v13.1.0.1443

05.June.2020

Performance report

Blueriq forms
with a Redis session store

blueriq

Disclaimer

The results in the following report have been achieved on a specific hardware, software and application combination. The results in this report may differ from results on other combinations of hardware, software, custom code and application settings. No rights can be derived from this document.

It is strongly recommended to perform appropriate performance tests on the application that is modelled with Blueriq before taking it into production.

Introduction

Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload.

This report provides an overview of the performance tests for Blueriq 13.1 when using a forms application with a Redis session store. The purpose of this report is to give insight into the performance of such a Blueriq application on a typical hardware configuration and to validate the use of the new session store component. Based on this report, you should be able to review the performance characteristics of your own application in production and to assess the infrastructure adequacy.

How to use this report
Testing and reporting on performance is complex since it depends on a large number of variables. In the documentation on the Blueriq Community you can find the test approach and assumptions for the tests. It contains the reference application, user scenarios, the test environment, the test methodology, key performance indicators and the acceptance criteria.

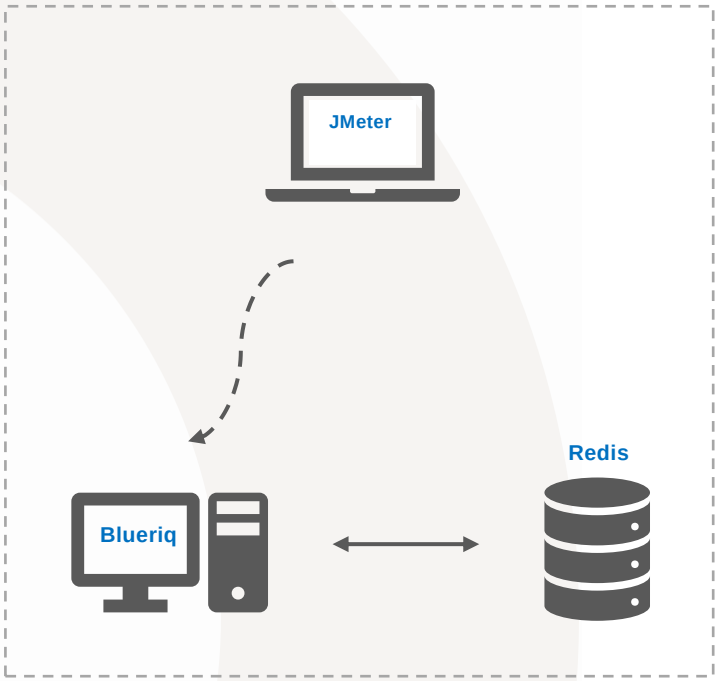
Hopefully, the results from this performance test can help you to estimate the hardware configuration required to support your application when “going live” to production operation.

Load testing is the simplest form of performance testing and its goal is to see how the software behaves under normal use circumstances. Except for the situations where we introduce new modules or make breaking changes to one of our existing ones, the load testing is the way to go for Blueriq in order to check the application’s current shape.

blueriq

Test environment

Three separate virtual machines were used in this test: one for the Blueriq Runtime, one for the Redis server and one for the JMeter application which simulates the user load and runs the performance script.



JMeter machine:

- OS Windows Server 2016 Standard
- Intel® Xeon® CPU E5-2680 v2 @ 2.80GHz 2.79GHz (2 processors)
- 2,00 GB memory (RAM)
- 39,10 GB HDD

Blueriq machine:

Server hardware

- Intel® Xeon® CPU E5-2680 v2 @ 2.80GHz 2.79GHz (4 processors)
- 5 GB Memory
- 39,10 GB HDD

Server OS/AS/Database

- Windows Server 2016 Standard
- JBoss 7.2 EAP
- JAVA 1.8.0 Update 151

Redis machine:

- OS CentOS Linux 7 (Core)
- Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz
- 2,00 GB memory (RAM)
- 16,00 GB HDD

Test results

The table below shows the results of all individual steps of the performance script. Please note that this test simulated a constant user load of 800 users during a one hour run. The scenario is also explained on the community page.

For each step (Key Performance Indicator) the following values are given:

- the T-value used for calculating the Apdex value
- the Apdex value with the colour indicating the rating

Test step name	*Actions*	*T-Value (ms)*	*Apdex value R12.9*	*Apdex value R12.10*	*Apdex value R12.11*	*Apdex value R12.12*	*Apdex value R13.0*	*Apdex value R13.1*
1. Login	Log in with a username and password	500	**1,00**	**0,99**	**1,00**	**1,00**	**1,00**	**1,00**
2. Start Project	Start the application and display the welcome screen.	500	**1,00**	**1,00**	**0,99**	**1,00**	**1,00**	**1,00**
3. Open ATMs section	Switch to the *Find Local ATMs* page.	500	**1,00**	**0,99**	**0,99**	**1,00**	**1,00**	**1,00**
4. Search ATMs	Creates 500 instances using the AQ_CsvConnectivityService and an external CSV file, then switches to the *Confirm ATMs Found* page.	500	**1,00**	**0,99**	**1,00**	**1,00**	**1,00**	**1,00**

5. Display search results	Open the Display ATms page where the previously created instances are displayed in an *AQ_InstanceList* with page size 20.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
6. Go to loan offers page	Navigate to the Loan Offers page. The page contains dozens of containers and text items, each of them having presentation or content styles assigned to them.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
8. Apply for prime offer	Updates an instance through the *AQ_InstanceUpdate* service and navigates to the Loan Form page.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
9. Open calculator	Opens the *Borrowing Calculator* page;	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
10. Fill in fields (1)	Fills in the available page fields. Other conditional fields are presented to the user because of this action.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
11. Fill in fields (2)	Fills in the newly available page fields. New fields are displayed because of this action.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**

12. Fill in fields (3)	Fills in the newly available fields on the page.	500	**1,00**	**1,00**	**1,00**	**0,99**	**0,99**	**1,00**
13. Calculate	Updates 3 instances belonging to separate entities using the *AQ_InstanceUpdate* service and displays the calculation results to the user.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
14. Go to the first page	Returns to the welcome page.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
15. Open the loan offers page	Navigates to the Loan Offers page.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
16. Apply for a fixed rate	Updates an instance through the *AQ_InstanceUpdate* service and navigates to the Loan Form page.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
17. Fill in fields (4)	Fill in the available fields on the page. Some fields are refresh fields and have validations.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
18. Fill in fields (5)	Fills in the available fields on the page. Some fields are refresh fields and have validations.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
19. Fill in fields (6)	Fills in the available fields on the page. Some fields are refresh fields and have validations.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**

20. Continue	Navigate to the *Personal Information* page.	500	**1,00**	**1,00**	**1,00**	**0,99**	**0,99**	**1,00**
21. Fill in personal form	Fill in all fields and navigate to the *Document Upload* page.	500	**1,00**	**1,00**	**0,99**	**1,00**	**1,00**	**1,00**
22. Upload file	Upload a file to certify the user's identity. As the upload is successful an *AQ_FileDownload* container is generated on the page.	500	**1,00**	**1,00**	**0,99**	**1,00**	**1,00**	**1,00**
23. Download file	Download the previously uploaded file.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
24. Submit request	Navigate to the *Confirmation page*.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
25. Go to the loan offers page	Navigate to the *Loan Offers page*.	500	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**	**1,00**
26. Log out	The user logs out	500	**1,00**	**1,00**	**0,94**	**0,5**	**0,5**	**0,49**

Conclusion

The results look good with the exception of the log out step which has a low response time when using Redis. We will investigate what is causing this.

As new 13.x releases will appear this document will be updated with new performance figures and possible explanation and analyses on the performance variations