Optimize 25 Testing at Scale: Test Scenarios



Prepared by

**Zachery Olson**

Solution Architect

Contributors

**List of Contributors**

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 3/7/2023 | zacolson | 1.0 | Initial draft for review/discussion |
|  |  |  |  |
|  |  |  |  |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Table of Contents

[1 Introduction 5](#_Toc130202788)

[1.1 Scenarios 5](#_Toc130202789)

[1.1.1 Tests 5](#_Toc130202790)

[1.1.2 Ramp-up period 6](#_Toc130202791)

[1.1.3 Pacing 7](#_Toc130202792)

[1.2 External Services 8](#_Toc130202793)

[1.3 Resources 8](#_Toc130202794)

1. Introduction

Test scenarios describe the steps and patterns that will simulate complex and realistic user volume. Apache JMeter refers to these scenarios as a Test Plan.

A test plan will consist of one or more thread groups, logic controllers, sample generating controllers, listeners, timers, assertions, and configuration elements. A thread group is responsible for executing realistic and complex business transactions.

The thread group configuration will include the number of virtual users (threads), ramp-up period, and duration elements that the test will execute against an application.

* 1. Scenarios

In a production situation, different people will be interacting with the system at the same time. For example, a client might interact with an agent using a chat channel. The agent will add notes to the clients’ case using a model-driven app. If the agent is not able to resolve the clients’ questions, the agent may apply routing logic to get the case to the right agent for a follow-up call. This is a factor that needs consideration in your testing strategy.

The samples provided through this guidance simulate load against out-of-box configuration. The sample scenarios provided include:

* Case (incident) lifecycle
* Customer lifecycle
* Unified Routing
* Omnichannel chat lifecycle
* Multi-session application

The samples provided may require additional configuration to execute against a customer’s application. You may define additional test scenarios that fall within the scope for your deployment.

* + 1. Tests

The HTTP sampler elements inside a Thread Group define an individual test that executes realistic business transactions against an application. Each individual test is comprised of a series of steps to simulate the user actions required to accomplish a task. Each step is an interaction that the user has with the platform. These interactions can be navigation, searching for data, or performing an update.

In a test that simulates the steps that a user will take to create a case record, the user will simulate the following actions in the test:

1. User starts from a dashboard or home table.
2. Navigate to the Case table.
3. Search against the table.
4. Open a new case form and enter the necessary data.
5. Saves and closes the new Case.

Each individual interaction may be comprised of one or more HTTP requests to the application platform. It is typical for each business transaction to be represented by 5-10 commands against the platform. These commands can take anywhere from 1 to 60 seconds to execute.

* + 1. Ramp-up period

A Thread Group element controls the number of threads (virtual users) that Apache JMeter will use to execute the test. The **Ramp-up period** configuration on a Thread Group tells Apache JMeter how long it should take to gradually increase the number of thread (virtual users) to the configured number of threads.

The ramp-up period configuration is a duration measured in seconds. The duration should be long enough to avoid a sudden and unexpected surge of traffic at the start of a test. The formula below can help calculate the duration in seconds between successive thread starts:

Ramp-up duration / number of threads = duration between threads

Using this example, if you are going to simulate the traffic of 150 users in a test, and use a ramp-up period of 15 minutes, the formula will determine that a new thread will start every 6 seconds:

900 (15 minutes) / 150 (threads) = one new thread every 6 seconds

There is no prescriptive guidance for a ramp-up duration. Apache JMeter recommends that the duration between threads is short enough that the last threads start running before the first one has finished. A ramp-up period of about 15 minutes is the best practice.

* + 1. Pacing

Apache JMeter, by default, will execute iterations of a test in immediate succession of each other. This is not a realistic scenario, as users will not repeatedly perform a task in a rapid-fire pattern.

Load test **pacing** is a configured delay between the iterations of your tests. Pacing helps you to control the requests sent to an application and generate a realistic usage pattern. This also allows the load test to better simulate the time gap between two sessions and create real-world scenarios that reflect user behavior.

To effectively pace a thread group in Apache JMeter, you can use one of these methods:

* **Constant Throughput Timer:** Use this element to set the Target Throughput property to your desired rate of samples per minute.
* **Pacing Worksheet:** Using a fixed delay timer determined using the Pacing Worksheet.

Use the **Pacing Worksheet** to help you determine the target delay between thread group iterations, and the number of tests per thread (user) to execute over the course of the test. On the Pacing Estimator tab, populate the values described below to generate a fixed configurable delay to use in your test:

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| Threads | Count | Total number of users simulating traffic |
| Loop Count | Count | Total number of test iterations per thread |
| Avg Thread Duration | Duration | Duration of a single thread iteration in seconds |

Apache JMeter does not natively pace by the number of tests per user. This can be accomplished manually with the **Configurable Pacing** outputs of the pacing worksheet, and a **Constant Timer** or **Flow Control Action** element. Both elements can pause a thread by a fixed value in milliseconds.

* A **Constant Timer** will run in the context of a sampler. The timer will delay a sampler by a fixed value in milliseconds. Using a constant timer, the pacing delay is the sum of the configured delay and the execution time of the sampler. WhoAmI() is a great request to execute with a constant timer.
* A **Flow Control Action** can run the context of a transaction controller. The pause configured in the flow control action is not dependent on any sampler and therefore is static.

There are other methods that can be employed using think time or third-party plugins for Apache JMeter to accomplish the appropriate pacing in your test.

* 1. External Services

Integrations with other systems need consideration in a load testing scenario. The scope of load testing scenarios often does not directly evaluate the performance of integrations under load.

Digital Contact Center Platform will take a dependency on communication channels that give customers the choice to connect in ways that work best for them. Engagement channels include:

* Voice and telephony
* Chat
* Email
* Virtual Assistants
* SMS
* Social messaging

These channels integrate with the platform but are based on their own architecture and APIs that help customers integrate communicating into their applications.

Activating any integration that will be involved in the user scenarios that are being load tested is the best practice. This will allow your test to simulate realistic load on the integrated systems and check their performance and scalability. This can apply to any non-Dataverse connector or interface that is external to the Power Platform. Indirectly testing external services will help avoid potential bottlenecks and failures that might occur when the integrations are exposed to high volume and concurrency.

Also consider that you are working with test data. Exercise caution in sending load test data to production integration endpoints. For example, sending load test emails to customers, validating fake addresses, or sending SMS messages to real customers during a load test.

* 1. Resources

The Elements of a Test Plan - <https://jmeter.apache.org/usermanual/test_plan.html>

Azure Communication Services - <https://azure.microsoft.com/en-us/products/communication-services/>

Omnichannel Channels - <https://learn.microsoft.com/en-us/dynamics365/customer-service/channels>