FIX Antenna Java 2.15.27 Benchmarks

Environment

Sender Host:

- Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz (2 CPU Hyper-Trading Enabled, 24 Cores)
- RAM 128 GB, 2133 MHz
- NIC Solarflare Communications SFC9120 (Firmware-version: 4.2.2.1003 rx1 tx1)
- Linux (CentOS 7.0.1406 kernel 3.10.0-123.el7.x86_64)
- SolarFlare driver version: 4.1.0.6734a

Receiver Host:

- Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz (2 CPU Hyper-Trading Enabled, 20 Cores)
- RAM 128 GB, 2133 MHz
- NIC Solarflare Communications SFC9120 (Firmware-version: 4.2.2.1003 rx1 tx1)
- Linux (CentOS 7.0.1406 kernel 3.10.0-123.el7.x86_64)
- SolarFlare driver version: 4.1.0.6734a

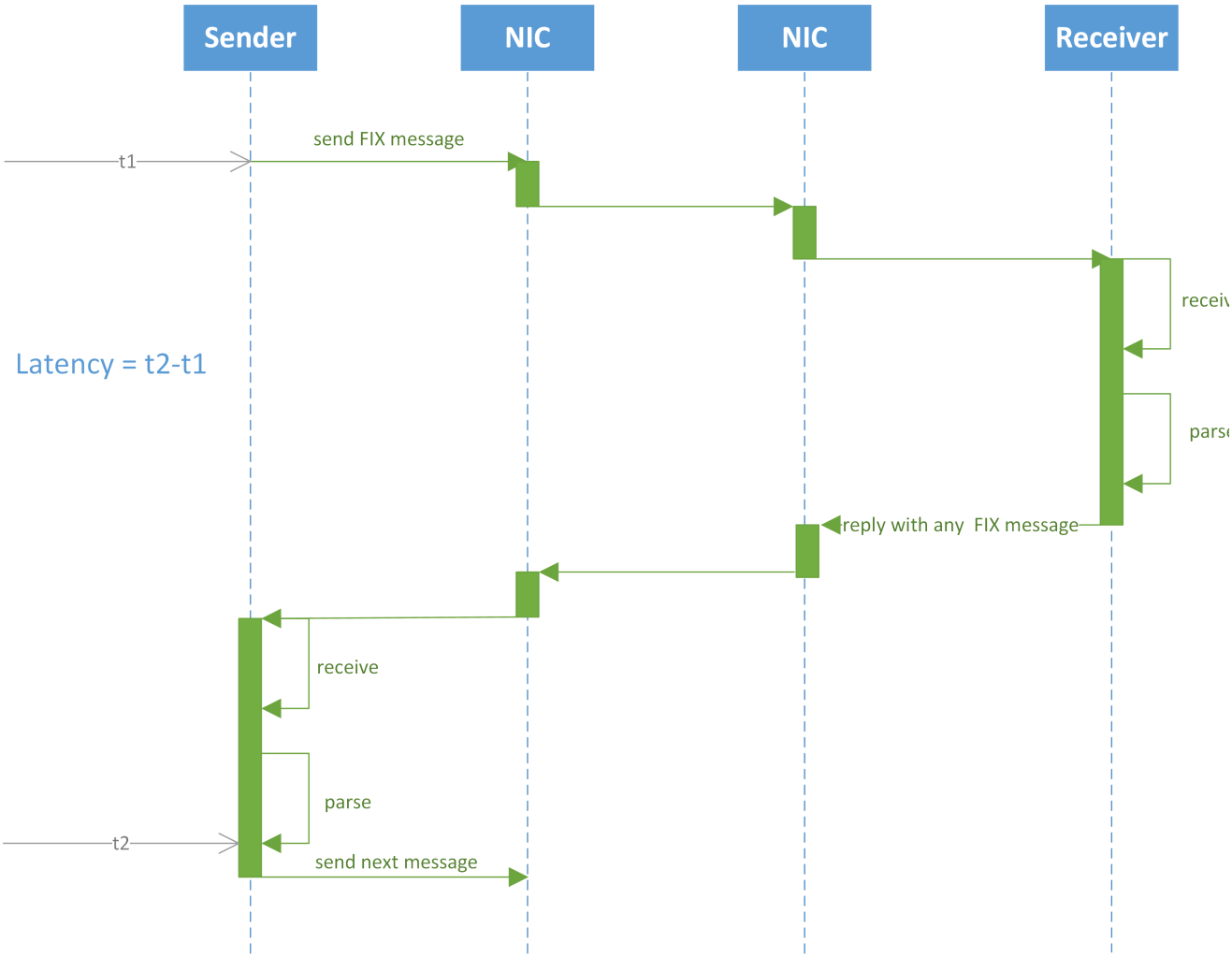
Test scenario

The test scenario is the following:

- Two test servers are connected via 10GB link.
- An Initiator FIX session is established on the Sender Host , an Acceptor FIX session is established on the Receiver Host.
- The Initiator sends the New Order Single (35=D) message to the Acceptor; the Acceptor receives, validates and parses the message and sends the Execution report (35=8) message back to the Initiator.

During the test Round-trip time (RTT) latency is measured. The first measurement, t1, is made before the message is sent by the Initiator, the second, t2, is made after the received message is parsed by the Initiator.

RTT=t2-t1.



Test configurations

Properties		Balanced	Optimized
Nagle's algorithm[1]			
Message validation parameters	validateCheckSum		
	validateGarbledMessage		
Storage type		Persistent	In memory
Queue type		Persistent	In memory

Nagle's algorithm[1] - the algorithm aimed at reducing the number of packets that need to be sent over the network. Nagle's algorithm works by combining a number of small outgoing messages and sending them all at once.

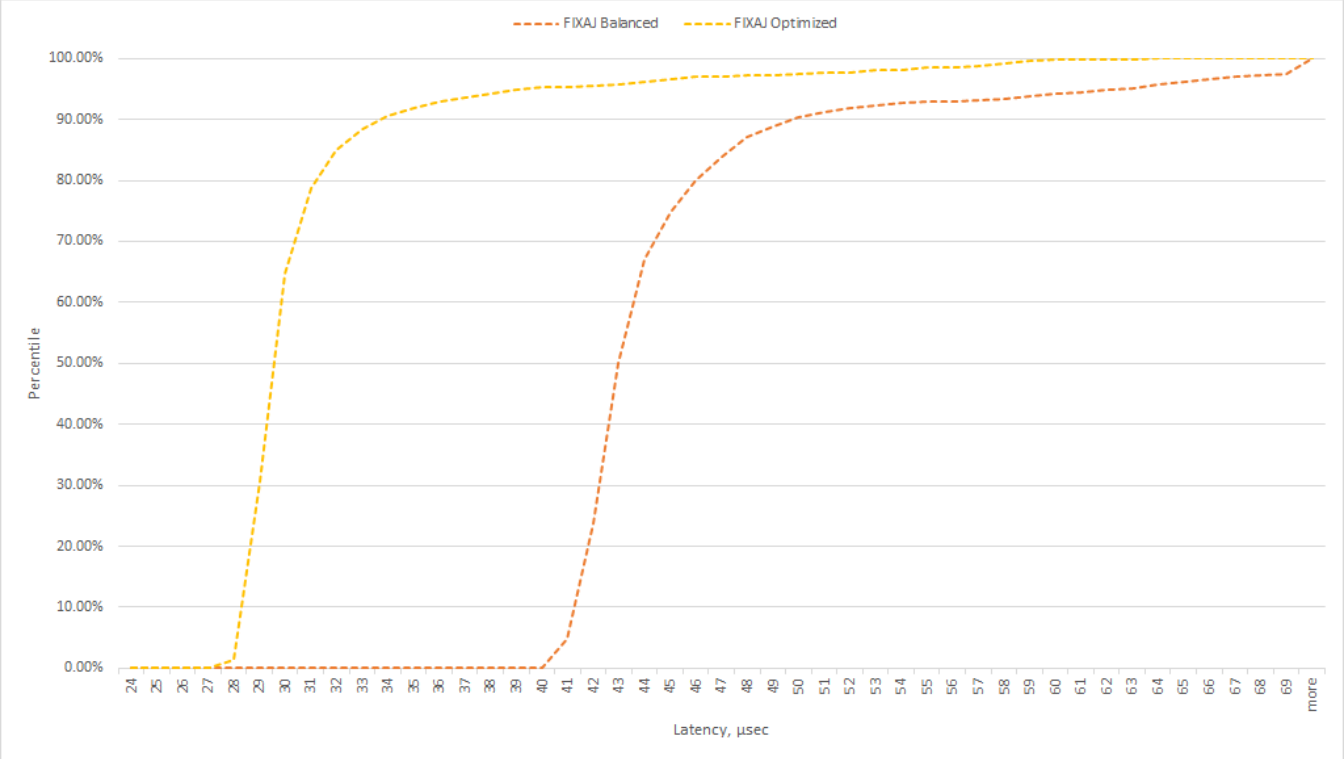
The following description will help to choose the most relevant FIX Antenna Java configuration:

- **Balanced**. It is the good starting point with balanced performance and security. Basic validation options are enabled to prevent corrupted message processing.
- **Optimized**. It is the configuration aimed at maximum performance. All validation options as well as message persistence are disabled, so it has to be used only in the fully controlled environment.

Results

FIX Antenna Java configurations comparison

Configuration	Balanced	Optimized
Latency values (microseconds)		
Min	39,9	27,2
Max	1442,3	1200,7
Average	45,8	31,1
Latency distribution (percentiles)		
50%	43	29,5
95%	62,6	39,3
99%	74,8	57,6



FIX Antenna Java 2.15.27 vs QuickFIX Java 1.6.3

The benchmark code was ported to QuickFIXJ:

Sender Host

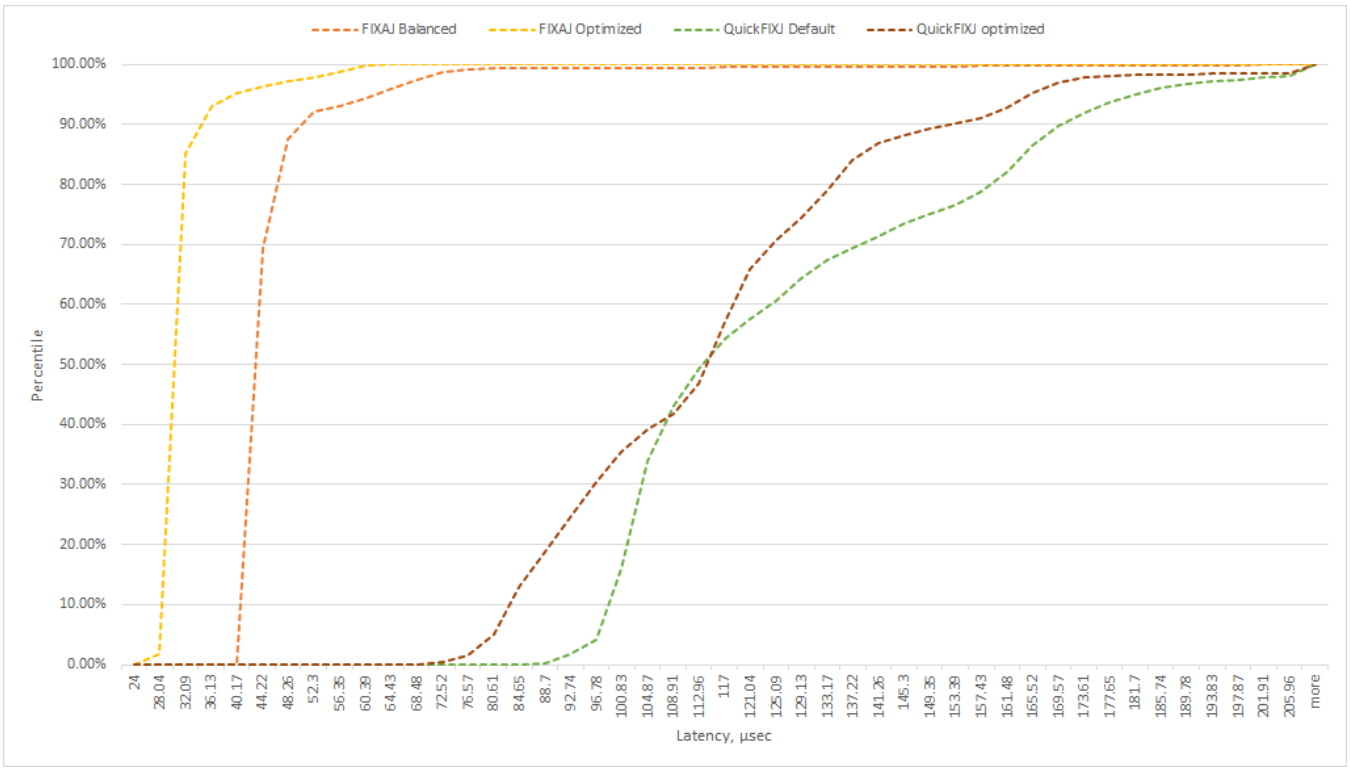
```
public final class RoundTripTester extends ApplicationAdapter {
    ..
    private void sendMessage() throws IOException {
        workMessage.getHeader().setInt(MsgSeqNum.FIELD, workMessageSequence++);
        start = System.nanoTime();
        session.send(workMessage);
    }


    @Override
    public void fromApp(Message message, SessionID sessionId) throws FieldNotFound, IncorrectDataFormat,
IncorrectTagValue, UnsupportedMessageType {
            long end = System.nanoTime();
            String type = message.getHeader().getString(MsgType.FIELD);
            if (workMessageType.equals(type)) {
                final long currentLatency = end - start;
                latencies[counter++] = currentLatency;
                if (limitEnabled) {
                    long nextSendTime = testStart + (counter * sendIntervalMsec);
                    while (true) {
                        long currentTime = System.currentTimeMillis();
                        long toSleep = nextSendTime - currentTime;
                        if (toSleep <= 0) {
                            break;
                        }
                    }
                }

                if (counter == NO_OF_MESSAGES) {
                    //print results
                } else {
                    sendMessage();
                }
            } else if (warmUpMessageType.equals(type)) {
                warmUpCounter++;
                if(warmUpCounter == WARM_UP_CYCLE_COUNT){
                    log.info("Warm up cycle has been done");
                    testStart = System.currentTimeMillis();
                    sendMessage();
                }
            }
    }
    ...
```

Receiver Host

```
  public final class RoundTripServer extends ApplicationAdapter {
    ....
    @Override
    public void fromApp(Message message, SessionID sessionId) {
         try {
            String type = message.getHeader().getString(MsgType.FIELD);
            if("D".equals(type)) {
                executionReport.setString(ClientID.FIELD, message.getString(ClientID.FIELD));
                executionReport.setString(Price.FIELD, message.getString(Price.FIELD));
                executionReport.setString(OrderQty.FIELD, message.getString(OrderQty.FIELD));
                executionReport.setString(TransactTime.FIELD, message.getString(TransactTime.FIELD));
                executionReport.setInt(ExecID.FIELD, ++execID);
                executionReport.setInt(OrderID.FIELD, ++orderID);
                Session.sendToTarget(executionReport, sessionId);
            } else {
                Session.sendToTarget(message, sessionId);
            }
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        }
    }
...
```

Results can be compared in the following table:

Configuration	FIXAntenna Java balanced	FIXAntenna Java optimized	QuickFIXJ default	QuickFIXJ optimized
Latency values (microseconds)				
Min	39,9	27,2	85,7	70,0
Max	1442,3	1200,7	3891,8	43565,9
Average	45,8	31,1	258,0	258,0
Latency distribution (Percentiles)				
50%	43,0	29,5	113,5	114,3
95%	62,6	39,3	182,1	165,0
99%	74,8	57,6	2107,0	1336,4


FIXAJ Balanced
FIXAJ Optimized
QuickFIXJ Default
QuickFIXJ optimized
Percentile
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
24
28.04
32.09
36.13
40.17
44.22
48.26
52.3
56.35
60.39
64.43
68.48
72.52
76.57
80.61
84.65
88.7
92.74
96.78
100.83
104.87
108.91
112.96
117
121.04
125.09
129.13
133.17
137.22
141.26
145.3
149.35
153.39
157.43
161.48
165.52
169.57
173.61
177.65
181.7
185.74
189.78
193.83
197.87
201.91
205.96
more
Latency, µsec