The Pega® Platform is different to the standard Java application in a number of ways; however, the approach to performance testing and tuning Pega 7 is no different. There are a number of key considerations that you need to take into account when setting up a performance test to ensure that each test is successful.

**Key considerations**

- Define the Application Usage Model (AUM) and evaluate with Business to make sure it is realistic.
- Create the scenarios from the key functional areas as detailed in the AUM.
- Evaluate test data requirements and validate.
- Create test user IDs with correct roles and privileges.
- Do not process more work than expected in any time period.
- A common mistake can be to process a whole day’s volume in 1 hour.
- Tune and optimize the chosen scenarios.
- Use Pega 7 tools PAL / DB Trace to identify issues.
- Fix issues and validate fixes, i.e. new DB index and query tuning.
- Baseline scenarios on a single JVM.
- Validate performance metrics for low user concurrency (10-50).
- Validate test data.
- Increase concurrent usage to evaluate optimal user concurrency per JVM.
- Don’t use a single test ID to test for concurrent user scenario; this can lead to other unwanted errors.

**Test approach**

The below diagram illustrates the approach that is used for performance testing a Pega 7 application.
Define success criteria

The following key performance indicators (KPIs) should be considered as applicable to any Pega 7 application. The tested application must be performing within the bounds of these for the test to be classed as successful. The examples below are listed for illustration only. You need to define the KPIs after you speak with the client team and understand the application Use Cases.

Run the test cases before and after each planned release to see if the system is getting better or worse with new code and fix the issues before moving new code to production.

<table>
<thead>
<tr>
<th>KPI</th>
<th>Example KPI value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average transaction response time</td>
<td>&lt;3 seconds</td>
</tr>
<tr>
<td>90th percentile transaction response time</td>
<td>&lt;10 seconds</td>
</tr>
<tr>
<td>Transaction failure rate</td>
<td>&lt; 1% of transaction fail</td>
</tr>
</tbody>
</table>
Performance testing

**Test objective**

The aim of the test is to measure the ability of the environment to process the selected business functions within the specified timeframes or rates of concurrency detailed in the service level agreements (SLA). The tests detailed below should be carried out during this test phase if applicable. The business or other requirements will usually determine this.

<table>
<thead>
<tr>
<th>Performance testing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU utilization of application server</strong></td>
<td>&lt; CPU should not exceed 50% benchmark tests, 80% peak hour</td>
</tr>
<tr>
<td><strong>Java memory of application server</strong></td>
<td>&lt;70% of the available heap</td>
</tr>
<tr>
<td><strong>Time spent garbage collecting</strong></td>
<td>&lt;1% of total processing time</td>
</tr>
</tbody>
</table>

**Load test**

A test type concerned with measuring the behavior of a component or system. With increasing load, e.g. number of parallel users and/or numbers of transactions to. Determine what load can be handled by the component or system. This test can be used to ‘break’ the component or system.

**Expected response times**

Tests that the system provides users with responses to their requests in accordance with the performance requirements, defined KPIs and SLAs.

**Batch run time**

Ensures that any Batch processes required execute successfully and in acceptable timeframes.

**Stress test**

Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements. This test can also be used to determine how the system or
Performance testing

Component handles variations in peak load. Testers may use this test to try to “break” the component or system.

Soak test
Ensures that performance bottlenecks do not occur when the application is run for a long period of time with typical (or peak) load (e.g., looks for memory leakage). This testing will normally mimic a typical production day or days and must include any and all background processing in an application.

Scale test
Testing conducted to evaluate how an application will perform at scale in the client’s environment. The aim is to prove linear scalability within the confines of the testing environment. Extrapolation of results may be required if the testing environment is not identical to production.

An additional aspect of scale testing will be testing of regional usage of the application to ensure that an acceptable level of performance is achieved for users in different geographical locations or on terminal emulation software.

Tasks for performance testing
Define entry and exit criteria
Entry criteria
• Test scripts exist.
• Controlled testing environment for Non-Functional testing is configured and available.
• The test environment is uploaded with the application to be tested.
• Production like test data is available for test execution.

Exit criteria
• Test results for each test run and executive summary report.
Performance testing

Preparation (pre-test validation phase)
Prior to undertaking the performance testing process a number of validation steps need to be undertaken to ensure that the following:

- Pega 7 has been correctly deployed.
- Pega 7 is operating in a stable manner.
- Supporting infrastructure is in place to ensure testing consistency.

Prepare the test data and test IDs

- Data volumes need to be scaled in relation to the relative size of the test and production systems.
- The minimum test data required for each scenario will need to be estimated; and the data should be provided.
- Test data will need to be made available for Batch test executions.
- Existing data needs to be in the correct state to ensure that background processing can also be executed.
- Test IDs need to be setup with correct roles and privileges.
- System should be tested with number of test IDs, with single test ID load test, results can be misleading and we can see some unwanted errors in logs.

Pega 7 PAL and DB trace review

- Identify the processes that are to be executed as part of the test and perform Pega 7 PAL and DB Trace analysis to identify easy tuning opportunities.
- Correct any issues identified in the PAL and DB Trace analysis, i.e. missing DB index, query tuning, blob fetch, or any service call that can run in parallel rather than running sequentially. Look for Pega0004 and Pega0005 alerts in the alerts logs.
- Try looking the data being fetch in single call or rendered on single screen and try to work with business to find out if the screen or process can be redesigned. Look for Pega 0001 alerts in the alert logs.
- Iterate through the process until all significant issues have been corrected.

Baseline and tune

Baseline the Pega 7 application based on a single user performing the automated scripts.

- This is the best case performance.
Performance testing

Baseline the Pega 7 application based on a large subset of users (50 or 100) performing the automated scripts.

- Identify performance bottlenecks and fix them.
- Tune and re-baseline.

**Approach overview**

- Prepare and audit the test environment (System Test, Clone, Pre-Production).
- Complete, review, and agree upon the engagement document with all stakeholders.
- Build simulators, test harnesses, and any stubs required.
- Define AUM.
- Setup monitoring and diagnostic tools.
- Verify stability of performance environment.
- Build benchmark scenarios and undertake performance optimisation.
- Execute acceptance testing.
- Analyse and compile results.
- Tune application and environment, re-baseline, and retest.
- Execute assessment testing.
- Analyse and compile results.
- Produce interim report on findings.
- Implement recommendations.
- Produce final report and recommendations.

**Test data requirements**

There are certain elements of the testing process that need to be documented in separate document(s). The main four elements of data are listed below:

- AUM
  - Quantitative details of how the application is expected to be used in production detailing the split of processing by frequency and user volumes
- Reference data
  - Details of the data required to ensure that the system functions
Performance testing

- User log on and passwords
  - Details of the user login ID, password, and user roles
- Test data
  - Details of the data required to enable the scripts to be executed including how the data is to be created

Test results
The following artefacts are required to evaluate the outcome of each iteration of the performance test.

- Pega 7 application and alert log files
- Pega 7 log usage data
- Database statistics data
  - Oracle stats pack reports
    - AWR
    - ADDM
    - ASH
- Operating System data
  - CPU utilization
  - Memory utilization
- JVM memory
  - Verbose Garbage Collection data
- Network utilization data

It is a best practice to ensure that each testing iteration starts from the same set of data and that the log files listed above are deleted before each iteration. When using Oracle, the Flashback functionality should be considered as an option to ensure each test starts from the same point in time.

Before execution of each testing phase, the Pega 7 and WebSphere will need to be restarted to ensure the log files represent only the elapsed time for each test.
Frequently Asked Questions

Our organization is new to the Pega 7 Platform; what are a few performance tools that have been used by other clients in the past and supported by Pega?

JMeter and LoadRunner are a few tools that have been used by clients in the past for performance testing. Pega 7 supports a wide range of software for testing, please work with your Pega representative for any specific questions. Here are a few examples:

**JMeter**

- Apache JMeter is a 100% pure Java desktop application designed to load test client/server software (such as a web application).
- It may be used to test performance both on static and dynamic resources, such as static files, Java Servlets, CGI scripts, Java objects, databases, and FTP servers.
- JMeter can be used to simulate a heavy load on a server, network, or object to test its strength or to analyze overall performance under different load types.

**Types of Test Plans Supported with JMeter**

- Building a Web Test Plan
- Advanced Web Test Plan
- JDBC
- FTP
- JMS Point-to-Point
- JMS Topic
- LDAP
- LDAP Extended
- WebServices (SOAP)

**LoadRunner**

HPE LoadRunner is a software testing tool from Hewlett Packard Enterprise. It is used to test applications, measuring system behavior and performance under load. HPE LoadRunner can simulate thousands of users concurrently using application software, recording and later analyzing the performance of key components of the application.
LoadRunner simulates user activity by generating messages between application components or by simulating interactions with the user interface, such as keypresses or mouse movements. The messages and interactions to be generated are stored in scripts. LoadRunner can generate the scripts by recording them, such as logging HTTP requests between a client web browser and an application's web server.

**Our organization has completed the first RPA implementation using Pega. How can we make sure the performance of the system will not be an issue in production?**

An RPA or RDA application doesn't need any special performance testing. Unlike other technologies, Robotics simply overlays the applications that are currently in place. So in simple terms, the "robots" are doing nothing more than pressing the same buttons within any given application that a user would press when performing the same transaction sequence. Load testing is really dependent upon the speed and performance of the applications that we lay on top of. RPA/RDA applications can move no faster than the speed of the underlying applications and we can always move faster than those apps. So load burden does not typically fall to RPA/RDA implementation.

In the field, when doing something like an RPA (aka - "batch") transaction, it's always good practice to test with higher volumes just to make sure nothing crashes. But since the robots just keep the load on the underlying systems, things often are "perceived' as being better. But that's mostly because the processing can happen off hours, or more robots can be assigned to do the same work as long as the underlying applications can handle more robots pecking at the keys without contention.

**Our organization has completed first implementation of outbound marketing. We can test the campaign using seed and distribution tests, but how can we do the load testing?**

In addition to multi-channel Next-Best-Action, Pega Marketing enables the creation of simple outbound campaigns in which a message can be sent to a target audience over a single outbound channel. Testing the outbound campaign is tricky, as rules are not tested with the same kind of data needed by an actual campaign.

Here are some open source tools that can be used for performance testing of the email outbound marketing campaigns:

**FakeSMTP**

FakeSMTP is a Free Fake SMTP Server with GUI for easily testing emails in applications. It is written in Java. Configure your application to use "localhost" as your SMTP server, and all emails
Performance testing

will be intercepted and displayed in this software. Please visit http://nilhcem.com/FakeSMTP/ for more information.

DevNull SMTP

DevNullSMTP is a dummy SMTP server that can be used for testing purposes. It helps you see all communication between a client and the server and is very useful if you are trying to find problems with your email server or a client that you wrote. Please visit http://www.aboutmyip.com/AboutMyXApp/DevNullSmtp.jsp for more information.

Ten guardrails for successful performance load testing: