Bon Trade, Mellanox and HP Benchmark Report
RMC™ messaging over Shared-Memory and InfiniBand Transports

White Paper
June 2012
By Tom McSherry
## Contents

- Introduction .................................................................................................................. 1
- Executive Summary ......................................................................................................... 1
- Benchmark Overview ...................................................................................................... 2
- Description of Measurements ......................................................................................... 2
- Hardware ....................................................................................................................... 4
- Bon Trade Network Component .................................................................................... 6
- Shared Memory Results ................................................................................................. 7
- RDMA Results ................................................................................................................ 10
- Multi-Core Scaling Results ........................................................................................... 15
- Conclusion ..................................................................................................................... 21
- Resources ..................................................................................................................... 22
- Credits ......................................................................................................................... 23
Introduction

This paper details Bon Trade’s unprecedented industry milestone of 598.4 million messages per second across an InfiniBand network, running on Windows Server. The latency tolerances of the testing were single digit microsecond. Bon Trade developed a proprietary high bandwidth/low latency messaging system trademarked RMC to meet the current and future demands of global electronic markets. These benchmarks were produced with RMC messaging used in Bon Trade’s InfiniBand Enterprise System.

Taking full advantage of multi-core processor design innovation, RMC delivers unparalleled performance across InfiniBand. The results underscore the viability of deploying an ultra high performance enterprise risk, trading and messaging solution in a Windows Server environment. The Bon Trade team would like to thank HP, Mellanox and Microsoft for their support, advice and the opportunity to run our tests on their newly released generation of commercially available technology.

Executive Summary

- Message throughput scaled almost perfectly across all 16 cores, resulting in a sustain message rate of over 598 million per second. These results demonstrate the enormous power of having a dedicated adapter for each server CPU.

- Single digit microsecond latency with message rates of 520 million messages per second across 14 cores.

- The latency and throughput results achieved on the Windows operating system rival any messaging layer on any operating system.

- Many industry benchmarks are setup with a single producer and hundreds of consumers to increase the message count. Such tests emulate a broad cast of data rather than a mission critical trade message.

- All benchmarks were obtained using commercially available equipment including an InfiniBand switch in between the test Host servers.

- This technology is incorporated and deployed in Bon Trade production systems.

- Bon Trade technology can process huge amounts of data and handle order bursts with no performance loss.
Benchmark Overview

All test connections used a 1:1 model of communication. This is the predominant form of communication used by an order-routing network. For example the path taken by a client order request to an exchange consists entirely of 1-1 connections. The path taken by the order acknowledgement back to the client is similarly composed. See the connectivity diagram below for an illustration.

We benchmarked the RMC messaging protocol over the following transports focusing on the characteristic data paths in each case.

- RMC / Shared Memory
- RMC / RDMA / InfiniBand

The latency, maximum message rate (RateMAX) and total bandwidth were measured for each 1-1 connection and selected message size. The units of measurement used throughout are as follows:

- All Latency measurements are in nanoseconds (ns)
- Message rates are in millions of messages per second (M/s).
  Frame sizes are shown in parenthesis next to the rate inside the table.
- Bandwidth is given in Giga-bits per second (Gbps) where:
  1 Gbps = $10^9$ bits/second as defined by the International System of Units.

Description of Measurements

The **Latency** is defined as the average time per hop. Latency $= T/10000000$, where $T$ is the total time in nanoseconds it took to pass the message back and forth between client and server ten million times (i.e. five million round-trips).

The **Bandwidth** is a direct function of the message Size and RateMAX values.

$$\text{Bandwidth} = \text{Size} \times \text{RateMAX} / 125 \text{ Gbps}$$

where the Size is in bytes and the RateMAX is in millions of messages per second.

For example: (12 bytes/msg * 41.6 million msgs/sec) / 125 = 3.99 Gbps
Message rates were tested across several different application-level frame sizes which are shown in parenthesis next to the rate inside the table.

An application uses message framing when it wants to increase message throughput. A common usage is during a re-sequencing operation where minimizing latency is less important than maximizing throughput. Here the server will enable framing on the respective client connection and proceed to send messages, calling the "Flush" function every "frame size" messages (i.e. once per "frame").

Message framing introduces additional latency into the stream. Under a sustained message flow of rate 'R', and using a frame size 'F' in messages per frame, the additional latency introduced by "holding back" the subsequent (F-1) messages is: \( \text{FrameLatency} = \frac{F-1}{R} \)

For example suppose we obtain a sustained message flow at a rate of: R= 20 Million/second, using a frame size of F= 101 messages. Then \( \frac{1}{R} = 50 \text{ ns} \) is the average time spent processing a message and \( \frac{1}{R} \times (F-1) = 50 \text{ ns} \times 100 = 5 \text{ microseconds} \) is the average time a message spends sitting in either the sender-side or receiver-side application message queues. The exact position of the message in the frame determines what portion of this time is spent in the sender-side queue and what portion is spent sitting in the receiver-side queue.

The amount of additional FrameLatency an application is willing to tolerate depends on the latency of the underlying transport itself. For example, an application communicating with a satellite experiencing two second latency is likely to tolerate another 5 microseconds in the transmission if the increase in throughput is significant enough.

We measured the message Rate across multiple frame sizes in the range: 1••• 512.

The RateMAX measurement used a frame size large enough to come within a few percent of the maximum achievable throughput. Here we maximized throughput at the expense of introducing additional latency. The frame sizes here never exceeded 256 messages.

The RateMIN measurement used a frame size of F=1. Here no additional FrameLatency is tolerated. This setting minimizes latency at the expense of message throughput.
Hardware

- Two 16-core HP ProLiant DL380p Gen8 Host servers. Each server had Dual (8-core) Intel Xeon E5-2690 CPUs @ 2.9GHz
- Microsoft Windows Server 2008 R2
- Two Mellanox ConnectX-3 FDR InfiniBand Adapters @ 54Gbps were installed on each server.
- Mellanox SwitchX® based FDR 56Gb/s InfiniBand switch

HP ProLiant DL380p BIOS settings

<table>
<thead>
<tr>
<th>Menu</th>
<th>Submenu</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Options</td>
<td>Processor Options</td>
<td>Intel(R) Virtualization Technology</td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel(R) Hyperthreading Options</td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel(R) Turbo Boost Technology</td>
<td>Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel(R) VT-d</td>
<td>Disabled</td>
</tr>
<tr>
<td>Power Management</td>
<td>HP Power Profile</td>
<td></td>
<td>Maximum Performance</td>
</tr>
<tr>
<td></td>
<td>HP Power Regulator</td>
<td></td>
<td>HP Static High Performance Mode</td>
</tr>
<tr>
<td></td>
<td>Advanced Power Management Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel QPI Link Power Management</td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum Processor Idle Power Core State</td>
<td>No C-states</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum Processor Idle Power Package</td>
<td>No Package State</td>
</tr>
<tr>
<td>State</td>
<td>Energy/Performance Bias</td>
<td></td>
<td>Maximum Performance</td>
</tr>
<tr>
<td></td>
<td>Collaborative Power Control</td>
<td></td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td>DIMM Voltage Preference</td>
<td></td>
<td>Optimized for Performance</td>
</tr>
<tr>
<td>Advanced Options</td>
<td>Thermal Configuration</td>
<td></td>
<td>Maximum Cooling</td>
</tr>
<tr>
<td>Service Options</td>
<td>Processor Power and Utilization Monitoring</td>
<td></td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td>Memory Pre-Failure Notification</td>
<td></td>
<td>Disabled</td>
</tr>
</tbody>
</table>
The Adapter to CPU / NUMA-node correspondence on each Host was identical:

Host1 : Adapter1 <-- Cpu1[Cores 0-7]
Host1 : Adapter2 <-- Cpu2[Cores 8-15]
Host2 : Adapter1 <-- Cpu1[Cores 0-7]
Host2 : Adapter2 <-- Cpu2[Cores 8-15]

Single adapter measurements used the link:

Host1 : Adapter1 <-- Host2 : Adapter1

Dual adapter measurements used the two links (see scalability tests):

Host1 : Adapter1 <-- Host2 : Adapter1
Host1 : Adapter2 <-- Host2 : Adapter2
All connections were established using two instances of our Bon Trade Network Component. By convention the side initiating the connection had the responsibility of recording the test results. We will refer to this side as the ‘Client’.

Each instance of the Bon Trade Network Component may be configured to use a specific subset of available CPU cores or to run without affinity. Client and server affinity settings can have a dramatic effect on performance, most notably as it relates to the message Latency. In the tests below we affinitized each instance of the Bon Trade Network Component to a single/dedicated CPU core and speak of the "data path" connecting them. Each path-endpoint corresponds to a specific CPU core located on a specific Host. Hence every path-endpoint has associated with it, a distinct NUMA node, each with its own local memory and local FDR adapter attached.

The Bon Trade Network Component employs the standard "ping-pong" approach to latency measurement. Here the client-side starts by sending a ping message to the server and the server responds by echoing it back. If the total number of roundtrips is fewer than five million the client will issue another ping message, otherwise it will end the test and compute the average (half-roundtrip) latency according to: \[ \text{Latency} = \frac{\text{ElapsedTime}}{10000000} \]

Throughput tests have the client-side as the message producer and the server-side as the message consumer. The producer generates and delivers a continuous stream of messages as fast as the consumer can process them, maintaining a steady and sustained maximal flow. Messages will continue to stream until the user explicitly stops the transfer. The server will periodically issue a "transfer status" message to the client containing/acknowledging the total number of messages it has received since the transfer began. It is only in response to this status message that the client updates its message count. Hence the reported message rates always reflect the actual number of messages consumed and not simply, the number of messages produced. Every message sent is acknowledged in this way.
RMC / Shared Memory

Here we measured the Latency, RateMAX and Bandwidth across two characteristic shared memory data paths;

- Data Path 1 connected two cores on the same CPU /NUMA-node
- Data Path 2 connected two cores on separate CPUs /NUMA-nodes.

In both cases the shared memory region was allocated from memory local to the server-side NUMA-node.

Data Path 1

Data Path 2

<table>
<thead>
<tr>
<th>Data Path 1 Results</th>
<th>Data Path 2 Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td><strong>Latency (ns)</strong></td>
</tr>
<tr>
<td>12</td>
<td>187</td>
</tr>
<tr>
<td>16</td>
<td>208</td>
</tr>
<tr>
<td>32</td>
<td>223</td>
</tr>
<tr>
<td>64</td>
<td>223</td>
</tr>
<tr>
<td>128</td>
<td>236</td>
</tr>
<tr>
<td>256</td>
<td>248</td>
</tr>
<tr>
<td>512</td>
<td>303</td>
</tr>
<tr>
<td>1024</td>
<td>398</td>
</tr>
<tr>
<td>2048</td>
<td>565</td>
</tr>
<tr>
<td>4096</td>
<td>898</td>
</tr>
</tbody>
</table>
The following tables show the effects of message framing over the RMC / Shared Memory transport:

### Shared Memory Data Path 1 Framing Results

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Rate 12</th>
<th>Rate 16</th>
<th>Rate 32</th>
<th>Rate 64</th>
<th>Rate 128</th>
<th>Rate 256</th>
<th>Rate 512</th>
<th>Rate 1024</th>
<th>Rate 2048</th>
<th>Rate 4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.1</td>
<td>7.91</td>
<td>6.95</td>
<td>7.15</td>
<td>6.19</td>
<td>5.68</td>
<td>6.33</td>
<td>4.67</td>
<td>3.18</td>
<td>1.99</td>
</tr>
<tr>
<td>2</td>
<td>20.2</td>
<td>11.2</td>
<td>10.4</td>
<td>10.1</td>
<td>9.01</td>
<td>7.36</td>
<td>6.80</td>
<td>4.81</td>
<td>3.26</td>
<td>2.05</td>
</tr>
<tr>
<td>3</td>
<td>33.2</td>
<td>13.2</td>
<td>11.1</td>
<td>10.4</td>
<td>9.41</td>
<td>7.87</td>
<td>7.05</td>
<td>4.87</td>
<td>3.31</td>
<td>2.05</td>
</tr>
<tr>
<td>4</td>
<td>41.8</td>
<td>13.5</td>
<td>11.9</td>
<td>11.2</td>
<td>10.3</td>
<td>8.70</td>
<td>7.43</td>
<td>4.98</td>
<td>3.29</td>
<td>2.06</td>
</tr>
<tr>
<td>8</td>
<td>41.8</td>
<td>17.2</td>
<td>15.3</td>
<td>14.0</td>
<td>12.3</td>
<td>10.1</td>
<td>7.78</td>
<td>5.32</td>
<td>3.44</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>41.8</td>
<td>19.5</td>
<td>16.5</td>
<td>15.1</td>
<td>13.5</td>
<td>11.0</td>
<td>8.15</td>
<td>5.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>41.9</td>
<td>20.9</td>
<td>17.5</td>
<td>16.0</td>
<td>14.2</td>
<td>11.5</td>
<td>8.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>41.8</td>
<td>22.0</td>
<td>18.0</td>
<td>16.5</td>
<td>14.5</td>
<td>11.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>41.7</td>
<td>22.6</td>
<td>18.3</td>
<td>16.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>41.6</td>
<td>22.8</td>
<td>18.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>41.5</td>
<td>23.0</td>
<td></td>
<td></td>
<td></td>
<td>RateMAX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Shared Memory Data Path 2 Framing Results

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Rate 12</th>
<th>Rate 16</th>
<th>Rate 32</th>
<th>Rate 64</th>
<th>Rate 128</th>
<th>Rate 256</th>
<th>Rate 512</th>
<th>Rate 1024</th>
<th>Rate 2048</th>
<th>Rate 4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.04</td>
<td>4.33</td>
<td>3.48</td>
<td>3.43</td>
<td>3.15</td>
<td>3.06</td>
<td>4.11</td>
<td>2.60</td>
<td>2.16</td>
<td>1.35</td>
</tr>
<tr>
<td>2</td>
<td>9.49</td>
<td>6.45</td>
<td>6.04</td>
<td>5.83</td>
<td>5.11</td>
<td>5.28</td>
<td>4.12</td>
<td>2.63</td>
<td>1.96</td>
<td>1.40</td>
</tr>
<tr>
<td>3</td>
<td>13.5</td>
<td>8.84</td>
<td>7.11</td>
<td>5.78</td>
<td>5.65</td>
<td>5.74</td>
<td>4.36</td>
<td>2.82</td>
<td>2.05</td>
<td>1.42</td>
</tr>
<tr>
<td>4</td>
<td>14.5</td>
<td>8.43</td>
<td>7.50</td>
<td>7.85</td>
<td>5.91</td>
<td>6.18</td>
<td>4.45</td>
<td>2.86</td>
<td>2.13</td>
<td>1.41</td>
</tr>
<tr>
<td>8</td>
<td>39.8</td>
<td>12.0</td>
<td>10.9</td>
<td>9.87</td>
<td>8.55</td>
<td>8.04</td>
<td>5.08</td>
<td>2.97</td>
<td>2.18</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>40.4</td>
<td>15.4</td>
<td>12.9</td>
<td>11.7</td>
<td>10.4</td>
<td>8.90</td>
<td>5.24</td>
<td>3.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>40.4</td>
<td>18.0</td>
<td>14.7</td>
<td>13.3</td>
<td>11.8</td>
<td>9.87</td>
<td>5.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>40.3</td>
<td>19.7</td>
<td>16.0</td>
<td>14.5</td>
<td>12.7</td>
<td>10.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>40.2</td>
<td>21.0</td>
<td>17.0</td>
<td>15.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>40.1</td>
<td>21.8</td>
<td>17.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>40.0</td>
<td>22.4</td>
<td></td>
<td></td>
<td></td>
<td>RateMAX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Latencies between two cores on separate CPUs were roughly 2-3 times higher than those between cores on the SAME CPU. Some of this additional latency can be attributed to the QPI link(s) connecting the two CPUs and some of it to the cache coherency protocols implemented internally in the hardware such as MESI and snooping.

Memory contention is the most severe at small frame sizes and even more so when communicating between cores on separate CPUs. This leads to shared memory message rates that fall far below the respective RDMA message rate.
The jump in message rate (see Path [1], Size=12) between FrameSizes 1 and 4 is the result of memory contention between reader & writer threads for the same 64-byte cache-line. It's exaggerated for 12-byte messages because the shared memory reader in this case is slower than the shared memory writer. Hence the reader thread always trails the writer thread when both execute at the same clock speed and has the effect of reducing memory contention.
Here we focused testing on the four characteristic data paths between Host1 (client) and Host2 (server):

**Data Path 1:** Core0(Cpu1) <--Adapter1 --Switch-- Adapter1--> Core0(Cpu1)

**Data Path 2:** Core8(Cpu2) <--Adapter1 --Switch-- Adapter1--> Core0(Cpu1)

**Data Path 3:** Core0(Cpu1) <--Adapter1 --Switch-- Adapter1--> Core8(Cpu2)

**Data Path 4:** Core8(Cpu2) <--Adapter1 --Switch-- Adapter1--> Core8(Cpu2)

**RDMA Data Path 1:** Core0(Cpu1) <--Adapter1 --Switch-- Adapter1--> Core0(Cpu1)
RDMA Data Path 2: Core8(Cpu2) <-- Adapter1 -- Switch -- Adapter1 --> Core0(Cpu1)

RDMA Data Path 3: Core0(Cpu1) <-- Adapter1 -- Switch -- Adapter1 --> Core8(Cpu2)
RDMA Data Path 4: Core8(Cpu2) <--Adapter1 --Switch-- Adapter1--> Core8(Cpu2)

RDMA Data Path 1 provided the lowest Latency. This is because Adapter1 was closest to Cpu1 on both Hosts.

<table>
<thead>
<tr>
<th>Size</th>
<th>Latency</th>
<th>RateMAX</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Framing)</td>
<td></td>
</tr>
<tr>
<td>Bytes</td>
<td>Ns</td>
<td></td>
<td>Gbps</td>
</tr>
<tr>
<td>12</td>
<td>1034</td>
<td>37.6(256)</td>
<td>3.61</td>
</tr>
<tr>
<td>16</td>
<td>1062</td>
<td>18.6(256)</td>
<td>2.38</td>
</tr>
<tr>
<td>32</td>
<td>1087</td>
<td>15.9(128)</td>
<td>4.07</td>
</tr>
<tr>
<td>64</td>
<td>1136</td>
<td>14.6(64)</td>
<td>7.48</td>
</tr>
<tr>
<td>128</td>
<td>1220</td>
<td>13.7(64)</td>
<td>14.0</td>
</tr>
<tr>
<td>256</td>
<td>1797</td>
<td>11.5(32)</td>
<td>23.6</td>
</tr>
<tr>
<td>512</td>
<td>1952</td>
<td>8.79(16)</td>
<td>36.0</td>
</tr>
<tr>
<td>1024</td>
<td>2267</td>
<td>5.79(8)</td>
<td>47.4</td>
</tr>
<tr>
<td>2048</td>
<td>2894</td>
<td>2.98(4)</td>
<td>48.8</td>
</tr>
<tr>
<td>4096</td>
<td>3591</td>
<td>1.49(2)</td>
<td>48.8</td>
</tr>
</tbody>
</table>

Bandwidth Limit reached with 90% Utilization
The following table shows the effects of message framing over the RMC / RDMA / InfiniBand transport:

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>28.0</td>
<td>16.4</td>
<td>14.1</td>
<td>12.5</td>
<td>10.2</td>
<td>8.42</td>
<td>5.76</td>
<td>3.48</td>
<td>2.14</td>
<td>1.42</td>
</tr>
<tr>
<td>16</td>
<td>30.6</td>
<td>17.1</td>
<td>14.5</td>
<td>12.7</td>
<td>11.1</td>
<td>8.54</td>
<td>5.74</td>
<td>3.96</td>
<td>2.77</td>
<td>1.49</td>
</tr>
<tr>
<td>32</td>
<td>32.0</td>
<td>17.0</td>
<td>14.7</td>
<td>13.1</td>
<td>11.0</td>
<td>9.10</td>
<td>7.76</td>
<td>5.79</td>
<td>2.98</td>
<td>1.49</td>
</tr>
<tr>
<td>64</td>
<td>33.0</td>
<td>17.7</td>
<td>15.0</td>
<td>13.3</td>
<td>11.8</td>
<td>10.6</td>
<td>8.79</td>
<td>5.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>35.0</td>
<td>18.0</td>
<td>15.1</td>
<td>13.8</td>
<td>13.0</td>
<td>11.5</td>
<td>8.89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>35.4</td>
<td>18.1</td>
<td>15.5</td>
<td>14.6</td>
<td>13.7</td>
<td>11.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>36.4</td>
<td>18.2</td>
<td>15.9</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We measured the Latency across all four paths and determined the cost of accessing an adapter over the QPI link. For example it's more expensive to access Adapter1 from Cpu2 than it is from Cpu1.
These results demonstrate one of the benefits of having a second adapter installed on a machine with two CPUs. Intel has moved the I/O controller from a separate chip on the motherboard directly onto the (Xeon E5-2690) processor dies. Now every CPU has direct access to a local PCIe 3.0 adapter. Having one adapter for each CPU ensures that every thread in an application has optimal access to the network. It also frees the QPI link of the additional I/O traffic.

Using both adapters we were able to achieve nearly perfect linear scaling of message throughput across all 16 cores. See the SINGLE and DUAL adapter scaling results below for further details.

**Latency of the FDR switch**

We measured message Latency with and without the FDR switch in the path to determine the latency of the FDR switch itself. All measurements here were taken along **RDMA Data Path1**.

<table>
<thead>
<tr>
<th>Size</th>
<th>Switch Latency</th>
<th>No Switch Latency</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1034</td>
<td>841</td>
<td>193</td>
</tr>
<tr>
<td>16</td>
<td>1062</td>
<td>871</td>
<td>191</td>
</tr>
<tr>
<td>32</td>
<td>1087</td>
<td>900</td>
<td>187</td>
</tr>
<tr>
<td>64</td>
<td>1136</td>
<td>950</td>
<td>186</td>
</tr>
<tr>
<td>128</td>
<td>1220</td>
<td>1034</td>
<td>186</td>
</tr>
<tr>
<td>256</td>
<td>1797</td>
<td>1604</td>
<td>193</td>
</tr>
<tr>
<td>512</td>
<td>1952</td>
<td>1757</td>
<td>195</td>
</tr>
<tr>
<td>1024</td>
<td>2267</td>
<td>2073</td>
<td>194</td>
</tr>
<tr>
<td>2048</td>
<td>2894</td>
<td>2704</td>
<td>190</td>
</tr>
<tr>
<td>4096</td>
<td>3591</td>
<td>3404</td>
<td>187</td>
</tr>
</tbody>
</table>

The switch latency was roughly constant (186-195) for all message Sizes <= 4096.
Multi-Core Scaling of Message Throughput

We tested the scaling of message throughput using both single and dual adapter setups. Both tests used thirty two instances of the Bon Trade Network Component. Sixteen were configured to run on Host1 as message producers and sixteen were configured to run on Host2 as message consumers. Each instance was affinitized to a distinct/dedicated CPU Core.

Next we connected each producer on Host1 to a distinct consumer on Host2 giving 16 separate 1:1 connections. We made sure to connect the producer on Core K (Host1) to the matching consumer on Core K (Host2). We will refer to the producer on Core K (Host1) as producer[K], the consumer on Core K (Host2) as consumer[K] and the connection between producer[K] and consumer[K] as connection[K].

Under the single adapter setup only Adapter1 on each host machine was used. This resulted in the following 16 data paths:

- producer[0-7] -------------> Adapter1 --Switch-- Adapter1------------> consumer[0-7]
- producer[8-15]--> QPI --> Adapter1 --Switch-- Adapter1--> QPI --> consumer[8-15]

Connections[8-15] all cross a QPI link twice - first on Host1 and then again on Host2. Here half of the connections[0-7] were optimal and half were not.
Under the dual adapter setup we had connections[0-7] use Adapter1 and connections[8-15] use Adapter2. Here all 16 connections were optimal:

producer[0-7] --------> Adapter1 --Switch-- Adapter1--------> consumer[0-7]
producer[8-15]-------> Adapter2 --Switch-- Adapter2--------> consumer[8-15]

Message streams (for a given message Size) were then opened one at time on each of the 16 connections. After opening each stream we observed the flow on all other streams immediately rebalance. Specifically all connections sending data through the same adapter reported the same message rate(±1%) at all times. In the limit where the adapter bandwidth had been reached it was evenly distributed across all connections sending data through it. This is one of the remarkable properties of the InfiniBand transport.

We recorded the per-connection message rates after each stream was opened. In the dual adapter case, per-connection message rates through Adapter1 were completely independent of the per-connection message rates through Adapter2. For example, opening six 128-byte message streams on Adapter1 and three 128-byte message streams on Adapter2 resulted in six streams at 7.67M/s and three streams at 13.6M/s giving a total message rate of:

\[6 \times 7.67 + 3 \times 13.6 = 86.82 \text{ M/s} \] between the two machines (see the results for Size=128)
Below are the multi-core scaling results for message Sizes 12 and 128.
Single and Dual Adapter Scaling Results for 12 Byte Messages

### Single Adapter Scaling Results

<table>
<thead>
<tr>
<th># Cores</th>
<th>RateMax per Core</th>
<th>RateMax Total</th>
<th>Bandwidth Total (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>37.6</td>
<td>37.6</td>
<td>3.61</td>
</tr>
<tr>
<td>2</td>
<td>37.6</td>
<td>75.2</td>
<td>7.22</td>
</tr>
<tr>
<td>3</td>
<td>37.6</td>
<td>112.8</td>
<td>10.8</td>
</tr>
<tr>
<td>4</td>
<td>37.6</td>
<td>150.4</td>
<td>14.4</td>
</tr>
<tr>
<td>5</td>
<td>37.6</td>
<td>188.0</td>
<td>18.0</td>
</tr>
<tr>
<td>6</td>
<td>37.6</td>
<td>225.6</td>
<td>21.7</td>
</tr>
<tr>
<td>7</td>
<td>37.6</td>
<td>263.2</td>
<td>25.3</td>
</tr>
<tr>
<td>8</td>
<td>37.5</td>
<td>300.0</td>
<td>28.8</td>
</tr>
<tr>
<td>9</td>
<td>37.5</td>
<td>337.5</td>
<td>32.4</td>
</tr>
<tr>
<td>10</td>
<td>37.4</td>
<td>374.0</td>
<td>35.9</td>
</tr>
<tr>
<td>11</td>
<td>37.4</td>
<td>411.4</td>
<td>39.5</td>
</tr>
<tr>
<td>12</td>
<td>35.0</td>
<td>420.0</td>
<td>40.3</td>
</tr>
<tr>
<td>13</td>
<td>32.6</td>
<td>423.8</td>
<td>40.7</td>
</tr>
<tr>
<td>14</td>
<td>30.5</td>
<td>427.0</td>
<td>41.0</td>
</tr>
<tr>
<td>15</td>
<td>28.5</td>
<td>427.5</td>
<td>41.0</td>
</tr>
</tbody>
</table>

99% Linear Scaling to 11 cores
Bandwidth Limit reached with 75% utilization

### Dual Adapter Scaling Results

<table>
<thead>
<tr>
<th>Adapter 1</th>
<th>Adapter 2</th>
<th>12 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td># Cores</td>
<td>RateMax per Core</td>
<td>RateMax per Core</td>
</tr>
<tr>
<td>1</td>
<td>37.6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>37.6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>37.6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>37.6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>37.6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>37.6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>37.5</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>37.4</td>
<td>8</td>
</tr>
</tbody>
</table>

99% Linear scaling across 16 cores
## Single and Dual Adapter Scaling Results for 128 Byte Messages

### Single Adapter Scaling Results

<table>
<thead>
<tr>
<th># Cores</th>
<th>RateMax per Core</th>
<th>RateMax Total</th>
<th>Bandwidth Total (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.6</td>
<td>13.6</td>
<td>13.9</td>
</tr>
<tr>
<td>2</td>
<td>13.6</td>
<td>27.2</td>
<td>27.9</td>
</tr>
<tr>
<td>3</td>
<td>13.6</td>
<td>40.8</td>
<td>41.8</td>
</tr>
<tr>
<td>4</td>
<td>11.8</td>
<td>47.2</td>
<td>48.3</td>
</tr>
<tr>
<td>5</td>
<td>9.27</td>
<td>46.4</td>
<td>47.5</td>
</tr>
<tr>
<td>6</td>
<td>7.67</td>
<td>46.0</td>
<td>47.1</td>
</tr>
<tr>
<td>7</td>
<td>6.50</td>
<td>45.5</td>
<td>46.6</td>
</tr>
<tr>
<td>8</td>
<td>5.67</td>
<td>45.4</td>
<td>46.4</td>
</tr>
<tr>
<td>9</td>
<td>5.01</td>
<td>45.1</td>
<td>46.2</td>
</tr>
<tr>
<td>10</td>
<td>4.47</td>
<td>44.7</td>
<td>45.8</td>
</tr>
<tr>
<td>11</td>
<td>4.04</td>
<td>44.4</td>
<td>45.5</td>
</tr>
<tr>
<td>12</td>
<td>3.71</td>
<td>44.5</td>
<td>45.6</td>
</tr>
<tr>
<td>13</td>
<td>3.41</td>
<td>44.3</td>
<td>45.4</td>
</tr>
<tr>
<td>14</td>
<td>3.17</td>
<td>44.4</td>
<td>45.4</td>
</tr>
<tr>
<td>15</td>
<td>2.94</td>
<td>44.1</td>
<td>45.2</td>
</tr>
</tbody>
</table>

100% Linear Scaling to 3 cores
Bandwidth Limit reached with 89% utilization

### Dual Adapter Scaling Results

<table>
<thead>
<tr>
<th>Adapter 1</th>
<th>Adapter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Cores</td>
<td># Cores</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>

100% Linear scaling to 6 cores
Bandwidth Limit reached with 89% utilization
Single digit microsecond latency under extreme workloads

We measured the average (half-roundtrip) Latency while transferring over 520 Million 12-byte messages/sec. On each host, fourteen cores were used for the transmission and one core for the ping-pong operation (i.e. latency calculation). The remaining core was unused. These were the results:

<table>
<thead>
<tr>
<th>Size</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1436</td>
</tr>
<tr>
<td>16</td>
<td>1482</td>
</tr>
<tr>
<td>32</td>
<td>1513</td>
</tr>
<tr>
<td>64</td>
<td>1560</td>
</tr>
<tr>
<td>128</td>
<td>1654</td>
</tr>
<tr>
<td>256</td>
<td>2527</td>
</tr>
<tr>
<td>512</td>
<td>2683</td>
</tr>
<tr>
<td>1024</td>
<td>3011</td>
</tr>
<tr>
<td>2048</td>
<td>3432</td>
</tr>
<tr>
<td>4096</td>
<td>4165</td>
</tr>
</tbody>
</table>

We repeated the above latency test, transferring 90 Million 128-byte messages/sec (using fourteen cores per host again). Unlike the first test, the transfer here used all available bandwidth. There was little to no bandwidth left for the ping-pong operation and this increased the message Latency. Notice the plateau at small message sizes.

<table>
<thead>
<tr>
<th>Size</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4149</td>
</tr>
<tr>
<td>16</td>
<td>4197</td>
</tr>
<tr>
<td>32</td>
<td>4274</td>
</tr>
<tr>
<td>64</td>
<td>4353</td>
</tr>
<tr>
<td>128</td>
<td>4196</td>
</tr>
<tr>
<td>256</td>
<td>7644</td>
</tr>
<tr>
<td>512</td>
<td>8080</td>
</tr>
<tr>
<td>1024</td>
<td>8705</td>
</tr>
<tr>
<td>2048</td>
<td>9453</td>
</tr>
<tr>
<td>4096</td>
<td>9875</td>
</tr>
</tbody>
</table>

In both tests, the latency was below ten microseconds for all message sizes, 4096 bytes and smaller.
Conclusion

Our test results demonstrate the tremendous processing power afforded by today’s networking and multi-core CPU architectures. They also demonstrate the stability and ultra-high performance of the Windows Server 2008 operating system under extreme workloads.

The efficient design of the RMC messaging layer allows an application to take full advantage of the substantial bandwidth and ultra low latency offered by the shared memory and InfiniBand transports. Many messaging systems peak at 2-3 million messages/sec regardless of the underlying transport. In contrast, our solution is capable of achieving roughly 10-20 times this. Message rates through a high-bandwidth transport are ultimately decided by the efficiency of the messaging-system and not the underlying transport.

Client benefits of RMC messaging include the substantial reduction of internal network hop latency necessary for a high availability system. Leveraging this new technology can substantially reduce datacenter foot prints as full enterprise trading systems can now be deployed with fewer than half the servers previously required.

Together RMC, Mellanox’s FDR InfiniBand, Windows Server 2008 and HP ProLiant servers have set new performance standards that Bon Trade clients can immediately benefit from.
About Bon Trade

Bon Trade Solutions is an innovative Independent Software Vendor offering pre-trade risk managed order routing gateways and products that complement the trade lifecycle process. Our best in industry, market proven systems are designed to handle the current and future demands of global electronic markets.

For more information please contact:
John Paul DeVito
Director of Sales
jpd@bon-trade.com
www.bon-trade.com

About Mellanox

Mellanox Technologies (NASDAQ: MLNX, TASE: MLNX) is a leading supplier of end-to-end InfiniBand and Ethernet interconnect solutions and services for servers and storage. Mellanox interconnect solutions increase data center efficiency by providing the highest throughput and lowest latency, delivering data faster to applications and unlocking system performance capability. Mellanox offers a choice of fast interconnect products: adapters, switches, software and silicon that accelerate application runtime and maximize business results for a wide range of markets including high performance computing, enterprise data centers, Web 2.0, cloud, storage and financial services. More information is available at www.mellanox.com.

Mellanox, BridgeX, ConnectX, CORE-Direct, InfiniBridge, InfiniHost, InfiniScale, PhyX, SwitchX, Virtual Protocol Interconnect and Voltaire are registered trademarks of Mellanox Technologies, Ltd. FabricIT, MLNX-OS, Unbreakable-Link, UFM and Unified Fabric Manager are trademarks of Mellanox Technologies, Ltd. All other trademarks are property of their respective owners.

About HP

HP creates new possibilities for technology to have a meaningful impact on people, businesses, governments and society. The world’s largest technology company, HP brings together a portfolio that spans printing, personal computing, software, services and IT infrastructure to solve customer problems. More information about HP (NYSE: HPQ) is available at www.hp.com/go/hpc and www.hp.com/go/proliant.
Credits

Testing and Result Analysis:

Thomas McSherry, Chief System Architect/ Bon Trade Solutions
William McSherry, Chief Developer/ Bon Trade Solutions
Robert Colucci, Software Engineer/ Bon Trade Solutions
Sagi Schlanger, Manager POC Engineering/ Mellanox Technologies
Lee Fisher, WorldWide Business Development for FSI/ HP
Chuck Newman, Low Latency Performance Engineer/ HP
Mike Nikolaiev, High Performance Evaluation Lab/ HP

Authors:

Thomas McSherry, Chief System Architect/ Bon Trade Solutions
Victor Tartaglia, Managing Director/ Bon Trade Solutions

Design:

Anthony Bove, Systems Engineer/ Bon Trade Solutions