan and strategy which involve the following activities:

- Study and review the application architecture
- Understand system functionality and flow
- Review hardware components
- Assess the usability of the performance testing tool
- Define expectations for transaction performance, and system behavior
- Define performance metrics for a number of parameters including server and network utilization
- Define a testing approach
- Define a test validation strategy
- Collect the workload profile
 - Peak and off-peak time periods
 - Total and concurrent number of users
 - Key transactions, frequency and complexity
 - Transaction mix with respective user profile
 - Usage pattern and think time between transactions
 - Data access intensity and usage of data

Test suite development

In this phase the following activities are covered:

- Set up the test environment
- Create and validate test scripts
- Create data population scripts
- Create transaction data pools for use during tests
- Setup performance monitoring process

Performance test execution

This phase consists of a number of benchmarking runs, and each run has more concurrent virtual users. It continues till the performance testing objectives are met or system breakpoint is reached. If the system breaks down because of the test environment infrastructure the test results need to be extrapolated. A typical benchmarking cycle will consist of the following steps:

- Executing performance test scenarios and monitor scripts
- Collecting test and monitoring results
- Validating the results

Analysis and review

Data collected from the performance tests is consolidated and analyzed. The steps include:

- Data review (monitoring logs)
- Performance break-up across servers
- Analysis of potential bottlenecks and scalability limitations
- Identification of system and infrastructure related issues
- Analysing system performance sensitivity with the workloads

If the timeframe for issue resolution is considered, the execution cycle will be repeated after the issues have been resolved.

Recommendations

Creation of production profile in performance testing environment is almost impossible, it can only be simulated. The effectiveness and accuracy of the performance benchmarking exercise depends on the similarity of the workload simulated in test environment to that in the production environment. This section covers some of the best practices that can be adopted while performing the performance benchmarking exercise.

Transaction scripting

The performance benchmark test suite is developed by scripting selected critical transactions. These scripts when executed generate a virtual user in the system. This user performs actions or transactions recorded in the script. Various data sets can be provided to the scripts for use during the test execution to ensure the coverage of different data sets in various iterations of the transaction. To simulate the real time production scenario, sleep time is introduced in the scripts which helps execute transactions similar to real time. The sleep time corresponds to think time, data entry time and time between the iterations. A few points worth considering during transaction scripting for virtual users:

- **Think time generation:** Most tools provide the facility to generate think time (data entry time and time between the iterations) randomly, but this method generally makes the test profile random and the results can not be reproduced or repeated for verification. To avoid this scenario, think time can be scientifically generated. It can be calculated in advance to distribute the transaction profiles evenly for whole of the test duration or randomly generated and recorded in the first test. The recorded think time can be used in future tests to verify the repeatability of the test results.
- **Transaction distribution:** The test results for the steady state phase of the performance testing are considered for analysis and evaluation. The results of ramp-up (initial) and ramp-down (logout) phases are ignored. If the think time generation process is random, a major chunk of users might come into the system and start their transaction iterations at the same time which will bring about a spike in resource usage and skew the results. Also, infrequently occurring transactions that get executed in ramp up and ramp-down phases will be absent in the system throughput considered for analysis and evaluation. Control over the scientifically generated think time helps in devising the login behavior and transaction start-phase for virtual users. Users start the iterations gradually, achieve a system steady state and come out of the system once throughput of all the critical transactions has been met. This portrays a scenario that closely resembles the real-time usage of the application.

- **User session management:** Scripts should be created in such a manner that all virtual users get injected gradually during the initial user ramp-up phase and remain in the system till the test is over even if their assigned iterations have been completed, which can be achieved by setting a large value for the think time towards the end of virtual user scripts. This helps maintain the ratio of active to inactive sessions in the system. Testing scenarios depend on the performance test requirements and real time transaction usage pattern. The script design should be according to the scenario being tested.
- **Data pool selection:** Data pool sets for transactions should be carefully selected. The transaction data usage should be in proportion to the real time usage of the data. If the real time system is supposed to perform an enquiry, the distribution of data in terms of different query parameters, data creation dates, range of data selected (example wild card enquiries or date range), data access intensity (rows to be returned for query), usage of list of values, repetition intensity of the same data or similar data in further transaction iterations should also be considered to simulate different types of enquiries and ensure a wider coverage of transaction scenarios. These parameters produce the workload profile and test results near to real time scenario.

System configuration and setup

The configuration and setup of the test environment should be similar to that of the production environment. A few factors that need to be considered on this front are:

- **Load balancing scenario:** Load balancing in the test environment should be the same as in the production environment. A scenario where users are distributed on a round robin basis might give rise to issues in the test environment. Users in real time execute multiple transactions in a random fashion, the transaction usage pattern and behavior do not remain constant over a period of time. In the simulation exercise, a transaction or a group of transactions are recorded in a virtual script and that script is executed with multiple users. A virtual user repeats the same activity in multiple iterations with different data sets. If virtual users or scripts with heavy transactions connect to any of the servers they will repeatedly send requests to the same server. Though the number of users will be balanced on multiple load balanced servers the load will still be imbalanced between multiple requests.
- **Distribution of loaded data:** The test environment should be pre-populated to a level which is comparable to production environment data for around 6 months or one year. The time duration, for which data needs to be loaded, depends on the data archive and purging requirements of the system and the expected system longevity. The pre-populated data should have distribution similar to that in production environment. Various business parameters and data distribution parameters need to be collected in advance or production profiling should be done to generate the data load scripts. If production data is available, the ideal scenario will be to replicate or clone the production data in the test environment. This helps in generating the response time and elapsed time of the transactions similar to that of the production environment.
- **Placement of data on disks:** If data population scripts are used for loading data in the test environment and the data is loaded sequentially in the tables, data related to specific tables will be populated on the same disk. While performing the transactions on these tables or related tables, either the data is accessed quickly (due to operating system caching mechanism) or the disk becomes a bottleneck (due to excessive I/Os on specific disks). This might not be the case for the real time scenario as multiple transactions are performed during a day and the data for a specific transaction gets created gradually on a daily basis. The data is distributed automatically on various disks and the probability of the disk being a bottleneck is reduced. The placement of data on disks should be as close to production as possible. The ideal scenario will be to clone the production data in the test environment keeping the disk structure similar to that in the production environment.

Load injector machine setup

Virtual user scripts contain a single transaction or a group of similar transactions which are generally executed sequentially. These scripts are configured to be executed on load injector machines with a designated number of iterations as per the system workload profile.

Factors that need to be considered while configuring the load injector machine and virtual scripts:

- **Data caching:** The virtual user script for a single transaction with assigned number of users is configured on a single load injector machine. The data pool or pre-requisite data for the multiple iterations of the transaction can be different (as configured in the data pool) but the requests from all the assigned users for the candidate script are generated from the single load injector box. If the load injector box caches transaction specific static data the response time of further iterations of the candidate transaction will be reduced and will not portray the real time scenario. To avoid issues with data caching the machine should be set-up properly or the distribution of the virtual user script and transactions should be in place.
- **Distribution of IP addresses on the load injector:** Multiple transaction scripts and virtual users configured on load injector machines use the same IP address for execution. This means that all virtual users will be using the same IP subnet mask. In a real time scenario only a few of the users will be using the same subnet mask. The test results in this scenario might differ from the real time results. Tools are available which can assign multiple IP addresses to different virtual users assigned on the same load injector box, this helps generate a better picture of the real time scenario.
- **Bandwidth simulation:** The performance benchmarking exercise is normally performed on the server side. The load controller and injector machines are connected to the servers with high speed LANs, network congestion and delays in transactions are not taken into account. Tools are available to define the usage of the network bandwidth by virtual users on load injector machines. If the system is locally deployed or de-centralized, the benchmark tests should be performed with real time network bandwidth. And, if the system is globally rolled out, with multiple sites injecting load on the system at the same time, the transaction load from various sites should preferably be allocated to different load injector machines. These site specific machines should be simulated with the network bandwidth available at respective sites. The pre-requisite for this scenario will be the availability of network topology of various production sites. This will give an accurate picture of the network congestion and delays for the system.
- **Memory requirements of virtual users:** All the users assigned to a load injector box require memory to execute a transaction. The maximum memory consumption by a single virtual user should be calculated in advance and distributed accordingly, to avoid congestion on client machines. Not more than 70-80% of the available memory on load injector machine should be consumed.

Execution and analysis

Factors that need to be considered for execution and analysis of test results:

- **Validation of transaction response:** Usually the data used during tests are different from the data used during recording or creating a transaction script which means response at various steps in a transaction script can be different from what is expected while recording or creating. To overcome this issue, the virtual user scripts should be devised in such a way that the response from the request is validated for success and further progress. The next step in a transaction should be executed depending on the response received from servers.
- **Profiling of transaction steps:** Transactions can be further divided into multiple logical steps and these steps can also be monitored for response or elapsed time. If the transaction response or elapsed time is above the expected response time these logical steps will help provide a better picture to understand bottlenecks in the transactions. The problematic steps can be further looked into for removal of performance issues.
- **Validating execution of transactions:** After the benchmark test, the simulation tool provides the summary of test results including transaction response time, transaction throughput etc. The throughput mentioned in the test results summary is based on the execution of the steps mentioned in the virtual script. There could be a scenario where the transaction fails but the virtual user step gets executed, this will show up in the test results summary. The best way of verifying the transaction throughput is to verify the occurrence of the transaction from the database or the affected areas. This will be easy if the transaction is performing create, update or delete operations. For “read” operations, the opening of the screen can be logged somewhere so that the transaction occurrence or success can be verified.

- Transaction response time: Generally the transaction response times are presented as 90th percentile transaction response time. This means that 90% of the total transaction response time occurrences are below the figure marked as 90th percentile figure for the candidate transaction. This is better than presenting the average response time figure, which averages out the peaks and variations in response times. Sometimes even 90th percentile figure does not portray the correct picture and further analysis should be carried out before presenting the results summary. For example, if a transaction that has very low iterations (2 per test duration) and if one of the iterations happened during the initial stage or when the system was not performing well, the 90th percentile response time will be the response time for the mentioned iteration and that will be very high. In this scenario, we can ignore the mentioned iteration with proper justification and consider the response time of the 2nd iteration. Similarly if the frequency of transactions is high and 90th percentile response time is quite above the expected response time, the difference between 85th percentile and 90th percentile response time should also be checked. If the difference is not much the transaction has performance issues, but if the difference is very high the iteration occurrences should be checked and the reasons should be examined. It is observed that usually the transaction is performing well but there are apparent issues in the system when the iterations for the candidate transactions are performed.

Conclusion

Performance benchmarking, pre-deployment, is an increasingly prevalent practice. It is an essential step in ensuring non-functional requirements are satisfied.

However, performance benchmarking isn't a trivial exercise. It requires extensive and detailed planning. There are a number of best practices that have been developed over the years. Following these best practices ensures that you get the best results from the exercise. When properly executed, it eliminates rework post-deployment, thus reducing the cost of an application.

About the Author

Shikhar Puri, a Senior Technical Architect with Infosys. He has 8 years of IT experience. His areas of interest are predominantly in the performance engineering domain concentrating on strategies & solutions for defining and satisfying non-functional requirements, workload modeling, system sizing, performance tuning & testing and simulation of enterprise systems.



For more information, contact askus@infosys.com

About Infosys

Many of the world's most successful organizations rely on Infosys to deliver measurable business value. Infosys provides business consulting, technology, engineering and outsourcing services to help clients in over 30 countries build tomorrow's enterprise.

For more information about Infosys (NASDAQ:INFY), visit www.infosys.com.