

Challenges and Solutions of Performance Testing on SaaS Applications

Software as a service (SaaS) is a widely adopted application delivery model in which applications are hosted by a vendor or a service provider and are made available to the customers over a network, like the Internet. SaaS solutions are developed to leverage Web technologies, such as the browser, thin client, thereby making them web-native. SaaS applications are particularly designed with a 'multi-tenant' backend, which enables multiple customers or users to effectively utilize a shared data model.

A SaaS implementation gives 'Scalability' a completely new dimension, as it enables the SaaS providers to leverage enormous economies of scale in deployment, management, and support. Hence, it is crucial for a SaaS vendor to ensure that the SaaS application is able to handle the existing customer base while keeping pace with its projected growth. Testing the SaaS platform for its ability to handle increasing levels of demand and throughput is essential to avoid future bottlenecks and deliver a reliable service to customers.

SAAS PERFORMANCE TESTING

If SaaS applications are web-native, is there any difference between performance testing a SaaS application and a traditional web application?

An important difference between a SaaS application and a traditional Web based application is the number of end-users, underlying infrastructure, and the architecture complexity. It is essential to consider these factors before conducting performance testing on a SaaS application.

The Challenges

The SaaS model provides many benefits to the application providers and customers in terms of easy maintenance, cost savings (hardware, human resources), etc. But, it also introduces a number of

questions for the performance engineer, such as:

- What are the factors that can affect performance of the SaaS application? Is it important to differentiate between the SaaS platform and the applications hosted on it while performance testing?
- Is it necessary to carry out performance testing of all the nested applications (tenants) hosted on a SaaS platform or will testing only some of the tenants suffice?
- Which tenants need to be considered for the performance tests?
- Does customization affect the performance of the SaaS application?
- Can a traditional load testing tool be utilized?

It wouldn't be wrong to say that the performance questions don't change, but the answers do.

Two factors that directly affect the performance are the layers of users and customization.

SaaS applications have two kinds of users; the "tenants", which is, the SaaS platform provider's users (i.e. aggregator/resellers) and the "end users," the tenant's customer base. Consider that the SaaS application has 'n' tenants, i.e. Tenant 1, Tenant 2 ... Tenant N. Each tenant has 'p' end users for its application. For example, Tenant 1 can have 'P1' end users, Tenant 2 can have 'P2' ... Tenant N can have 'PN' end users. Hence, if 'm' is the total end users of the entire SaaS platform then $m = P1 + P2 + \dots + PN$.

The scale of end users is high because of the numerous layers involved. Each of the tenants hosts its customized application on the SaaS platform. Most often the services exposed by the platform needs to be customized to fit the business requirements or the workflow of the tenant SaaS application. These customizations could be in the form of permissions, UI, provisioning, etc. The degree of customization greatly influences the

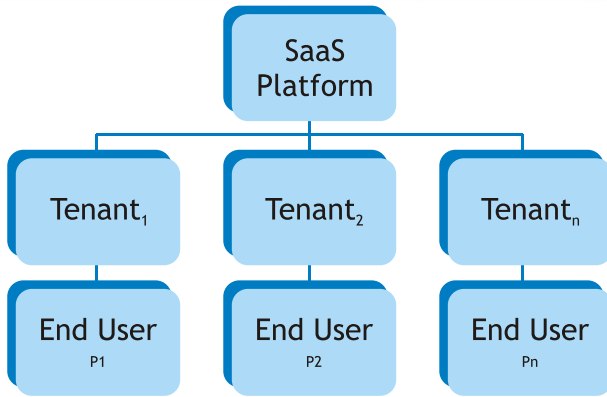


FIGURE: LAYERS OF USERS

complexity of implementation, which in turn can cause performance issues.

Solutions

To tackle these challenges a tiered approach can be taken. This approach is based on the fact that every application is built on a SaaS platform which provides core functional services like ordering, provisioning, billing, and management. These services are customized and consumed by the tenants to create the SaaS application which is in turn used by a set of end users.

The first tier is intended to set a baseline for the SaaS platform. This involves testing the SaaS platform on performance parameters like transactional throughput, response time, and error-rate.

The next tier in the approach is to execute baseline tests for multiple tenant applications. After this, the delta between the performance of the tenant and the performance of the platform is evaluated. If the delta between the baseline platform test and the baseline tenant test is negligible, then it indicates that the performance of the tenant has not been affected much due to customization. However, if the difference is large, then earmark the tenant for further performance tests.

Next, benchmark the SaaS platform for various load levels to obtain performance metrics. The final step in the approach is execution of the performance tests for the tenants (identified earlier based on higher delta).

The tiered approach presents an optimal way of performance testing SaaS applications.

PERFORMANCE TESTING METHODOLOGY FOR SAAS APPLICATION

Performance Testing is an intricate science, which can be dealt with by logically grouping the never-ending list of activities into 2 'phases'. At a high level, these are Strategy and Planning; and Scripting, Execution, Monitor and Reporting.

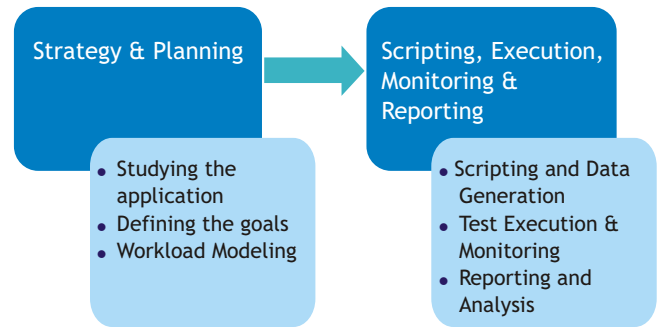


FIGURE: PERFORMANCE TESTING METHODOLOGY FOR SAAS

Phasing out the tasks that are carried out for performance testing not only helps in reducing the complexity to a manageable level, but also enables to address the questions raised in the previous section in relevant phases.

A performance engineer should remember that SaaS solutions are essentially web applications; hence the core concepts of performance testing are applicable here.

PHASE 1: STRATEGY PLANNING:

Challenge: How to define the performance goals

In the planning phase, the performance test engineer has to spend time studying the application architecture and implementation, even if the scenarios have been identified for the engagement.

While defining the performance goals for a SaaS application, a performance analyst has to consider 3 perspectives:

- 1) SaaS vendor/ SaaS platform provider
- 2) Tenant (Aggregators/Re-sellers)
- 3) End User

Performance metrics like availability, transactional throughput, response time and error rate must be

measured for the tenant applications as well the SaaS platform.

In the absence of a tiered approach, the perceived goal for performance testing a SaaS application would be to ascertain that 'm' number of total end users is supported for 'n' different available tenants (aggregators) for a pre-decided number of scenarios. For example, evaluating whether the performance criteria are met when the platform is expected to face 10 tenants and 1000 total end users.

In order to validate whether the application can survive 'm' total end users, the work load model and script will have to be executed for all the tenants. This theory is thus applicable if there are 2 or 3 tenants, but for a large number of tenants it will be considered as a highly complicated, time consuming, and impractical approach for performance testing a SaaS application.

However, by employing the tiered approach, the complexity reduces to a manageable level and the goal for a performance testing engagement for a SaaS model compulsorily becomes dual-pronged. The first objective is to gauge the performance of the SaaS platform, while the second objective is measuring the performance of SaaS tenants (application).

For testing the SaaS platform, the performance goal could be to establish the performance SLA's for the services exposed by the SaaS platform. An example of this could be finding the transaction throughput for the billing and metering service.

Here the assumption is that the underlying SaaS platform exposes the services for testing.

For SaaS tenant testing, a potential performance goal could be *to determine whether a tenant application can support 'p' end users without compromising the performance acceptance criteria.* For example,

- Evaluating whether the tenant-1 can support 200 hourly users with acceptable performance SLA
- Bulk loading tenant-2 with employee data comprising of 10000 employees in less than 1 hour
- When subjecting a tenant-3 to a design load of 200 concurrent users, 80% of all page responses shall take less than 5 seconds

Such goals need to be validated for the tenants.

The mantra to define performance goals would be dual-pronged/tiered SLAs for a tiered architecture.

Another important activity carried out in this phase is determining the types of tests which are most suitable for a SaaS application. Scalability is a perpetual bee in the bonnet for a SaaS vendor and performance tests (load test, stress test, etc) are recommended as per customer requirement.

A vital task that is done in the Strategy Planning phase is Workload Modeling. The primary challenge is to determine which tenants and scenarios to consider for the performance tests. The next is to arrive at a realistic distribution mix for the scenario as well as the tenant to end users ratio.

It is essential to come up with a workload distribution model for the tenants as well as the SaaS platform. The two important activities that are accomplished in this phase are Scenario Identification and Load Distribution.

Identifying scenarios:

The following types of scenarios are typically included for performance testing:

- Business critical - These involve transactions that your business depends on
- Frequently used - These scenarios are popular and hence can expect a large amount of load
- Resource/Data Intensive Scenarios which involve huge amount of data or a lot of processing

Deciding which business process to simulate is a key activity in the planning phase.

For base-lining the SaaS platform, it is important to identify the relevant exposed services. These services (could be web services or API's) may fall into aforementioned categories.

For performance testing the tenant's application, the primary question that needs to be answered is: **Which tenants to consider for tenant's-application baseline tests?**

Pointers for short-listing tenants to be considered for baseline tests are given below:

- Number of end users the tenant is expected to support selecting the tenant with lowest and highest end users

- Percentage customization selecting tenants with highest customization percentage
- Long term selecting the oldest tenant of the platform or a tenant registered for the longest duration
- The tenants suggested by the vendor

Load Distribution:

For SaaS platform baseline, to simulate a realistic mix, the load distribution must be considered on the following fronts:

- API / services mix: Defining the percentages of each of the chosen service/API in the overall load.
- User-Tenant distribution: If the platform is subjected to a load of say 1000 users then translate this number into the tenant - end user breakup.

The first step is to identify the number of tenants using the platform. Then the next step is to derive a realistic breakup for the total end users, i.e. $400+300+150+100+50 = 1000$.

For SaaS tenant-application performance testing, the method to arrive at scenario mix distribution for a tenant is similar to that for a web application.

PHASE 2: SCRIPTING, EXECUTION, MONITORING AND REPORTING

In this phase, we execute tasks, including Scripting and Data-Generation; Test Execution and Monitoring; and Reporting and Analysis.

Load generation tool

As SaaS applications are Web native, traditional load generation tools like LoadRunner, Webtest, VSTS, Webload, JMeter etc can be used for scripting (and execution) as long as the protocol used by SaaS application is supported by these tools.

Scripting

Test Scripts need to be created for the SaaS platform as well as the tenant applications.

These scripts must simulate the services, scenarios,

and user distribution as defined in the workload model.

Generally SaaS applications use single sign-on authentication mechanism, thereby requiring users to provide credentials only once. While scripting explicit care must be taken to ensure that user authentication is effectively handled.

Data-Generation

The crux of any SaaS application is the underlying data. Generating runtime and reference test data becomes more complicated when dealing with multi-tenancy and their respective customization. As every application is unique in its own right, one has to dedicate time on creating a customized data generation tool/ script for every engagement.

Consider a SaaS platform for e-commerce. One tenant might host a book store application while another tenant might host an application to sell CD's, DVD's etc. Here though, both the tenants share a similar data model, and the complexity of data generation increases exponentially as different flavors of data needs to be generated.

Limiting the extent of data generation by populating the database with minimal and limited reference may prove beneficial. Data-generation processes can begin in parallel with the scripting phase.

Test Execution

As per the tiered approach, the first task is to baseline the platform. Next, baseline the tenant's application for the scenarios defined in the Strategy Planning phase. Once the tests are executed, find the delta for the platform baseline tests and tenant baseline tests. Earmark the tenants having higher delta. The next step is to execute tests to benchmark the SaaS platform performance, at various load levels. The final step is to execute the performance tests (load, stress, etc that were agreed upon in the strategy planning phase) for the tenants having higher delta (identified earlier). The execution process can be depicted as below.

The traditional method of performance test execution doesn't change for SaaS applications. Effective testing of the platform may require creation of stubs around the exposed services. In case, third party /external components are involved (ex: payment processing/credit card gateways), and tools/mock services have to be built. In such cases, it is necessary to ensure that these mock services are optimized for performance and don't operate as a

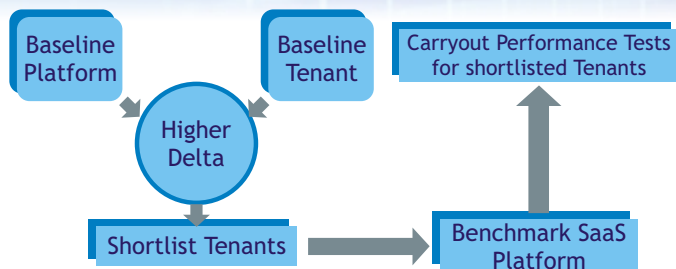


FIGURE: EXECUTION USING TIERED APPROACH

weak link. After all these third-party components are integrated, it becomes necessary to execute an additional performance test pass in the production or pre-production environment. All these tests should be adequately monitored during the execution process.

Monitoring

While monitoring, special care must be taken to monitor the availability of all the tenant's applications. This would help in determining whether other tenant applications are affected if one particular tenant application crashes due to high load. Load balancers (if any) should also be monitored. However, the scale of monitoring is directly proportional to the scale of the underlying infrastructure, which is quite high for a SaaS application. Performance engineers must always take this factor into account. There is a growing trend among SaaS platform providers to move towards virtualization because of large-scale infrastructure needs. In order to have a better control over the pulse of the overall application health, VM infrastructure monitoring will be important.

A SaaS platform provider should expose application instrumentation hooks/counters for core services. This can aid the process of monitoring and measuring the service transactions to a much granular level. Since, it is also important to monitor the performance of the underlying SaaS platform, along with the system level metrics, database performance

metrics, etc for a SaaS model. For example, in an engagement where the application is built using WCF, the WCF Service Contract (methods) throughput and response time is measured through metrics such as call per second and call duration, which are exposed by relevant instrumentation counters.

Performance SLA monitoring is a concept which has become the need of the hour for a SaaS platform. It is nothing but 24*7 monitoring of the application for performance criteria like availability, response time, error rate, etc. Here the threshold of all the performance criteria is fed into a system and an alert is raised if the threshold is breached.

Reporting

The performance engineer is bound to be bombarded with large amounts of data including results of the baseline tests for the platform and tenants, results of the performance tests for the tenants, and the monitoring logs. However, following the tiered approach simplifies the problem to a large extent.

A performance engineer has to ensure that all the results, observations, and conclusions are not only in accord with the tiered approach, but are also showcased appropriately. For example, the performance report should suitably highlight the performance criteria like transactional output, availability, and results of the baseline tests for platform and tenant as well as the delta between platform tests and tenant application tests.

CONCLUSION

While the SaaS model offers many benefits to a number of parties in terms of cost savings and time saving, it also brings certain challenges in terms of performance. The tiered approach explained here is a highly scalable and manageable approach to testing and assuring a SaaS platform and its tenant applications for performance.

About MindTree

MindTree Ltd. is a global IT Solutions Company specializing in IT Services, Independent Testing, Infrastructure Management and Technical Support (IMTS), Knowledge Services and Product Engineering, which comprises of R&D Services and Software Product Engineering. MindTree partners with its clients to create a transparent, value-based relationship. Our people build innovative solutions in a wide range of technology domains that enable our customers to succeed in their business goals.



Info@mindtree.com

USA:
Tel: +1 908-604-8080
Extn. No. 2506

Asia Pacific:
Australia: +612-9025 3538
India: +91 80-26711777
Japan: +81-3-3378-6081

Singapore: +65-6323 8135
UAE: +971-50-7395894

Europe:
Germany: +49-69-3085 5092
Sweden: +46-8-501 64 044
UK: +44 20-8832 1160