Load and Performance

Testing with soapUI

As defined on Wikipedia (http://en.wikipedia.org/wiki/Software\_performance\_testing), performance testing is in

general testing that determines how a system performs in terms of responsiveness and stability under a particular work load.

**Load testing** is a specific form of performance testing that is conducted to assess the behavior of the system under a specific load. In soapUI, though we generally use the term "load testing" for all types of non-functional testing, we can do all types of performance assessments of web services such as load, stress, and endurance.

**Limit of a load test**

**Limit** defines the load test execution interval. There are two variables for limit; the **limit value** and **limit type**. In the 4.0.1 Version of soapUI, there are three limit types; **Total Runs**, **Seconds** and **Runs per Thread**

The **Total Runs** limit type is used to set the number of times the TestCase needs to be executed during each load test run. If we set **60** total runs, you will notice that the TestCase is executed 60 times. Therefore, if a particular TestCase has *N* TestSteps, all *N* steps will be executed 60 times.

If we set the limit type as **Seconds**, the test will be run till the specified time is over. If we set 60 seconds as the limit, the test will be over in one minute. Once the test is started, you can see a log similar to the following in the load test log which is at the bottom of the load test window

The **Runs per thread** limit type can be used to set TestCase runs per threads as

needed. For example, if we set the limit as 5 runs per thread and the thread count is

2, TestCase will run 10 times.

**Threads in a soapUI load test**

Threads act as virtual users in a load test. If the thread count is set as *N*, soapUI

creates *N* number of clones of the associated TestCase and executes them. You can

set as many threads as required based on the capability of handling resources of the

system in which soapUI runs on.

**Load test strategies of soapUI**

SoapUI allows you to simulate different types of load on web services using multiple

load test strategies. In the free version of soapUI 4.0.1, we can identify the following

types of load test strategies:

• Simple

• Burst

• Thread

• Variance

**Simple load strategy**

The default load test strategy of a soapUI load test is simple. In simple

strategy, **Test Delay** defines the delay between each test run. Random is used to

set the relative amount of randomization for test delay. If **Test Delay** is **1000** ms

and **Random** is **0**, each test will be executed in intervals of 1 second. If **Random**

is **1**, all tests will be executed in random delay between each other relative to the

**Test Delay** value.

**Burst load strategy**

Burst load strategy can be used to generate a rapid load on the target web service.

With this mode, TestCases are executed in bursts without a pre-defined delay

between each of them.

**Burst Delay** represents the delay between bursts.

**Burst Duration** is used to define the number of seconds that the burst runs on the target service.

**Thread load strategy**

Thread load strategy is another useful approach for simulating load. In this strategy,

the thread count will be increased gradually from start threads values to end threads

values. If you want to monitor the behaviour of web services with an increasing

thread count, this approach is ideal.

**Variance strategy**

As the name implies, this strategy varies the number of threads over time. Within a

defined interval, the number of threads will decrease and increase as per the given

variance value simulating a more realistic load on the target web service.

**Test report and statistics of soapUI**

• min: This defines the minimum response time taken by TestStep

• max: This defines the maximum response time

• avg: This defines the average response time

• last: This defines the average response time for the last run

• cnt: This defines the total number of times a TestStep has been executed

• tps: This defines the number of transactions per second

• bytes: This defines the total bytes transferred

• bps: This defines the bytes per second

• err: This defines the number of errors occurred

• rat: This defines the percentage of failed requests