Performance Testing Course : JMeter - Timer

Author: Hien HOANG

Agenda

- Introduction about several popular timer.
- Showcases.
- Exercises.

Definition

While browsing a typical website, a real user always clicks, waits, reads and then provides input. The time spent on the page before providing any input is called "Think Time".

In the performance testing world, “Think Time” stands for simulating real user behaviour which causes people to wait between interactions with a web application.

The purpose of “Timer” element is to pause a JMeter Thread representing a virtual user for a certain amount of time. The main goal of using timers is simulating a virtual user’s “Think Time” and in some cases to add "Pacing"

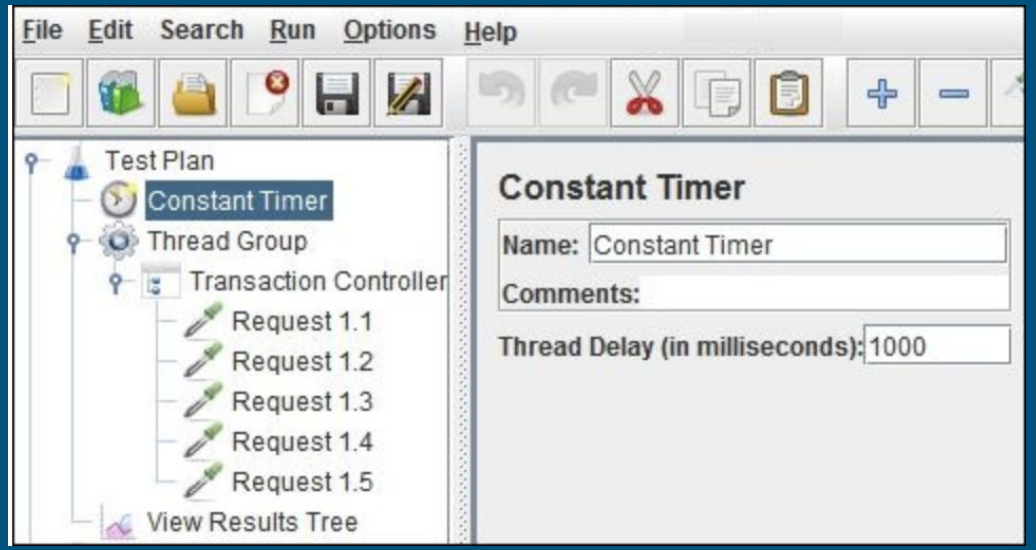
Constant Timer

Add a constant delay between 2 requests.On adding constant timer each thread pause for the same amount of time at that sampler before triggering the next request or before executing the next sampler.

Constant Timer

- Attribute:
- Name: Name of the timer.
- Comments: Description of timer.
- Thread Delay(in milliseconds):

The pause time in milliseconds.



Constant Timer

Showcase:

- Define a Constant Timer to pause each request to 1 s.

Exercise:

- Define a Constant Timer for your choice.

Constant Throughput Timer

Definition:

It helps to achieve the desired throughput (Total Number of Requests). This timer tries to maintain a constant throughput throughout the test and achieve the target.

Note:

Throughput may decrease if other timers contradict the Constant Throughput timer. Hence it is not recommended to use other timers along with the Constant Throughput Timer.

Constant Throughput Timer(continued)

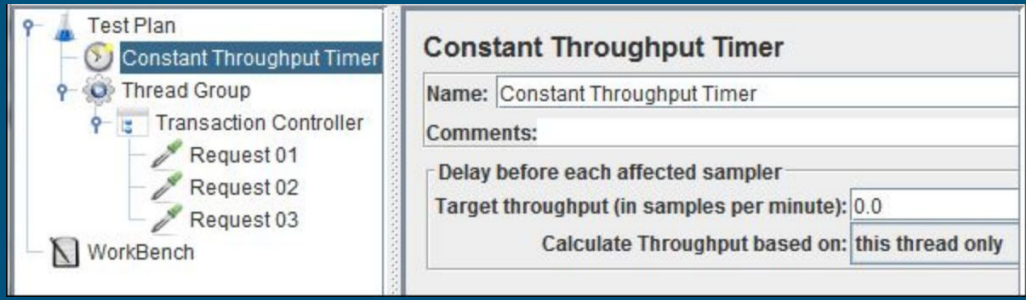
Limitations:

- Other timers may impact constant throughput timer target, so it is recommended not to use other timers.
- To get the exact sampler's count at the end of the test, you need to eliminate transaction count from the aggregate report to check whether constant throughput timer is working fine or not.
- The achieved target may be slightly high.
- The threads are not stopped in a graceful manner. Once the test duration ends, then all the threads stopped; even though they are in the middle of the iteration.
- You need to decide the number of threads wisely so that a real-time scenario can be prepared.
- Step-up and Spike Test scenario cannot be prepared using constant throughput timer.

Constant Throughput Timer(continued)

Attribute:

- Name: Name of timer.
- Comments: Comments of timer.
- Target throughput(in samples per minute):



No of requests that need to be triggered in a

Minute.

Constant Throughput Timer(continued)

Attribute:

- Calculate Throughput based on:
  - This thread only: Each thread maintain its own target throughput -> Overall throughput = target throughput * no of threads.
  - All active threads in current TG: Divide target throughput amongst all active threads of TG, and each thread will delay as needed, based on when it last ran.
  - All active threads: Divides target throughput amongst all active threads in all TG and each thread will delay as needed, based on when it last ran
  - All active threads (shared): Divide target throughput amongst all active threads of TG, and each thread is delayed based on when any thread last run.
  - All active threads in current thread group (shared):Divide target throughput amongst all active threads of TG.

Constant Throughput Timer(continued)

Showcase:

- Demo with constant throughput timer.

Exercise:

- Play with Constant Throughput Timer.

Uniform Random Timer

Definition:

Uniform Random Timer is used to generate the random delay in a uniform manner.

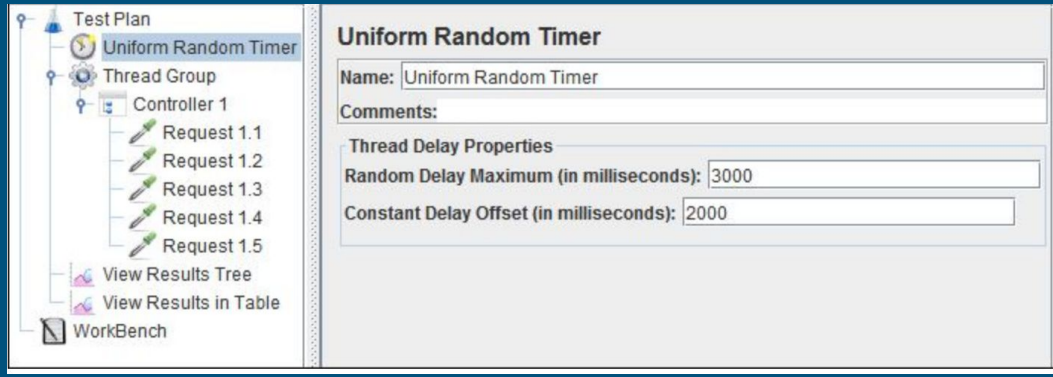
The minimum delay will never be less than constant delay offset.

The delay (think) time is the sum of the generated random number and the constant delay offset value.

Uniform Random Timer

Attribute:

- Name: Name of the timer.
- Comments: Comments for timer.
- Random Delay Maximum (in ms)**:**



The number shows how much maximum

delay can be added to constant delay offset.

- Constant Delay Offset (in ms)**:** The number shows a constant delay which will be added to the generated random number (in the range of given deviation value).

Uniform Random Timer

Showcase:

- Use Uniform Random Timer for demo.

Exercise:

- Do it your own.

Gaussian Random Timer

Definition:

Gaussian Random Timer is used to generate the random delay. This timer is based on Normal or Gaussian Distribution Function. The delay (think) time is the sum of the Gaussian distributed value (with mean 0.0 and standard deviation 1.0) times the deviation value you specify and the offset value.

Gaussian Random Timer

Attribute:

- Name: Name of the timer.
- Comments: Comments for timer.
- Deviation (in milliseconds):

The number shows how much the delay can deviate from the given offset towards higher and lower range.

- Constant Delay Offset (in milliseconds): The number shows a constant delay which will be added in random number generated by Gaussian Function in the range of given deviation value.

Gaussian Random Timer

Showcase:

- Use Gaussian Random Timer for demo.

Exercise:

- Do it your own.

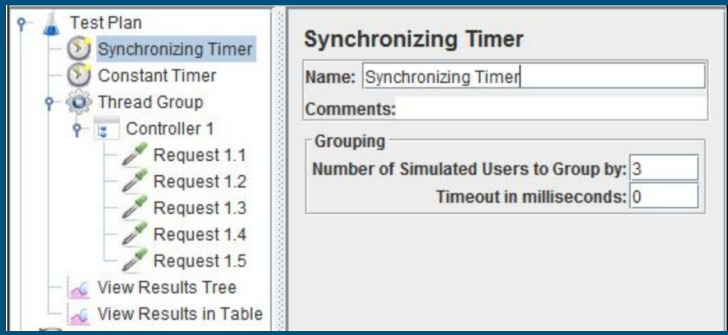
Synchronize Timer

Definition:

The purpose of the Synchronizing Timer is to hold the threads until X number of threads have arrived, and then they are all released at once. It adds delays between requests such that all (defined) threads fire at the same time thus creating heavy load bursts on the application. A Synchronizing Timer can thus create large instant loads at various points of the test plan.

Synchronize Timer

Attribute:



- Name: To provide the name of the timer
- Comments: To provide arbitrary comments (if any)
- Number of Simulated Users to Group by: This attribute is used to define the number of threads to release at a time. If 0 is given then JMeter will release the same number of users which is defined in the thread group.
- Timeout in milliseconds: This value helps to control the timeout of the requests during the test. If 0 is set then the timer will wait for all the threads to reach the value which is defined in "Number of Simultaneous Users to Group". If superior to 0, then the timer will wait till the number given in "Timeout in milliseconds" irrespective of thread count. If wait time exceeds the given timeout,then the VUs are released without meeting "Number of Simultaneous Users to Group" criteria.

Synchronize Timer

Showcase:

- Demo with Synchronize Timer.

Exercise:

- Do it by yourself.

JSR223 Timer

Definition:

JSR223 Timer is a scripting-based timer. You need to implement the thread delay logic by yourself using one of the supported scripting languages like Groovy, BeanShell, java, javascript, jexl etc. Usually, it is helpful when you need to define think times based on some unique algorithm which is not currently provided by JMeter. You can create your own implementation of the algorithm using the JSR223 timer

JSR223 Timer

Demo:

- Use JSR223 Timer to wait

Exercise:

- Do it by yourself.

Throughput Shaping Timer

Definition:

If you have a task to test the throughput of a system at a specified Requests Per Second (RPS) rate. To achieve the desired rate using regular thread groups you would need to play around with the number of threads and timers.

Formula:

Thread Pool Size = RPS * <max response time> / 1000

Throughput Shaping Timer

Pre-requisites:

- Open Plugin Manager.
- Search for Throughput Shaping Timer.
- Install then restart JMeter.

Throughput Shaping Timer

Attribute:

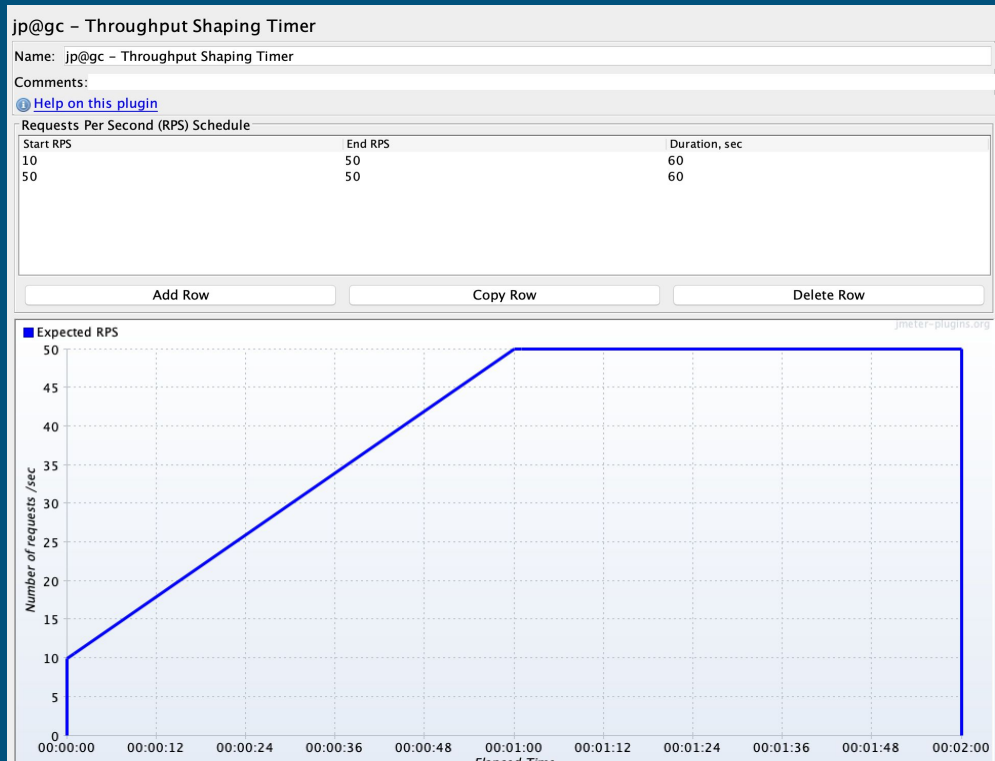
- Name: Name of the timer.
- Comments: Comment for timer.
- Start RPS: Begin number of transaction

per second.

- End RPS: End number of transaction

per second.

- Duration: Time to hold the specific TPS.



Throughput Shaping Timer

About custom function:

__tstFeedback, the schedule feedback function, integrates the Concurrency Thread Group with ‘blazemeter-timer’ Throughput Shaping Timer. 10 is the number of threads at the start. 40 is the maximum allowed number of threads. 10 (the last parameter) is how many spare threads are kept during the test run. The last parameter is interpreted as an absolute count, if it is above 1. It is interpreted as a ratio relative to the current estimate of threads needed, if it is a float value less than 1. then the Ramp Up Time and Ramp-Up Steps Count should be blank.

Throughput Shaping Timer

About custom property:

Another feature is the *“load_profile”* property. This property specifies the load pattern with a set of function-like declarations. It can be used in the *user.properties* or *jmeter.properties* files, or via the command line.

The 3 available types of functions are:

- **const(C, T)** keeps constant load of **C** RPS for **T** seconds;
- **line(C, N, T)** does gradual increasing/decreasing of load from **C** to **N** RPS during **T** seconds;
- **step(C, N, S, T)** does stepping load from **C** to **N** RPS; each step height will be **S** RPS; each step duration will be **T** seconds.

Demo : *jmeter -n -t throughputshapingTimer.jmx -l testresults.jtl -J"load_profile=const(50,60s) line(50,10,60s)”*

Throughput Shaping Timer

Showcase:

- Demo for throughput shaping timer.

Exercise:

- Do it yourself.