Low-Latency Trading over RoCE V2 on Oracle Cloud Infrastructure

Andrew Addison, Daniel Bardsley, Ryan Prins, Larry Ryan

BJSS Inc.

New York, USA

lowlatencytrading@bjss.com,

Abstract: To attract order flow and effectively match orders, matching engines have been designed to handle extremely high message rates at low-latencies. Historically, these environments were built with proprietary technology deployed on dedicated on-premise infrastructure to minimize latency and control jitter. However, as outlined in this paper, we believe that innovations in cloud infrastructure allow configuration and implementation of an equally low-latency trading platform in a cloud environment. We have demonstrated this by implementing a simple foreign exchange (FX) trading system and deployed it on RoCE V2 (RDMA over Converged Ethernet) Fabric on Oracle Cloud Infrastructure..

Keywords: trading, low-latency, high performance computing, cloud, Oracle Cloud Infrastructure (OCI)

I. Introduction

In a prior paper[1], we developed a simple trading system, deployed it to three public cloud environments and tuned it for low latency, to establish the feasibility and performance of a low-latency trading system hosted on cloud-based servers. That paper established that trading in the cloud is feasible where sub-millisecond latencies are acceptable.

In this paper, we extend that work by testing the same simple trading system on a RoCE V2 network[2] hosted in Oracle's public cloud, called Oracle Cloud Infrastructure (OCI).

This research aims to prove that you can achieve much lower latencies on a high-speed low-latency fabric.

This section summarizes the paper. Section II describes the cloud test environment. In section III, the paper describes the architecture of the system under test. Section IV describes the metrics captured and Section V presents results measured. Section VI summarizes our findings.

II. Test Environment

Oracle Cloud Infrastructure is an enterprise Infrastructure as a Service (IaaS) platform that hosts enterprise and cloud native applications in a core-to-edge secure environment.

OCI delivers a number of services, including compute, storage, networking, database, and containers.

OCI provides infrastructure services targeting high performant compute (HPC) workload [3] that include:

- Bare Metal and virtual compute instances with the latest NVIDIA GPUs and Intel Skylake processors, including high core count and high frequency options
- Up to 51.2 TB local NVMe storage per compute instance
- 2 x 25 Gbps network interfaces - or 1 x 25 Gbps and 1 x 100 Gbps for RDMA
- Block storage volumes up to 1 PB

RDMA [2] over Converged Ethernet (RoCE) is a network protocol that supports the remote direct memory access (RDMA) protocol over an ethernet network. It directly copies data between memory spaces on two different hosts, bypassing the Operating System and CPU, and in effect, reduces CPU usage and latency when compared to a standard TCP protocol over an ethernet network. For comparison, we have measured sub-µs latencies when pinging a host over RoCE while a TCP Ping requires >50 µs.

III. Trading Scenario

This section describes the test environment and the components of the FX matching engine.

Our test system implements an order book and FX matching engine in an "immediate-or-cancel" trading scenario (see Figure 1). This includes an Order Generator sending orders to an Order Book, which saves the order and forwards to a Matching Engine, which returns results to the Order Book.

The **Order Generator** constantly creates new open Order objects for its configured currency pair, at a prescribed rated. These messages are sent to an Order Book via a socket over RoCE.

The **Order Book** receives new Order Messages from the Order Generator, decodes the message and stores it in memory. The Order Book then enriches the Order Message with receive and send timestamps and forwards it to the Matching Engine via a Socket. Upon completion of order processing, the Matching Engine returns a cancel Order Message response through a different Socket over RoCE. Upon receiving this response, the Order Book removes the Order with the matching order ID from the Order Book.

A standard **Matching Engine** traditionally matches buy and sell orders within a given market**.** The implemented test engine simply receives an order and immediately responds with a cancel message. This simulates the most common action on an FX exchange. The Matching Engine receives a new order message over a TCP Socket, creates a copy of the order message but sets its status to cancelled, sets a new timestamp, then sends this cancel order message to the Order Book through a Socket over RoCE.

The test trading application is written in Java 11 and executed on Oracle's Version 12 JVM. The trading application statically allocated all variables to avoid garbage collection events; this is the standard approach for managing variables in a trading system. In addition, JVM parameters were tuned to mitigate garbage collection activity on latency.

The test application uses the LMAX implementation of the Disruptor pattern [6] to provide high-performance message queueing and inter-thread messaging in a thread-safe environment. Messages are sent between processes using the Java NIO Socket class, which was bound to the RoCE protocol via Mellanox's VMA (Mellanox Message Accelerator)[5].

The trading system was deployed across three Bare Metal instances in OCI's HPC Cluster environment. The three instances had identical configurations:

- BM.HPC2.36 Instance (3.0 GHz Gold 6154 Intel Xeon processor, 36 cores, 384 GB Memory
- Red Hat Server 7.6.
- Mellanox ConnectX-5 Ex Ethernet Controller
- Mellanox OFED Linux 4.6-1.0.1.1 Driver
- Mellanox Message Accelerator (VMA)
- Java Version 12 (Oracle Version)

IV. Performance Metrics

This section describes the process for capturing metrics during our tests.

The test system collects latency measurements in memory for the duration of the performance test and flushes them to disk about once/minute during the test via an independent thread. Although the writing to disk was performed in a separate thread, the thread shared the same memory space as the core trading software and therefore did impact performance of the system. We performed a separate test to measure performance of the trading system while NOT flushing to disk during the test and found minimal impact on response times. However, we did see a noticeable increase in garbage collection events. We did not capture this difference as this would require enabling GC logic, which would further impact test results. Instead, for the purpose of this test, we do not believe that logging to disk was a significant factor in test results. Removing this would decrease GC events, further reducing variance measurements.

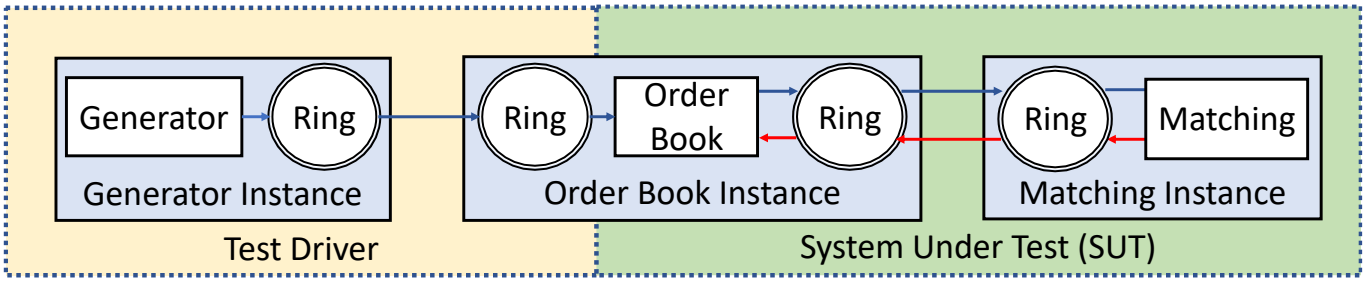

Figure 1: Test Environment.

Latency was recorded by invoking Java's *system.nanoTime()* timestamp method. For each order, latency is recorded at 5 points along the flow of data through the test system. Timing calculation starts when an order is read from the input ring buffer after reception from the Order Generator. The time delta is then calculated at each of the following five points:

1. The Order Book sends an order to the Matching Engine.
2. The Matching Engine receives the Order from the Order Book.
3. The Matching Engine sends the Cancelled Order to the Order Book.
4. The Order Book receives the Cancelled Order.
5. The Order Book marks the Order as completed.

Total latency is the sum of the latencies measured at these five points along the flow of data.

VI. Test Results

This section presents the measured results of running the FX Trading System as described in Section III.

We performed the following tests:

- Raw performance of RoCE network.
- Socket over VMA/RoCE.
- Single product generating orders at 10,000 orders/second for 1-hour.
- Single product generating orders at 10,000 orders/second for 1-day (day test).
- Single product generating orders at 50,000 orders/second for 20-minutes.
- Two products each generating orders at a rate of 50,000 orders/second for a total rate of 100,000 orders/second for 1-hour.
- Five products each generating orders at a rate of 50,000 orders/second for a total rate of 250,000 orders/second for 1-hour.
- Ten products each generating orders at a rate of 50,000 orders/second for a total rate of 500,000 orders/second for 1-hour.

We tested the latency of the raw RoCE network by running the ib_write_lat command tool across two nodes in the network[4]. We ran this test for 30 minutes at peak rate, which averaged 343,912 100-byte messages/second with an average latency of 1.45µs. This test only returns average latency and does not calculate the 99th percentile.

We also ran a sockperf test [5], which measures the latency of a socket connection over VMA/RoCE, for 1-hour by sending 100-byte messages at a message rate of 10,000 messages/second. We measured an average round-trip latency of 2.01µs and 99 percentile latency of 2.16µs. Therefore, VMA appears to increase latency for a round-trip message on average by 0.56 µs (i.e. 560 ns).

We replicated the 10,000 order/second test for 1-hour that was reported in the *Low-Latency Trading in a Cloud Environment* paper[1]. The following table compares these results:

	TCP/Ethernet[1]	VMA/RoCE
Date	2018-07-17	2019-06-11
Instance Type	Virtual[1]	Bare-Metal
Network	TCP/Ethernet	RoCE
Queue	Solace VMR	VMA
Msg Rate	10,000 msg/s	10,000 msg/s
Avg Latency	251µs	9.02µs
Std Deviation	185µs	0.70µs (700ns)
Min Latency	173µs	6.66µs
50th %	242µs	8.99µs
95th %	314µs	10.74µs
99th %	357µs	12.48µs
Max Latency	45ms	402µs

Table 1. Comparison of best performance achieved as reported in [1] vs VMA/RoCE

The 99th percentile total latency of RoCE is 3.5% (1/28th) of the 99th percentile total latency for the best result as reported in [1]. There are two reasons for this reduction in latency: replacement of Solace VMR Message Queue with VMA, which (1) eliminates a server, two message hops and queueing from the system under test and (2) replaces the TCP/Ethernet stack with RoCE. Note that VMA provides a socket interface over RDMA and is not a queueing system. Instead, we relied on the LMAX Disruptor ring[6] to provide queue support.

Single-Day Test Results

We ran a latency performance test on June 17th over a 24-hour period to determine the impact of time of day on total latency. The minimum latency measured over 24 hours was 6.36µs, the mean 8.27µs, the 99th% 9.94µs and the Standard Deviation 1.02µs. There are very few high latencies (481 out of 8.6 billion orders latency exceeded 100µs) during this test period. We believe these high latencies are due to Java garbage collection events. Further investigation would be required to validate this assumption. The test shows that the latency is stable across one-day. See figure 2.

Higher Volume Test Results

Because the RoCE fabric can support higher throughput rates at lower latencies, we increased the message rate beyond 10,000 orders/second and to 50,000 orders/second with no latency increase. In fact, latency decreased at this higher rate (see Table 1).

At 75,000 orders/seconds the average latency increased to 153.99µs due to queuing. At this rate, the arrival volume is 75% of the service rate and therefore queueing is expected.

[1] The Low-Latency Trading in a Cloud Environment [1] included testing of bare-metal servers, but the best results were measured on virtual instances.

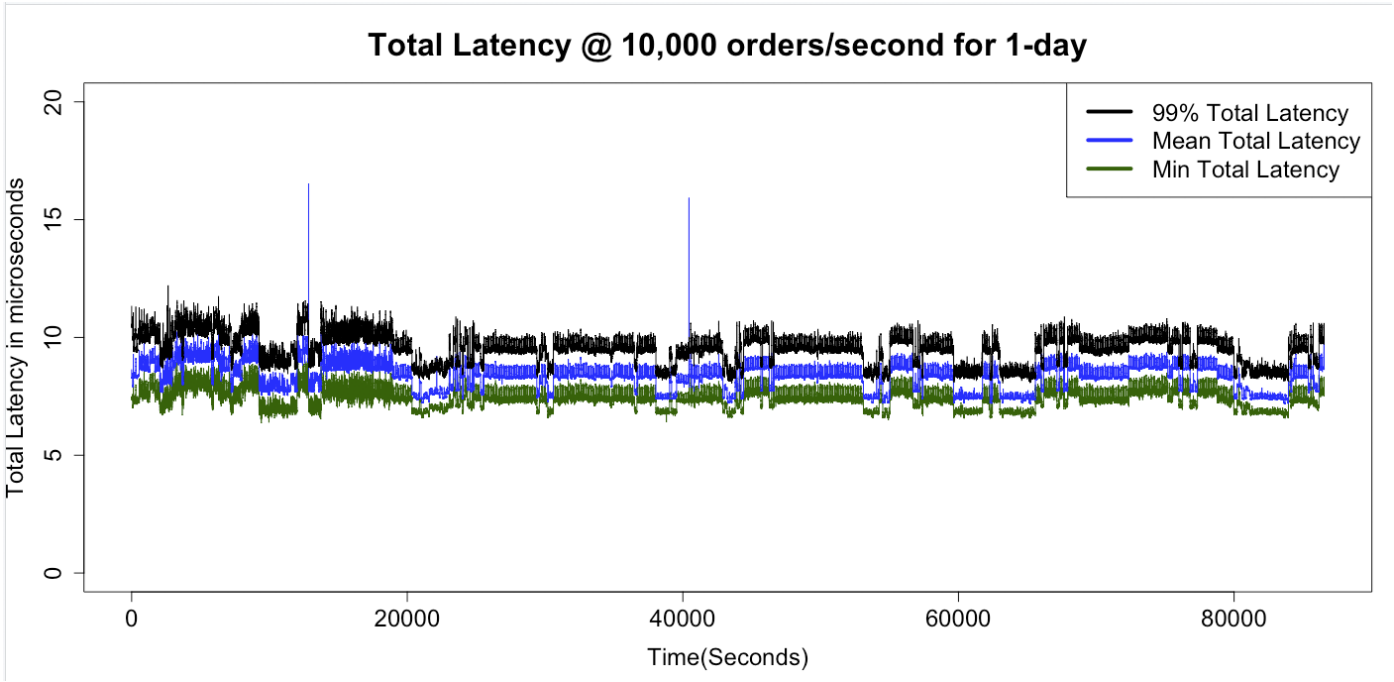

Figure 2: 1-Day Test

Because the maximum processing rate we could achieve for a single product, without increasing latency, was 50,000 orders/second, to increase volume we added products (Currency Pairs), which can be proceeded in parallel. We tested the latency of the system with 1, 2, 5 and 10 concurrent products reaching ½ million orders/second. Table 2 summarizes the results of these tests:

Table 2 shows that the minimum and quantiles up to the 99th percentile are stable and don't increase, as message rate increases from 10,000 orders/second to ½ million orders/second. Although jitter is low for volumes up to 100,000 orders/second, as shown by low standard deviation, at 100,000 orders/second the order rate doubled but maximum latency increased by five-fold. The 100,000 order/second test is configured as two concurrent 50,000 orders/second tests on different products. The increased maximum latency indicates resource contention across the two concurrent threads. The two threads may be contending for CPU, memory, network fabric and/or disk. Identifying the contention source requires further analysis. Because the threads execute in different process space, we don't attribute this to an increase in garbage collection activity. When the number of concurrent threads is increased from two to five, the maximum latency is increased by a factor of about ten while the standard deviation is increased by a factor of about thirty; indicating an increase in high spikes. Analysis of the data showed that the spikes appeared uniformly distributed over the test interval.

Date	2019-06-11	2019-06-18	2019-06-14	2019-06-14	2019-06-25
Instance Type	Bare-Metal	Bare-Metal	Bare-Metal	Bare-Metal	Bare-Metal
Network	RoCE	RoCE	RoCE	RoCE	RoCE
# Products	1	1	2	5	10
Msg Rate/Product	10,000 msg/s	50,000 msg/s	50,000 msg/s	50,000 msg/s	50,000 msg/s
Total Msg Rate	10,000 msg/s	50,000 msg/s	100,000 msg/s	250,000 msg/s	500,000 msg/s
Avg Latency	9.02µs	7.55µs	7.92µs	9.89µs	10.05µs
Std Deviation	0.70µs (700ns)	0.68µs	3.34µs	102.13µs	114.95µs
Min Latency	6.66µs	6.11µs	5.85µs	6.02µs	5.90µs
50th %	8.99µs	7.46µs	7.86µs	8.41µs	8.80µs
95th %	10.74µs	8.24µs	9.47µs	10.06µs	10.30µs
99th %	12.48µs	9.35µs	10.22µs	11.20µs	11.20µs
Max Latency	402µs	705.43µs	3,.516ms	37.33ms	37,33ms

Table 2. Comparison of higher volume test results

V. Conclusion

Low-latency trading in financial markets focuses on processing trades and retrieving market data at the fastest speed possible[1]. In this paper, we deployed a simple trading system on a RoCE V2 network fabric in the OCI cloud platform and tested latency under various configurations and order rates. While achieving very low minimum latency, we identified undesirable jitter when running concurrent threads at message rates exceeding 100,000 orders/second. This jitter was uniformly distributed and less than 0.1% of the traffic, but proportionally high when compared to the mean and 99th percentile latency. Determining the source of this jitter requires further investigation, but based on our analysis, we don't believe that this jitter is due to any cloud technologies. Rather, eliminating this jitter requires typical tuning and adjustment for any low-latency application.

The main contribution of this work is extending the work in [1] and demonstrating that public cloud platforms can support workloads that require deterministic latency in the 10µs range at order rates of 50,000 orders/second. Moreover, we believe that higher rates are possible with further tuning. This is significant because services requiring this service level avoid expensive on-site deployments.

VI. Acknowledgements

We would like to thank Oracle Corporation for their support and collaboration, and the use of their resources in concert with this effort. We understand that this research is of great importance to industry leaders in cloud computing and we appreciate their partnership.

VII. References

1. Addison, A., Andrews, C., Asad, N., Bardsley, D., Bauman, J., Diaz, J., Didik, T., Fazliddin, K., Gromova, M., Krish, A., Prins, R., Ryan L., Villette, N., *Low-Latency Trading in a Cloud Environmen*t, 22nd IEEE International Conference on Computational Science and Engineering (IEEE CSE 2019), 2019, available at https://www.bjss.com/wp-content/uploads/Low-Latency-Trading-in-a-Cloud-Environment.pdf .

2. *Remote Direct Memory Access (RDMA) Consortium*. 2018; Available from: http://www.rdmaconsortium.org.

3. *HPC on Oracle Cloud Infrastructure*; 2019, Available at https://www.oracle.com/cloud/solutions/hpc.html

4. *Perftest Package. Mellanox Knowledge Article*, February, 2019. Available at https://mymellanox.force.com/mellanoxcommunity/s/article/perftest-package

5. Mellanox Messaging Accerator documentation, available at https://www.mellanox.com/page/software_vma?mtag=vma .

6. Thompson, M., et al., *Disruptor: High performance alternative to bounded queues for exchanging data between concurrent threads*. Technical paper. LMAX, May, 2011: p. 206.