Performance Metrics

Performance Summary

OWF 6 includes performance metrics that test the application's responsiveness and stability under both real world and stressful workloads. The results for a realistic scenario with 400 concurrent users doing common OWF actions can be found in the [Realistic Scenario Test](#) section.

The most pertinent aspects of this report include the following:

- JMeter was used to simulate a realistic end-user scenario and individual request stress tests
- A maximum of 2,000 users (400 concurrent) is suggested for a non-clustered OWF
- Profiling of test systems was used to determine bottlenecks for future improvement to OWF
- Apart from OWF, widget performance presents a possible bottleneck
- Collaboration is encouraged as the OWF team's testing scripts are available to the community

Performance Environment

OWF established virtual testing environments with the intent of providing a basis for users to estimate their system's performance at various load levels. To do this, OWF tested the performance of its web application by impersonating a browser and hitting the server with a comprehensive range of request URLs that occur during a user's OWF session.

OWF used the performance testing tool JMeter to submit the request URLs and track the time it takes the server to respond. JMeter enabled the simulation of a realistic scenario where more than three hundred unique users were simultaneously launching requests at selected URLs in OWF. The individual requests and their response times were recorded and used to generate reports. These reports allowed the analysis of the server's general performance, as well as the performance of each individual request, enabling the identification of components in OWF whose performance could be improved to increase the application's user limit.

The OWF performance tests used multiple virtual machines to approximate a real world production system tuned to achieve the best possible performance for a non-clustered instance of OWF. The JMeter tests ran on a client machine that acted as a web browser to submit simple HTTP requests to the OWF application server. The application server stored data on a MySQL database server.

Figure 1: OWF Testing Environment

While real production systems may utilize separate, high-performance machines to host the system, the OWF performance results are only intended to provide a basis from which users can estimate their system's performance. As a result, the OWF performance testing environment consisted of virtual machines on an Intel Xeon 2.6 GHz server. The virtual test machines were also limited to finite resources on the server due to additional virtual machines unrelated to OWF performance testing being run in tandem on the server. The following table lists the resources dedicated to the virtual testing machines:

Table 1: OWF Testing Environment Specifications

Server	CPUs	RAM	OS	Software
Database Server	4 x 2.6 GHz	4 GB	CentOS 5.6 64-bit	MySQL 5.5
OWF Server	4 x 2.6 GHz	8 GB	CentOS 5.6 64-bit	Tomcat 7.0.21 Sun Java JDK 1.6u14
JMeter Server	4 x 2.6 GHz	8 GB	CentOS 5.6 64-bit	JMeter 2.6 Sun Java JDK 1.6u14
Metric Server	4 x 2.6 GHz	4 GB	CentOS 5.6 64-bit	Tomcat 7.0.21 Sun Java JDK 1.6u14

Note: Full configuration for these machines can be found in the Appendix of this document.

Realistic Scenario Test

In order to test OWF's performance in a realistic environment, JMeter was used to simulate a peak load of 400 users simultaneously performing the most common end-user actions. To approximate typical usage, the OWF Development Team quantified a distribution for these end-user actions. For example, it was assumed that out of 100 OWF actions performed by a user, 35 of the actions are to open a widget, 20 are to switch dashboards, and 15 to add a widget from Marketplace. The full distribution of user actions is visible in Table 2.

To begin the test the users were logged in using a uniform, random method over a period of 60 minutes. Of these users, 10% were not yet in the database in order to account for new

users in a real system. Following login, each user began executing the 100 common end-user actions at a pace independent of the other users in the test, following the action distribution strictly (i.e. each user opened a widget 35 times, switched dashboards 20 times, etc.). These actions were performed in a random order and were randomly delayed between 5 and 60 seconds to capture varying speeds a real user sends server requests. Following the completion of all 100 actions, the user logged out. After every user finished all 100 actions and logged out, the test was complete.

The test took 2 hours and 49 minutes to complete, in the middle of which there was an approximately 45 minute period where all users were logged in and executing actions, ensuring the server was tested at the maximum load of 400 users for a significant time span.

Table 2: Realistic Scenario Test Results

Testing Scenario	Distribution of User Actions	Requests/ Sec	Average Response Time (sec)
Login User authenticates and is added to the database, if a new user	Once Per User	0.11	.507
Launch Widget User launches a widget which sends a metric and the load time to OWF	35%	1.50	.546
Switch Dashboard User switches dashboards	20%	0.84	.154
Add Widget User opens a Marketplace widget and adds a new widget to OWF	15%	0.64	1.011
Create Dashboard User creates a new dashboard	10%	0.44	.100
Launch Menu User opens the Launch Menu which loads all assigned widgets	5%	0.23	.455
Edit Dashboard User edits and saves a dashboard	5%	0.23	1.279
Open Help User clicks the help icon from the toolbar	5%	0.23	.017
Edit Widgets User edits widget tags and visibility preferences	3%	0.15	.520
Change Theme User changes themes	2%	0.11	.785
Logout User logs out of OWF	Once Per User	0.09	.068

Note: For the cumulative test the average response time was 0.498 seconds with an average load of 4.2 requests per second.

Widget Performance

The overall performance of the system depends not only on OWF but the widgets that run inside the framework. The performance findings reflected in this report only outline performance of OWF—not the widgets that are running inside it. For optimal performance of the entire system, the OWF team recommends that widget developers spend time tuning their widget performance. This optimization effort can be better targeted by setting up an OWF Metric server to communicate with the OWF system, which will record widget usage data for easy identification of the most used widgets.

Commitment to Continuous Improvement

This OWF report will continue to evolve in future releases as the testing scripts are improved to better capture the performance of OWF and optimizations are implemented as a result of the findings. A future area of interest for the OWF Development team is to test how the performance can be increased through clustering. In addition, new test scripts will be created to quantify the performance of any new functionality included in OWF.

Appendix A   Test System Specifications

Below are the detailed specifications of the testing equipment, software, and scenarios.

OWF and Metric Servers

Table 3: Tomcat Configuration

Tool	Version
Tomcat Version	7.0.21
Java Version	1.6.0
Java Heap Size/ Max Heap Size	2 GB / 2 GB
Java Perm Size/ Max Perm Size	128 MB / 512 MB
Max Threads	800
Max Keep Alive Requests	100

Database and Data

Table 4: Database Configuration

Database	Version
MySQL	5.5.21

Table 5: Database Objects

Object Description	Quantity
Users	1,000 (995 standard users; 5 administrators)
Groups	50 (5 per user)
Widgets	500 (10 per user; 5 per group)
Dashboards	5,050 (5 per user; 1 per group)
Preferences	50,000 (50 per user)

JMeter and JMeter Scenarios

Table 6: JMeter Configurations

Description	Value
JMeter Version	2.6
Java Version	1.6.0
Java Heap Size/ Max Heap Size	768 MB / 4 GB
Java Perm Size/ Max Perm Size	64 MB / 512 MB

JMeter Scenarios

For the JMeter Scenarios, users logged on approximately every 11 seconds. Once users logged on they began executing operations per the distribution in Table 2: They executed an action and then waited a random time before executing the next action. The time waited was a Gaussian distribution averaging around 33 seconds with an approximately 28 second variation. Once a user executed 100 operations they logged out. After all the users were logged out, the individual response times of each operation was tallied and used to create overall statistics.

It is also important to note regarding the actions in the realistic scenario that they do not necessarily correspond to single URL requests—in fact, the Change Theme action executes at least nine distinct URL requests. In addition, the requests/sec defined in Table 2 is not an actual average of samples gleaned from simulation logs. It is computationally derived from input parameters as such:

$$\frac{n \text{ users } (\% \text{ operation})}{\text{average delay time}}$$