IBM MQ 9.1.x.0

XMS .NET Core Performance Report for Windows and Linux

Configuration and Measurements for the following products:

IBM MQ 9.1.x.0

IBM Corporation
IBM MQ
February 2020

Please take Note!
Before using this report, please be sure to read the paragraphs on “disclaimers”, “warranty and liability exclusion”, “errors and omissions” and the other general information paragraphs in the "Notices" section below.

First Edition, February 2020.
This edition applies to the XMS .NET component of IBM MQ for Linux and Windows V9.1.x.0 (and to all subsequent releases and modifications until otherwise indicated in new editions).

Notices
DISCLAIMERS
The performance data contained in this report were measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends on the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION
The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS

The information set forth in this report could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

INTENDED AUDIENCE

This report is intended for architects, systems programmers, analysts and programmers wanting to understand the performance characteristics of the XMS .NET component of IBM MQ for Linux and Windows V9.1.x.0. The information is not intended as the specification of any programming interface that is provided by WebSphere. It is assumed that the reader is familiar with the concepts and operation of the IBM MQ V9.1.x.0 XMS.NET component.

LOCAL AVAILABILITY

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES

Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS AND SERVICE MARKS

The following terms used in this publication are trademarks of International Business Machines Corporation in the United States, other countries or both:

- IBM

- DB2

Other company, product, and service names may be trademarks or service marks of others.

EXPORT REGULATIONS

You agree to comply with all applicable export and import laws and regulations.

How this document is arranged

Performance Headlines

Pages: 2-5

Chapter 2 details the performance headlines for XMS .NET applications. Each scenario is detailed fully with description in this section. The headline tests show how many messages put per second i.e., throughput.

We detail the number of messages put per second in each scenario for different size messages and other parameters.

Best Performance Achieved

Pages: 6-7

Chapter 3 displays the best performance achieved by XMS .NET applications.

Tuning Recommendations

Pages: 8

Chapter 4 discusses the appropriate tuning that should be applied to queue managers.

Test Environment

Pages: 9

Chapter 5 gives an overview of the environment used to gather the performance results. This includes a detailed description of the hardware and software.

Contents

Figures

Tables

1 Overview

.NET applications are developed using .NET Core framework to connect to IBM MQ queue manager. This report consists performance of XMS .NET Core applications.

This performance report details IBM XMS .NET Core applications in a range of scenarios, giving the reader information on number of messages put per second by XMS .NET client on queue manager. The report is based on measurements taken from client running on the Linux Server operating system and Microsoft Windows Server 2016 Standard when queue manager is running on Linux server.

At the end of each block of results is a summary of the findings. It should be noted that results obtained, and the inferences made depend on the test infrastructure hardware and any change could alter the results significantly. The reader is urged to use the findings in this report only as guidelines.

2 Performance Headlines

This section consists of different scenarios which are explained with detailed description, diagram and results.

The measurements for the performance headlines are based on the following:

- Number of messages put per second by XMS .NET client on queue manager.
- Number of messages received by XMS .NET client from queue manager.

The applications are built using .NET Core framework which can be run on Windows and Linux. Each scenario has run with different combinations by varying message size, persistence and sharing conversations. The following parameters are as follows:

Message Sizes:

- 256 bytes
- 512 bytes
- 1024 bytes or 1 KB
- 2048 bytes or 2 KB

Persistence:

- Non – Persistent
- Persistent

Transportation Mode:

- Managed

Sharing Conversations:

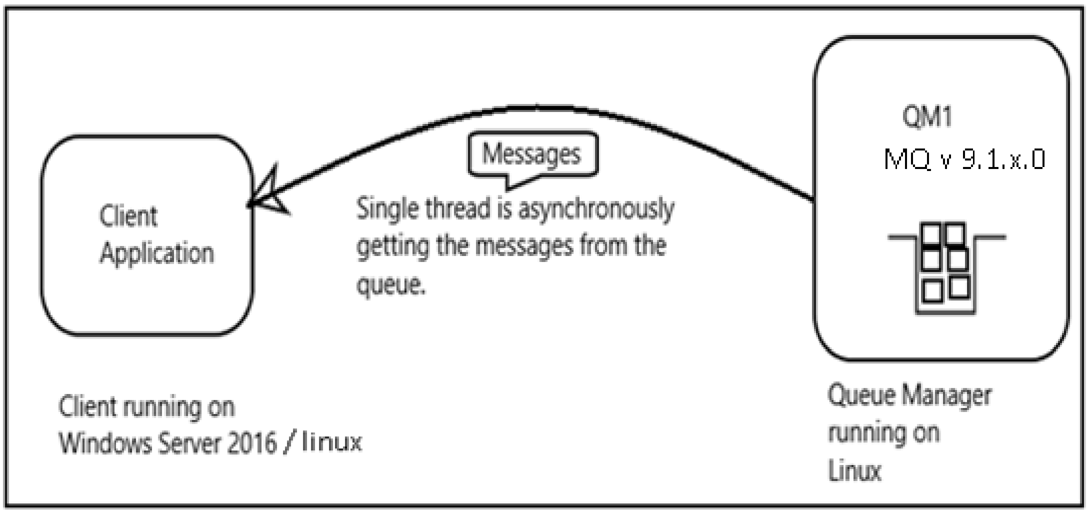
- Sharecnv=10
- Sharecnv=1

For example, when running a scenario, above mentioned parameters are taken into consideration and results have been captured. For every scenario, introduction of the scenario, results in form of tables are updated in report.

2.1 *Async Consume Scenario*

2.1.1 Introduction

A XMS .NET Consumer Application which is a single threaded application which uses message listener to asynchronously consume 10k messages from a queue.


QM1
MQ v 9.1.x.0
Messages
Client Application
Single thread is asynchronously getting the messages from the queue.
Client running on Windows Server 2016 / linux
Queue Manager running on Linux

2.1.1.1 Windows

	SHARECNV=10	
	MANAGED	
Message Size	Persistent	Non-Persistent
256 bytes	957	2288
512 bytes	907	1934
1 KB	813	1808
2 KB	826	1464

Table 1 Performance report for Scenario 1 on windows

2.5.1.2 Linux

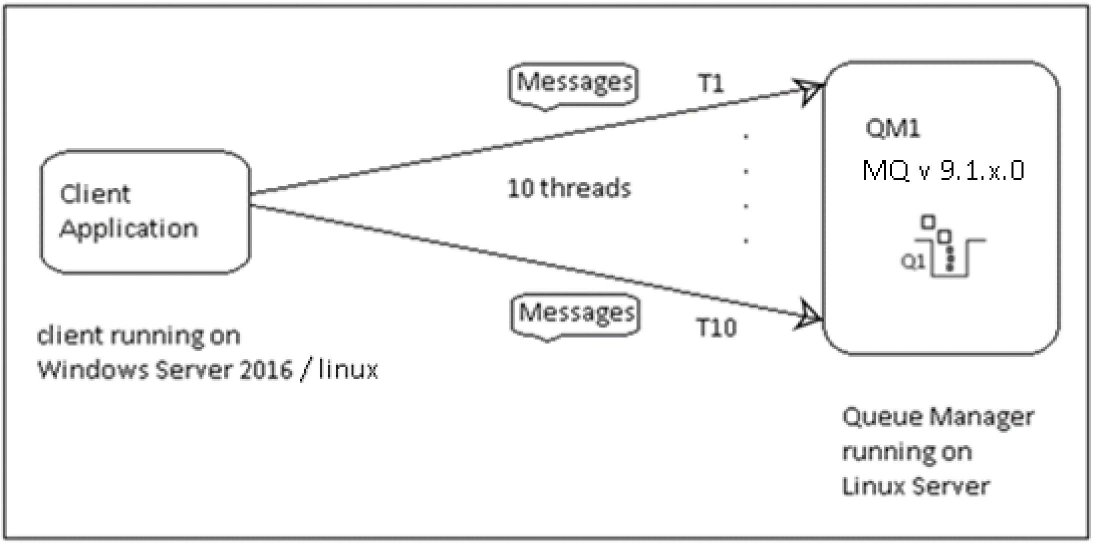
	SHARECNV=10	
	MANAGED	
Message Size	Persistent	Non-Persistent
256 bytes	422	826
512 bytes	382	563
1 KB	362	687
2 KB	360	683

Table 2 Performance report for scenario 1 on Linux

2.2 *XMS .NET 1 Queue – 10 Threads PUT Scenario*

2.2.1 Introduction

A XMS .NET multi-threaded put application which is connecting to 1 Queue - 1 Queue Manager using 10- threads. Each thread puts 5k messages as a warmup. And then each thread puts 10k messages on queue to capture performance statistics.



2.2.1.1 Windows

	SHARECNV=10		SHARECNV=1	
	MANAGED		MANAGED	
Message Size	Persistent	Non-Persistent	Persistent	Non-Persistent
256 bytes	3009	11300	8217	11507
512 bytes	3125	9099	7752	9345
1 KB	3160	6514	5784	6562
2 KB	2631	4310	3595	4333

Table 3 Performance report for Scenario 2 on windows

2.2.1.2 Linux

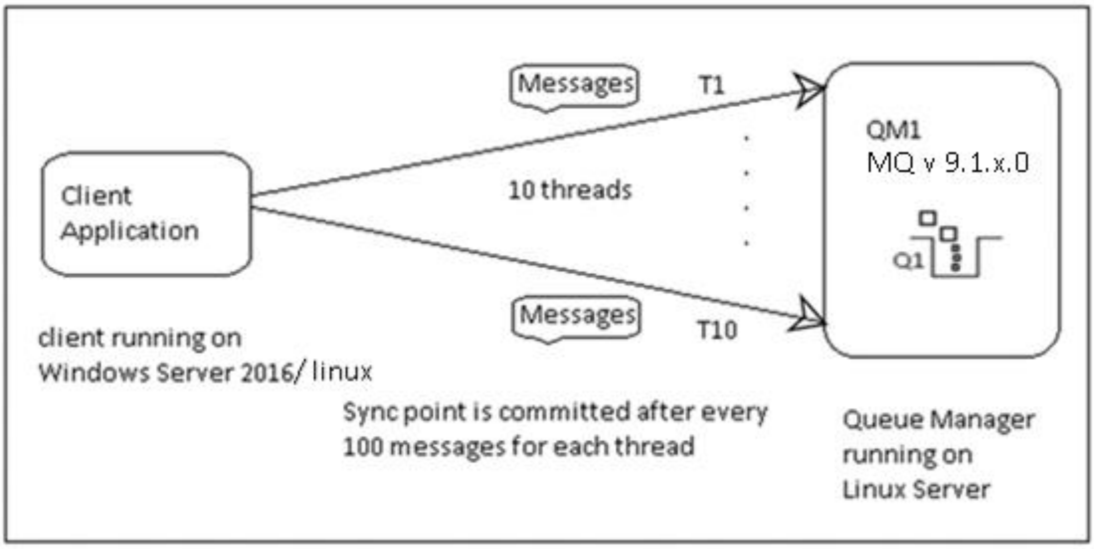
	SHARECNV=10		SHARECNV=1	
	MANAGED		MANAGED	
Message Size	Persistent	Non-Persistent	Persistent	Non-Persistent
256 bytes	3105	4762	4545	5650
512 bytes	3030	4347	4347	4761
1 KB	2932	4065	3875	4728
2 KB	2617	3802	3390	4609

Table 4 Performance report for scenario 2 on Linux

2.3 XMS .NET Sync Point scenario

2.3.1 Introduction

A XMS .Net Put application which is connecting to 1 Queue - 1 Queue Manager using 10 threads. This scenario covers messages put under sync point. A commit being issued after every 100 messages. Each thread puts 5k messages as a warmup. And then each thread puts 10k messages on queue to capture performance statistics.



2.3.1.1 Windows

	SHARECNV=10		SHARECNV=1	
	MANAGED		MANAGED	
Message Size	Persistent	Non-Persistent	Persistent	Non-Persistent
256 bytes	10246	11111	10504	11173
512 bytes	7698	9132	8598	9216
1 KB	5995	6419	6211	6472
2 KB	3607	4248	4060	4273

Table 5 Performance report for Scenario 3 on windows

2.3.1.2 Linux

	SHARECNV=10		SHARECNV=1	
	MANAGED		MANAGED	
Message Size	Persistent	Non-Persistent	Persistent	Non-Persistent
256 bytes	4484	4762	5714	5882
512 bytes	4425	4673	5050	5347
1 KB	4262	4545	4878	4975
2 KB	3703	4000	3921	4545

Table 6 Performance report for scenario 3 on Linux

3 Best Performance Achieved

3.1 XMS .Net – Windows

Sync Point 1 Queue – 10 Threads PUT scenario

A XMS .Net Put application which is connecting to 1 Queue - 1 Queue Manager using single thread and 10 threads. This scenario covers messages put under sync point. A commit being issued after every 100 messages. Each thread puts 5k messages as a warmup. And then each thread puts 10k messages on queue to capture performance statistics.

Scenario	Mode	Message Size	Persistence	Share Conversations	Throughput
Sync Point - 1 Queue, 1 Queue Manager, 10 threads	Managed	256 Bytes	Persistent	1	10504 messages/second

XMS .NET 1 Queue – 10 Threads PUT Scenario

A XMS .NET multi-threaded put application which is connecting to 1 Queue - 1 Queue Manager using 10- threads. Each thread puts 5k messages as a warmup. And then each thread puts 10k messages on queue to capture performance statistics.

Scenario	Mode	Message Size	Persistence	Share Conversations	Throughput
1 Queue, 1 Queue Manager, 10 threads	Managed	256 Bytes	Non-Persistent	1	11507 messages/second

3.2 XMS .Net – Linux

Sync Point 1 Queue – 10 Threads PUT scenario

A XMS .Net Put application which is connecting to 1 Queue - 1 Queue Manager using single thread and 10 threads. This scenario covers messages put under sync point. A commit being issued after every 100 messages. Each thread puts 5k messages as a warmup. And then each thread puts 10k messages on queue to capture performance statistics.

Scenario	Mode	Message Size	Persistence	ShareCnv	Throughput
Sync Point - 1 Queue, 1 Queue Manager, 10 threads	Managed	256 Bytes	Non-Persistent	1	5882 messages/second
Sync Point - 1 Queue, 1 Queue Manager, 10 threads	Managed	256 Bytes	Persistent	1	5714 messages/second

4 Tuning Recommendations

4.1 IBM MQ Setup

For this performance report, queue managers were created using the following crtmqm command:

```
crtmqm -lp 16 -lf 65535 <QueueManagerName>
```

Once the queue manager was created, tuning parameters were added to the queue managers' qm.ini as follows:

```
TuningParameters:
DefaultPQBufferSize=1045876
DefaultQBufferSize=1048576
```

Note that the qm.ini was updated before the queue manager was started.

By increasing the amount of memory available to queues for persistent and non-persistent messages, you can help to avoid writing messages out to disk unnecessarily. Please consult your documentation to understand what this means for your IBM MQ installation.

5 Test Environment

5.1 IBM MQ

- IBM MQ Version 9.1.x.0 was used for the queue manager.

5.2 Operating System

5.2.1 Client

- Microsoft Windows Server 2016 Server Standard
- Linux RHEL Workstation 7.5

5.2.2 Server

- Linux Server 1 3.10.0-327.el7.x86_64

5.3 Hardware

5.3.1 Windows Client

Machine Type:	Physical Machine Windows Server 2016 standard
Architecture:	Intel Xeon @ 2201 MHz
Processor:	2 CPU's with 8 Core, 16 Logical Processors
Memory (RAM):	256 GB

5.3.2 Linux Client

Machine Type:	RHEL Workstation 7.5
Architecture:	Intel Core i7-3770, 64 bit
Processor:	CPU @ 3.40 GHz * 8

5.3.3 Linux Server

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	4
Core(s) per socket:	2
Socket(s):	2
CPU family:	15
Model name:	Dual-Core AMD Opteron(tm) Processor 8220
CPU MHz:	2799.972

5.4 Dotnet

- .NET Core SDK 2.1.302 was used for developing and building the .NET Core applications for performance testing.