Red Hat Reference Architecture Series

# Performance and Scalability of the RMDS Market Data Platform

## (based on Reuters RMDS)

| |
|---|
| Reuters RMDS 6 |
| RRCP |
| Red Hat Enterprise Linux 4 / 5 |
| Intel Xeon (x86-64) |
| 1 Gigabit Ethernet |

Version 1

April 2008

redhat.

**Performance and Scalability of the RMDS Market Data Platform (based on Reuters RMDS)**

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA



The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

# Table of Contents

# 1. Executive Summary

In the financial services industry, automated trading is a race – firms develop sophisticated strategies and attempt to beat the competition by analyzing and acting on market information faster than other traders. The quicker the systems, the better the financial results. For these financial services firms, one of the keys to maximizing portfolio performance is reducing data latency, or the time it takes for the market information to get to the trading applications. RMDS (Reuters Market Data System) is a high throughput, low latency platform for market data applications. This reference architecture is Phase I of a study of the performance, scalability, tuning, and optimization of RMDS running on RHEL® (Red Hat® Enterprise Linux®).

All the performance tests described herein used the Reuters RMDS performance testbed. Typically these tests are run using configurations of 3 or more nodes – one node runs the load driver, the second node runs the RMDS infrastructure component that publishes market data (= source distributor), and a third node runs the RMDS infrastructure component that subscribes to market data (= Point-to-Point server = P2PS). Due to a shortage of hardware and being pressed for time, it was decided to run the latter two nodes on a single physical node with a loop-back ethernet connection to simulate the network hop. However, before deciding on using the 2-node configuration, a series of tests were run to position the relative performance of the 2-node and 3-node test configurations. As documented in this report, the 3-node configurations always outperformed the 2-node configurations because the cpu became the bottleneck in the 2-node configuration.

- With these results we are confident that real-world 3-node configurations would perform better than any 2-node configurations tested in the lab. With this assurance, we proceeded to use the 2-node configuration for the RHEL 4 versus RHEL 5 comparison.

Today, many of our existing customers use RMDS on RHEL-4. For many of these customers who are considering moving from RMDS-5 / RHEL-4 to RMDS-6 / RHEL-5, we wanted to ensure continued leadership performance. One of the primary goals of this study was to ensure that there would be no performance or other problems when moving RMDS from RHEL-4 to RHEL-5. The results clearly show:

- In the configurations tested, there is no statistically significant performance regression when going from RMDS-6 / RHEL-4.4 to RMDS-6 / RHEL-5.1
- Process and Interrupt binding to cpu cores did not make any appreciable difference in performance
- Setting TSO (TCP Segmentation Offload) to “on” and “off” did not make any appreciable difference in performance

The configurations employed for the tests in this report used 2-core Intel® Xeon® processors and a limited number of 1GigE NICs (Network Interface Cards). In follow-on testing, the goal is to use multi-core servers with a larger number of NICs. This will enable us to more fully explore the impact of binding instances RMDS infrastructure processes and NIC interrupts to cpu cores in different topologies

The Securities Technology Analysis Center (STAC®) is a provider of performance measurement services, tools and research to the securities industry. A summary of some recent STAC reports about RMDS/RHEL throughput and latency are included in the appendix. STAC reports are available at: http://www.STACresearch.com.

STAC, on 18-March-2008, reported that RMDS 6 / RHEL 5.1 running on IBM BladeCenter® H with HS21 Blades and using Blade Network Technologies (BNT) 10 Gigabit Ethernet Switch and Chelsio Communication's 10 Gigabit Ethernet Network Interface Card (NIC) achieved:

1. Lowest mean latency ever reported with RMDS
   - Less than 0.9 milliseconds of end-to-end infrastructure latency at up to 600,000 updates per second in the low-latency configuration of RMDS
2. Lowest standard deviation of latency ever reported with RMDS
   - Less than 0.5 milliseconds at rates up to 600,000 updates per second.
3. Very high output rate in the — "Producer 50/50" fanout test of a stacked P2PS
   - 5.8 million updates per second
   - 30% of this due to the TCP/IP Offload Engine (TOE) in the Chelsio NIC Blade Network Technologies 10GigE Switch

Based on recent STAC reports at www.STACresearch.com/redhat and a Sun® report dated November 2007 www.sun.com/third-arty/global/reuters/collateral/Benchmark_WP_110207.pdf a comparison of RMDS mean latency performance on Solaris/x86-64 and RHEL/x86-64 is shown below.

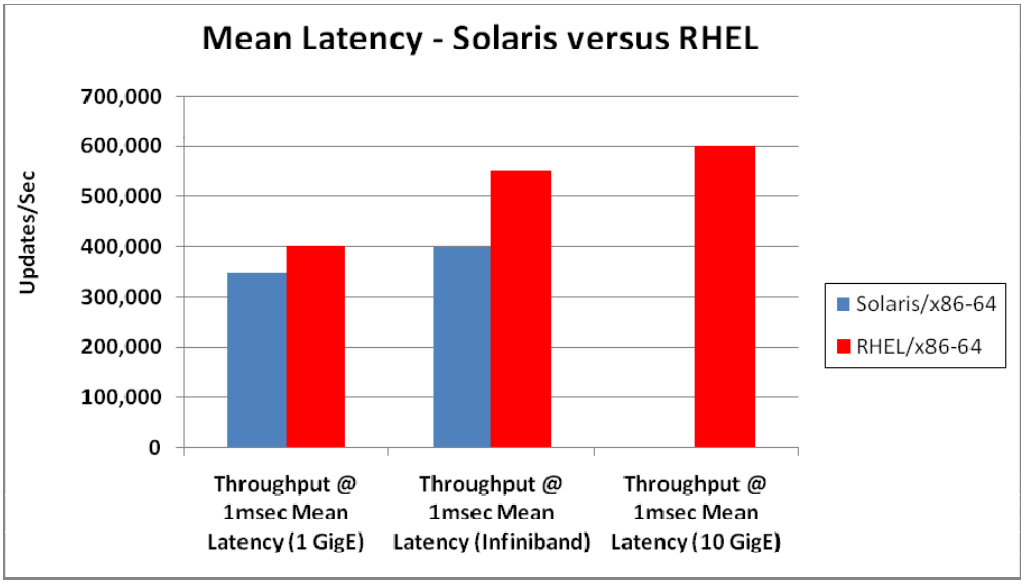


***Figure 1***

Similarly, data in these reports shows RHEL/x86-64 outperforming Solaris/x86-64 in the RMDS throughput delivered.

# 2. RMDS: Introduction and Architecture

## 2.1 Growing Demand for High Throughput and Low Latency

The rapid growth of data traffic in the capital markets industry continues to be a major concern for technologists, as they attempt to deal with the dual requirements of higher throughput and lower latency.

Market data latency has a huge impact on the overall speed with which a trading firm can execute a transaction in response to new information. In some markets, firms can profit from as little as one millisecond of advantage over competitors, which drives them to find sub-millisecond optimizations of the systems fueling their trades. The latency obsession has resulted from the spread of automated trading to nearly every geography and asset class, and the resulting imperative to exploit—or defend against—new latency arbitrage opportunities.

Another consequence of automated trading is a ballooning of market data traffic volumes, which complicates the latency race, thanks to a well-established tradeoff between throughput and latency. Update-rate increases of 2 to 6 times in a single year are not uncommon for today's exchanges. Automated trading drives this traffic by both increasing transaction volumes and increasing the ratio of quotes and cancellations to actual trades. While North American venues still produce the most traffic, many observers expect the Markets in Financial Instruments Directive (MiFID) to trigger a sharp increase in European traffic as the number of trade-reporting venues proliferates. On top of this, large sell-side institutions often generate enormous amounts of real-time data internally, which they pump onto their internal market data system. The traffic from internal content sometimes exceeds that of information coming in from external sources.

This combination of forces keeps market data technologists on the lookout for new technologies that can shift the performance tradeoffs in the right direction.

## 2.2 RMDS

The Reuters Market Data System (RMDS) provides a platform for efficient and reliable distribution of Reuters, value-added and third-party data to client applications. In response to the market demand for reduced total cost of ownership, improved performance, flexibility and ease of implementation, Reuters supports the next-generation version of Reuters Market Data System (RMDS 6.0) on Red Hat Enterprise Linux (RHEL) on both the Intel Xeon and AMD Opteron x86-64 platforms.

RMDS is a modular system consisting of several products which can be introduced incrementally. These products provide a number of powerful capabilities.

- **Point-to-Point delivery (P2PS)** – ideal for high volume applications that require a

dedicated distribution for desktops.

- **Multicast delivery (RTIC: TIC-RMDS Edition)** – ideal for desktop applications with high commonality.
- **Internet delivery (IFP: Internet Finance Platform)** – ideal for applications with low update frequency integration requirement.
- **Reuters Wireless Delivery System (RWDS)** – provides streaming access to Reuters and internal data on the BlackBerry.
- **Wide-Area Networking (WAN)** – ideal for efficiently sharing data with remote users or branches.

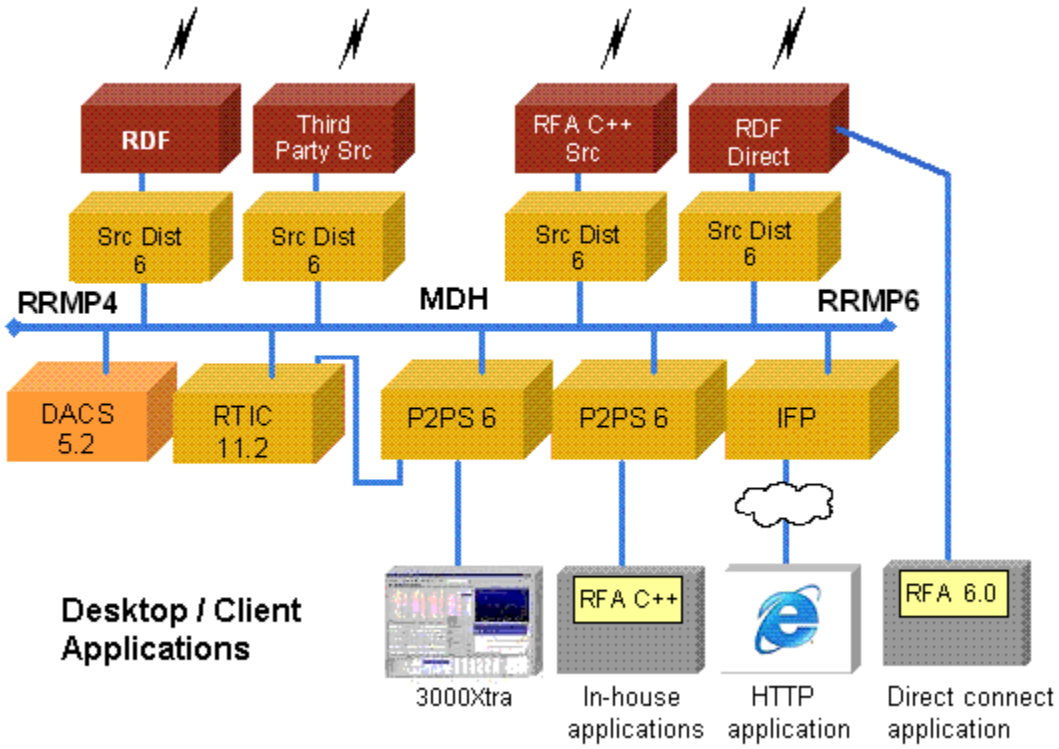


***Figure 2***

This new version of this low-latency market data platform has been comprehensively enhanced to give increased performance and increased flexibility. RMDS 6.0 adds support for new data formats (OMM/RWF), the **Reuter Data Feed Direct (RDF Direct)** and the **Reuters Foundation API 6.0 (RFA)**.

Reuters has introduced the **Open Message Model (OMM)** and **Reuters Wire Format (RWF)** in the RMDS 6.0 – a new and open set of data modeling tools which deals with complex data types. This new message model gives Reuters customers and partners the ability to publish data across their enterprise in a wide variety of formats via the RMDS platform. RWF, the new binary message format supported in RMDS 6, substantially reduces the size of the market data updates. This increased efficiency allows RMDS 6 to support higher update rates and comfortably manage the rapid growth in real-time update rates in the market.

RMDS 6.0 delivers significant performance benefits like reduced latency, increased throughput, reduction in network bandwidth requirement and a drop in image sizes.

RMDS 6.0 is also backward compatible with older version of RMDS. This means that customers can implement RMDS 6 components alongside RMDS 5 elements, allowing new investment to be focused on components which need new functionality or higher performance, while continuing to exploit previously deployed components. This flexibility is underpinned by the support for Red Hat Enterprise Linux across both RMDS 5 and RMDS 6. This allows skills, experience and capabilities to be re-used across both RMDS 5 and RMDS 6 while allowing customers to choose the appropriate hardware platform – for each new component.

The two key components that are the subject of this performance and scalability study are:

1. Source Distributor
2. P2PS (Point-to-Point Server)

## 2.2.1 Source Distributor

The Source Distributor is a key component used by all source server publishers in order to make their content available on the **Market Data Hub (MDH)**. It implements source specific features such as recovery, data quality, state management, load balancing, source mirroring, resource management and support for interactive/broadcast publishers.

Optionally, the Source Distributor is also capable of maintaining a cache of items so that new requests for currently serviced items can be satisfied without interaction with the source application. In this case the Source Distributor automatically applies any updates written by the application to the cache. The Source Distributor can simultaneously serve a configurable number of independent source applications for the same or different services.

The Source Distributor also optionally supports field and update filtering. Certain fields can be removed from the data and time-based conflation can be used to cut down on bandwidth requirements.

The Source Distributor can simultaneously support a configurable number of independent source applications for the same or different services. Source applications use the RFA or legacy SSL Library to communicate with the Source Distributor infrastructure component.

Source applications can run on the same node as the Source Distributor or they can run on a remote node if performance/security (e.g. firewalls) requirements warrant the added expense.

Source Distributor 6.0 supports standard RMDS capabilities such as Source Mirroring, Preemption and Load Balancing for OMM sources. It also supports an optional data cache for OMM, Marketfeed Record and ANSI Page data.

MDH is the RMDS backbone consisting of components that communicate using **RRMP (Reuters Reliable Management Protocol) & RRCP (Reuters Reliable Communication Protocol)**, i.e. Source Distributor, P2PS, and RTIC.

## 2.2.2 P2PS (Point-To-Point Server)

The P2PS combines an optional high-speed in-memory data cache with intelligent distribution

semantics used to implement the Point-to-Point Client LAN. The P2PS provides consolidated point-to-point access to all the information available on the Market Data Hub. It consumes data provided on the MDH and/or the Multicast Client LAN, optionally caches it and then re-distributes the data (using TCP/IP) to consumer applications upon request. This protects the consuming applications from the complexities of the multicast based traffic of the MDH and Client LAN. The P2PS supports all forms of simple publishing.

The P2PS can handle all forms of data, such as quotes, chains, time-and-sales, news stories and headline streams, etc. Access to P2PS data is authorized through **DACS (Data Access Control System)** and enforced within the P2PS.

Applications typically have backup Point-to-Point Servers in order to provide resiliency. When an application loses connectivity to a P2PS, it will attempt to connect to a backup P2PS and re-open all of the items of interest (also known as failover). During initialization an application will periodically try each P2PS (in its configured list) in a 'round robin' fashion until it successfully establishes communication with one of them.

P2PS 6 supports connections with SSL-based client applications, RFA 5.0 and RFA 6.0. It provides access to all of the data types currently available to SSL-based applications and the new OMM data. In addition, P2PS 6 automatically converts RWF in the MarketPrice domain to Marketfeed for SSL-based client applications. It also converts Marketfeed data to RWF so that RFA 6.0 applications do not need to parse Marketfeed.

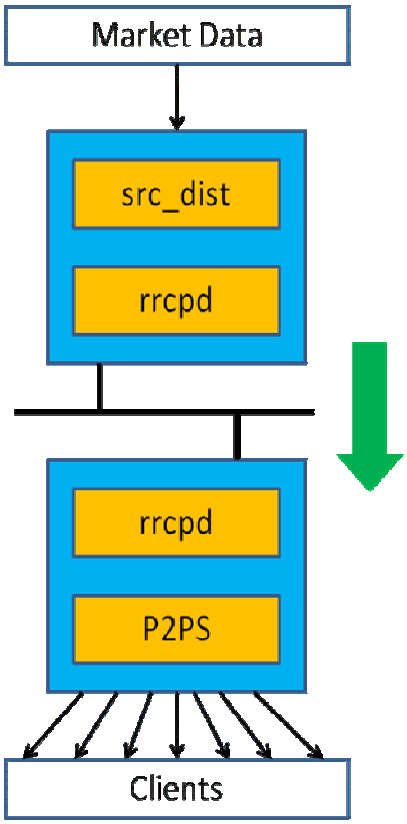


***Figure 3***

# 3. Testing Methodology

## 3.1 Two-Node Configuration

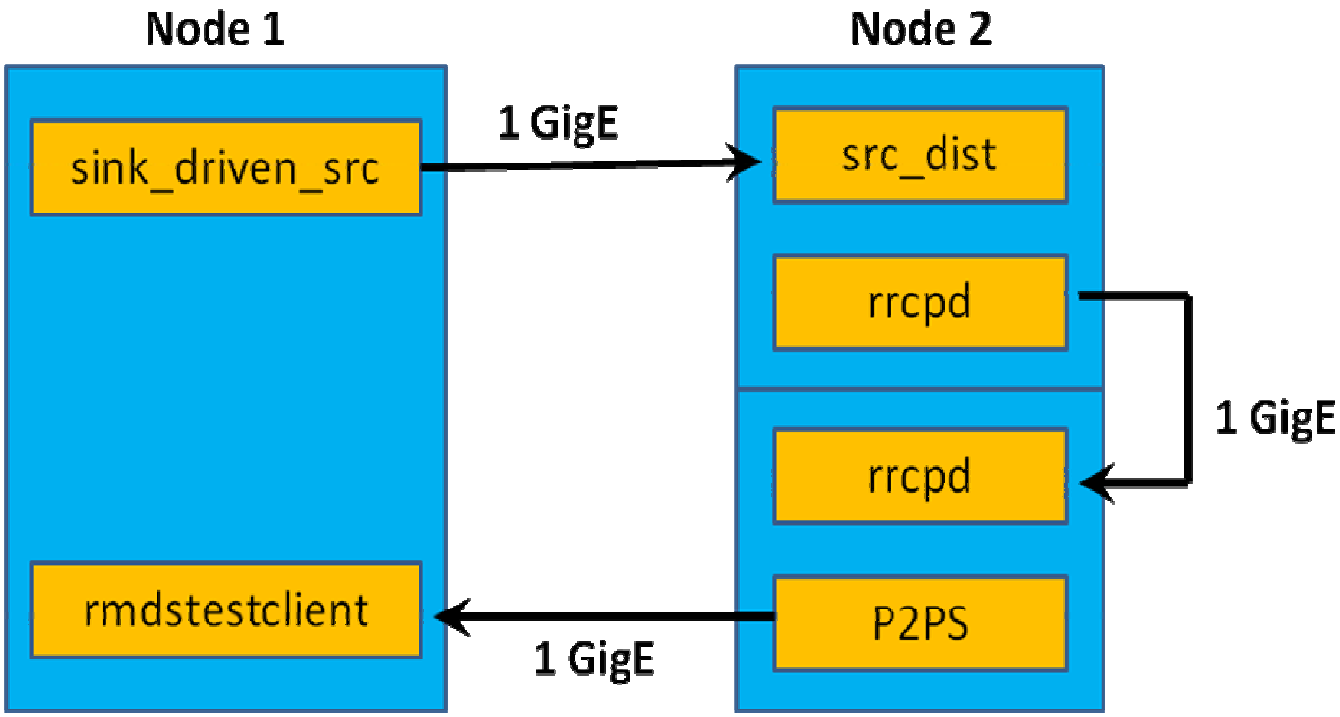


*Figure 4*

## 3.2 Three-Node Configuration

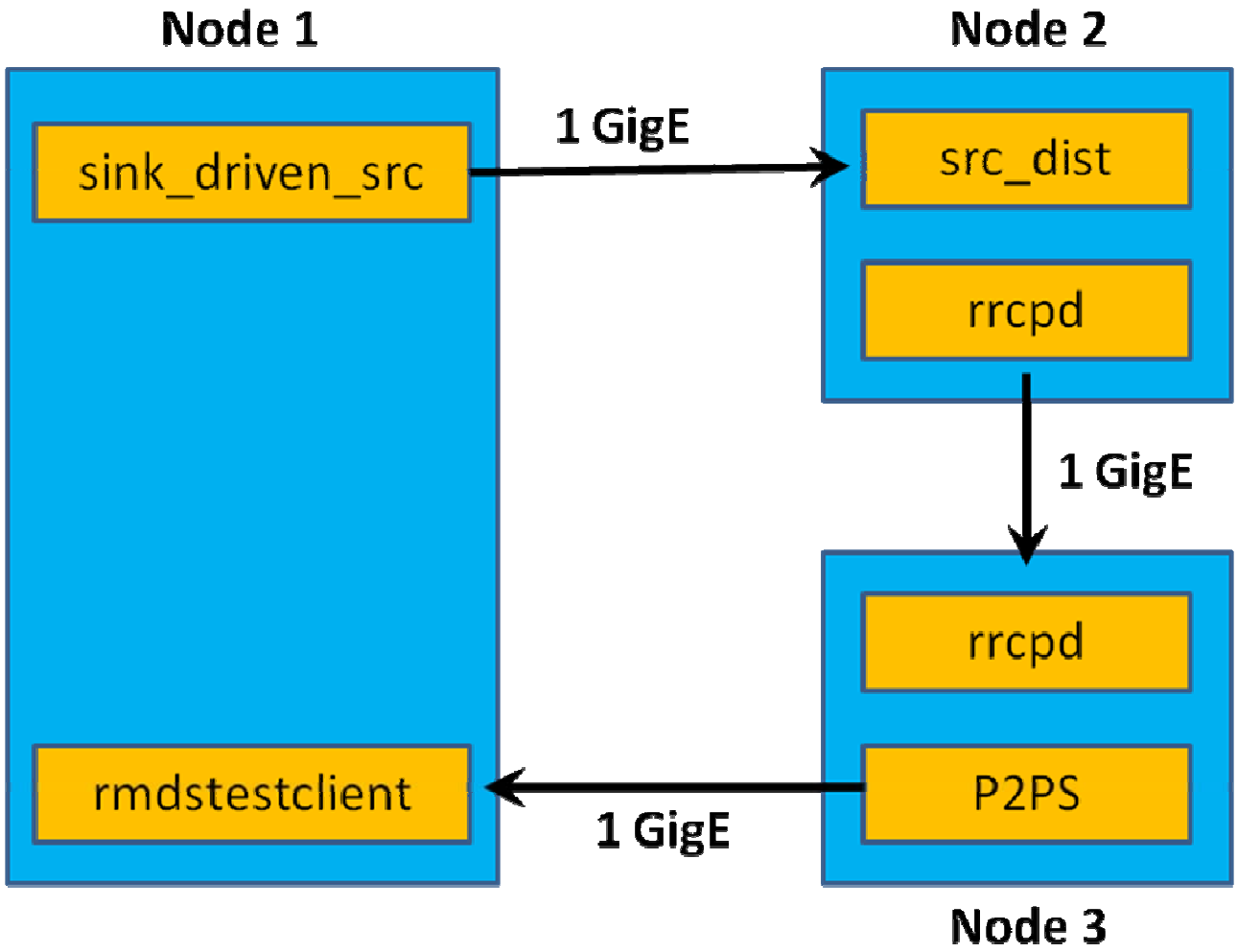


*Figure 5*

## 3.3 Publisher Application: Sink_Driven_Src

Sink_Driven_Src is a single threaded source application which can be used for both throughput and latency testing. Sink_Driven_Src provides simulated data with automated responses and updates. Sink_Driven_Src is used in conjunction with RMDSTestClient.

## 3.4 Subscriber Application: RMDSTestClient

RMDSTestClient is an RSSL/SSL consuming application that can request instruments listed in a file and log (optionally) the data to console or file. (RSSL = Reuters SSL protocol.) RMDSTestClient is a single threaded test tool. By default this tool decodes only the message header for updates

## 3.5 RMDS Components Being Tested: Src_Dist, P2PS

All experiments used a **single instance each of**:

1. **Source Distributor (Src_Dist)** and
2. **Point-To-Point Server (P2PS)**

## 3.6 Middleware / Transport: RRCP

Src_Dist and P2PS may communicate using either:

1. TIBCO Rendezvous
2. **Reuters Reliable Communication Protocol (RRCP)**

The tests in this study use RRCP. Physically, an instance of Src_Dist and an **RRCP daemon (RRCPD)** communicate with another instance of RRCPD and P2PS.

## 3.7 Data Format: RWF

There was a choice:

1. MarketFeed (MF) update size of 140 bytes, and an equivalent
2. Reuters Wire Format (RWF) update size of 74 bytes.

The tests in this study used the RWF format.

## 3.8 Performance Metrics: Update Rate & Latency

For throughput testing, the Sink_Driven_Src utility was used to generate update traffic, and the RMDSTestClient utility was used to consume the updates.

The infrastructure is tuned for maximum throughput, and the update rate was increased until the CPU limit was reached with no errors reported.

### 3.8.1 End-to-End Latency

Latency is defined as the time for a data item to propagate through one or more RMDS components. “End to end” latency is defined as the delta between the time an update is posted by the publisher application to an RMDS API and the time the same update is received by the consuming application from an RMDS API, i.e. it includes both the latency contribution from the API and the core infrastructure components.

The results below include the mean, standard deviation, and minimum and maximum latencies measured for the tests. While looking at latency results, it is important to not only look at the average latency but at the distribution of latency. Standard deviation is the most commonly used measure of spread. In a normal distribution, about 68% of the data points are within one standard deviation of the mean and about 95% of the data points are within two standards deviations of the mean.

The embedded timestamp approach was used to calculate end-to-end latency (Quotes and

Trades) data. RMDS 6.0 end-to-end update latency is measured by using Sink_Driven_Src as the publisher and RMDSTestClient as the subscriber. In the embedded timestamp approach, the publisher embeds timestamps into selected updates that the subscriber uses for latency calculations. In this scenario, the publisher and subscriber must be running on the same node for accurate timestamps.

## 3.9 Watchlist(s)

Each subscribing client (connected to the P2PS) can specify a watchlist of securities. Any updates the P2PS receives relating to a security on a watchlist is forwarded to (all) the clients which have that security on their watchlist. Long watchlists can significantly increase the load on the P2PS. The worst case scenario is when there are a lot of clients connected to a P2PS, each with a long watchlist and a high degree of commonality among the watchlists of the clients. In this case each update the P2PS receives must be forwarded to many clients.

For experiments in this study we used a 100,000 item watchlist.

## 3.10 Process and Interrupt Binding

Tests have shown that there is less fluctuation in the update rate if the RMDS infrastructure components are bound to individual CPU(s). In addition to such binding, Reuters has found that binding NIC interrupts to a single CPU helps achieve better CPU utilization on the System Under Test (SUT) node(s). [By default, on a dual NIC dual CPU SUT node, interrupts from eth0 are bound to CPU 0 and interrupts from eth1 are bound to CPU 1.]

The optimization tried was as follows: Since the RMDS transport daemons (RRCPD) use less CPU than the Source Distributor or P2PS at a given update rate, we ran experiments with NIC interrupts from both eth0 and eth1 bound to CPU 0, thereby making CPU 1 completely available for the Source Distributor or P2PS main thread.

Binding NIC interrupts to cpu cores is done using:

**# echo** <*cpu_core_hex_mask*> **/ proc / irq /** <*interrupt #*> / **smp_affinity**

Binding of RMDS processes to cpu cores was done using:

**# taskset** <*cpu_core_hex_mask*> **-p** <pid>

## 3.11 IRQbalance

IRQbalance is a Linux daemon that distributes interrupts over the processors and cores. The design goal of IRQbalance is to find a balance between power savings and optimal performance.

In the runs where process/interrupts were bound to cpu cores, IRQbalance is also disabled.

This command will stop the irqbalance service immediately and prevent it from starting on subsequent reboots. There is no need to reboot the system after making this change.

**# chkconfig irqbalance off**

## 3.12 TCP Segmentation Offload (TSO)

TCP Segmentation Offload (TSO) is used to reduce the CPU overhead of TCP/IP on fast networks. TSO breaks down large groups of data sent over a network into smaller segments that pass through all the network elements between the source and destination. This type of offload relies on the NIC to segment the data and then add the TCP, IP and data link layer protocol headers to each segment. The NIC must support TSO.

[TSO is “off” by default in RHEL 4.4 and TSO is “on” by default in RHEL 5.1]

Previous testing has shown that TSO being set to “on” or “off” can sometimes have a performance impact. Tests performed in this study were run with TSO “on” and “off” to confirm that there was no performance impact.

TSO is shut off using the command:

```
# ethtool -K eth* tso off
```

# 4. Hardware/Software Versions & Configurations

## 4.1 Hardware

The test harness consisted of Xeon-based HP Servers sufficient to generate load on the target System(s) Under Test (SUT). Load Driver and Target SUT specification are as follows.

### 4.1.1 Load Driver System

**Node 1: Dual core DL320p.**

One computer used for test applications only (Sink_Driven_Src and RMDSTestClient) in any kind of 3-node configuration and in 2-node configuration when running RMDS 6.0.3.L1 on RHEL 4.4 and on RHEL 5.1

| | |
|---|---|
| Vendor Model: | HP Proliant DL320 G5p |
| Processors: | 1 |
| Processor type: | Dual-Core Intel® Xeon® X3075, 2.66GHz |
| Cache: | 4GB Level 2 |
| Bus speed: | 1333MGz |
| Memory: | 3x1GB PC2-6400 |
| Eth0: | NC110T PCIe Gigabit Adapter |
| Eth1: | NC326i Dual Integrated Gigabit NIC |
| Eth2: | NC326i Dual Integrated Gigabit NIC |
| Disk1: | 72GB 15K SAS |
| Disk2: | 160GB 7.2K SATA |

### 4.1.2 System(s) Under Test

**Node 2 & Node 3 in 2-Node versus 3-Node Comparison: Dual core DL320 .**

Two computers used in 2-node and 3-node configuration when running RMDS 6.0.2.L2 on RHEL 4.4

| | |
|---|---|
| Vendor Model: | HP Proliant DL320 G5 |
| Processors: | 1 |
| Processor type: | Dual-Core Intel® Xeon® 3050, 2.13GHz |
| Cache: | 2MB Level 2 |
| Bus speed: | 1066MGz |
| Memory: | 1x1GB PC2-5300 |
| Eth0: | Embedded NC324i Dual Port Gigabit NIC |
| Eth1: | Embedded NC324i Dual Port Gigabit NIC |
| Eth2: | NC324i Dual Port Gigabit NIC |
| Eth3: | NC324i Dual Port Gigabit NIC |
| Disk Controller: | Integrated Intel® 82801GR Serial ATA |
| Disk: | 80GB SATA 1.5GB 7,200 rpm |

**Node 2 in RHEL 4.4 versus RHEL 5.1 Comparison: Dual core DL320p.**

One computer running Src_Dist and P2PS processes in 2-node configuration - RMDS 6.0.3.L1 on RHEL 4.4 and on RHEL 5.1.

| | |
|---|---|
| Vendor Model: | HP Proliant DL320 G5p |
| Processors: | 1 |
| Processor type: | Dual-Core Intel® Xeon® X3075, 2.66GHz |
| Cache: | 4GB Level 2 |
| Bus speed: | 1333MGz |
| Memory: | 3x1GB PC2-6400 |
| Eth0: | NC326i Dual Integrated Gigabit NIC |
| Eth1: | NC326i Dual Integrated Gigabit NIC |
| Eth2: | NC360T PCIe Dual Port Gigabit Adapter |
| Eth3: | NC360T PCIe Dual Port Gigabit Adapter |
| Disk1: | 72GB 15K SAS |
| Disk2: | 160GB 7.2K SATA |

## 4.2 Network

All of the wires were run directly between Ethernet sockets using CAT6 crossover cables. No Ethernet switches were used. Each peer-to-peer connection belonged to a separate network.

| | |
|---|---|
| Txqueuelen | 5000 |

## 4.3 Software

### 4.3.1 RMDS 6.0.2.L2

| | |
|---|---|
| RMDS 6 software | Src_Dist ver. 6.0.2.L2<br>P2PS ver. 6.0.2.L2 |
| RMDS 6 Test Tools | Sink_Driven_Src from Src_Dist ver. 6.0.2.L2<br>RMDSTestClient from P2PS ver. 6.0.2.L2 |

### 4.3.2 RMDS 6.0.3.L1

| | |
|---|---|
| RMDS 6 software | Src_Dist ver. 6.0.3.L1<br>P2PS ver. 6.0.3.L1 |
| RMDS 6 Test Tools | Sink_Driven_Src from Src_Dist ver. 6.0.3.L1<br>RMDSTestClient from P2PS ver. 6.0.3.L1 |

### 4.3.3 RHEL 4.4

| | |
|---|---|
| Operating system | Red Hat Enterprise Linux 4 AS x86 update 4 |
| Kernel | 2.6.9-42.ELsmp |

### 4.3.4 RHEL 5.1

| | |
|---|---|
| Operating system | Red Hat Enterprise Linux Server x86 ver. 5.1 |
| Kernel | 2.6.18-53.1.14.el5xen |

### 4.3.5 RHEL & RMDS Configuration & Parameter Settings

RedHat Linux kernel configurations were modified by Reuters recommendations before running latency tests:

| | |
|---|---|
| OS Configuration | net.core.rmem_max = 8388608<br>net.core.rmem_default = 8388608<br>net.core.wmem_max = 8388608<br>net.core.wmem_default = 8388608<br>net.ipv4.tcp_mem = 4096 8388608 16777216<br>net.ipv4.tcp_rmem = 4096 8388608 16777216<br>net.ipv4.tcp_wmem = 4096 8388608 16777216<br>net.ipv4.ip_local_port_range = 34800 65535 |

RMDS Configuration Parameters were set by Reuters recommendations listed in the document “RMDS Core Infrastructure 6.0 – Demo Tools” Issue 0.4, 5 September 2006., paragraph “5.3 Configuration Parameters”, pp 10-11.

Before starting RMDS components, the appropriate entries are set in the configuration file (default location is /var/reuters/rmds/rmds.cnf).

| | |
|---|---|
| RMDS Configuration | The default RMDS configuration was modified as directed in the document: “RMDS Core Infrastructure 6.0. Demo Tools. Issue 0.4. 5 September 2006”<br><br>{node}*Src_Dist*{route_inst}.route*hostList : <sinkDrivenSrc_hostname><br>{node}*Src_Dist*{route_inst}.route*port : 14002<br>{node}*Src_Dist*{route_inst}.route*sslMountVersion : 5<br><br>*P2PS*14002*clientToServerPings : False<br>*P2PS*14002*pingTimeout : 30<br>*P2PS*14002*serverToClientPings : False<br>*P2PS*aggregateItemLimit : 100000<br>*P2PS*allowCompMode : True<br>*P2PS*allowRSSLConnections : True<br>*P2PS*allowSSLConnections : True<br>*P2PS*compressionType : 0<br>*P2PS*disableHostLookup : True<br>*P2PS*flushInterval : 0<br>*P2PS*guaranteedOutputBuffers : 1600<br>*P2PS*hashTableSize : 200000<br>*P2PS*itemLimit : 100000<br>*P2PS*maxMounts : 100<br>*P2PS*maxOutputBuffers : 1600<br>*P2PS*minimumOpenWindow : 500<br>*P2PS*outputThresholdBreach : 30<br>*P2PS*outputThresholdOK : 10<br>*P2PS*pingInterval : 10<br>*P2PS*poolSize : 2000 |

| | |
|---|---|
| | *P2PS*portList : 8101<br>*P2PS*rrmpProtocols : rrmp4and6<br>*P2PS*rsslMsgPacking : True<br>*P2PS*rsslPort : 14002<br>*P2PS*sessionStatsWindow : 1<br>*P2PS*sslMsgPacking : True<br>*P2PS*tcpNoDelay : True<br>*P2PS*tcpSendBufSize : 64240<br>*P2PS*timedWrites : False<br><br>*RRCP*switchReorderFixEnabled : False<br><br>*{serviceName}*cacheType : sinkDriven<br>*{serviceName}*dataType : 6<br>*{serviceName}*rrmpFlushInterval : 0<br><br>*Src_Dist*server*ipc*transmissionBus*inputBias : 60 |
| Affinities | When trying to achieve maximum performance the next settings were used:<br>All of Ethernet interrupts were bound to core 0.<br>RRCPD processes were bound to core 0.<br>Src_Dist and P2PS processes were bound to core 1. |

## 4.4 Test Tools and Test Data Files

Sink_Driven_Src was used as a data simulator and the file sample.xml (located in <mdh_path>/demo/bin) was used as a data file for the simulator. Connection type RSSL (Reuters SSL) and data format RWF were used for all of tests. The next command was used to run Sink_Driven_Src:

*./Sink_Driven_Src -S RDFD -N 14002 -Q sample.xml -q GOOG.O -K -c -tcpnodel -ts -tsn RTRSY.O -UL 10 -tps 500*

after the program was started the interactive command “ur <update rate>” was used in order to get results for every particular update rate.

RMDSTestClient was used as a test client application and file rdf.100kitems was used as an item file for the test client. The next command was used to run RMDSTestClient:

*./RMDSTestClient -S RDFD -h <P2PS host>_2 -p 14002 -md 6 decodeFormat 3 -l 1 -o 500 -u rmds -tt -lmfile <result_file> -f rdf.100kitems -ct rssl -rf 12*

## 4.5 Disabled RHEL Services and Monitoring Tools

The following services were disabled when running the tests:

1. Selinux

2. Crond
3. Ntpd
4. cpuspeed

The top command was used to monitor CPU and memory usage when running tests.

# 5. 2-Node versus 3-Node Configuration Comparison - Latency Test Results

Every test was ran a little bit longer than 10 minutes and processed about 6,050 samples. Before the results were processed, 300-600 of the first samples were discarded. This is because only after 35-50 seconds from launch does the test start producing samples with the required update rate. See Appendix I.

## 5.1 Two-Node Configuration (RHEL 4.4) Latency

This shows the results with irqbalance running and without any processes and interrupt bindings to CPUs.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 231.653 | 45.6364 | 1046 | 163 | 5080 |
| 5000 | 300.891 | 41.554 | 788 | 218 | 5716 |
| 10000 | 378.999 | 42.0595 | 814 | 304 | 5700 |
| 20000 | 489.698 | 51.8419 | 1295 | 370 | 5690 |
| 30000 | 612.822 | 72.2084 | 1509 | 376 | 5728 |
| 40000 | 710.484 | 51.8994 | 1369 | 468 | 5690 |
| 50000 | 778.228 | 83.608 | 2774 | 443 | 5680 |
| 60000 | 888.474 | 102.604 | 1908 | 398 | 5700 |
| 70000 | 958.175 | 113.968 | 2121 | 387 | 5670 |
| 80000 | 1027.57 | 137.694 | 2591 | 399 | 5650 |
| 90000 | 1130.83 | 174.21 | 3207 | 461 | 5670 |
| 100000 | 1210.24 | 198.731 | 3493 | 438 | 5660 |
| 150000 | 1718.59 | 454.742 | 6795 | 430 | 5690 |
| 200000 | 2202.87 | 632.246 | 9972 | 493 | 5680 |

## 5.2 Two-Node Configuration (RHEL 4.4) Latency – with Process & Interrupt Binding

This shows the results with irqbalance disabled and Ethernet interrupts bound to CPU 0, RRCPD bound to CPU 0, Src_Dist and P2PS bound to CPU 1. Also, in contrast with all other tests in this section which were running RMDS software version **6.0.2.L2** this set of tests was run using RMDS and RMDS Test Tools version **6.0.3.L1**.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 192.458 | 32.9876 | 2026 | 179 | 5684 |
| 5000 | 250.179 | 39.164 | 2099 | 233 | 5692 |
| 10000 | 320.132 | 45.9514 | 2030 | 298 | 5629 |
| 20000 | 446.084 | 45.1649 | 2029 | 353 | 5610 |
| 30000 | 579.942 | 52.3071 | 2014 | 385 | 5650 |
| 40000 | 638.432 | 50.0861 | 1308 | 380 | 5639 |
| 50000 | 716.657 | 88.995 | 2201 | 376 | 5669 |
| 60000 | 825.057 | 102.589 | 2277 | 382 | 5668 |
| 70000 | 892.875 | 123.172 | 2829 | 381 | 5650 |
| 80000 | 969.464 | 146.136 | 2737 | 378 | 5650 |
| 90000 | 1076.39 | 179.055 | 4165 | 395 | 5520 |
| 100000 | 1157.87 | 242.476 | 6035 | 399 | 5610 |
| 150000 | 1642.39 | 497.969 | 9439 | 400 | 5489 |
| 200000 | 2132.48 | 580.424 | 10064 | 504 | 5700 |

## 5.3 Three-Node Configuration (RHEL 4.4) Latency

This shows the results with irqbalance running and without any processes and interrupt bindings to CPUs.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 260.496 | 44.5063 | 1002 | 192 | 5705 |
| 5000 | 332.886 | 41.8998 | 1022 | 259 | 5700 |
| 10000 | 405.63 | 124.322 | 3035 | 325 | 5710 |
| 20000 | 510.213 | 57.042 | 2037 | 372 | 5678 |
| 30000 | 641.655 | 72.0836 | 2041 | 405 | 5700 |
| 40000 | 655.038 | 88.5922 | 3607 | 393 | 5710 |
| 50000 | 762.694 | 103.887 | 2192 | 411 | 5690 |
| 60000 | 842.812 | 113.765 | 2146 | 401 | 5690 |
| 70000 | 882.978 | 151.172 | 4086 | 409 | 5670 |
| 80000 | 977.006 | 171.798 | 5145 | 410 | 5695 |
| 90000 | 1060.05 | 223.663 | 9081 | 414 | 5680 |
| 100000 | 1074.74 | 242.613 | 6698 | 415 | 5690 |
| 150000 | 1342.27 | 355.98 | 4114 | 418 | 5690 |
| 200000 | 1357.76 | 425.761 | 4125 | 420 | 5680 |
| 250000 | 1617.6 | 534.443 | 4993 | 419 | 5670 |
| 300000 | 1886.05 | 839.897 | 12658 | 417 | 5690 |
| 350000 | 2130.9 | 1544.54 | 20697 | 425 | 5620 |

## 5.4 Three-Node Configuration (RHEL 4.4) Latency – with Process & Interrupt Binding

This shows the results with irqbalance disabled and Ethernet interrupts bound to CPU 0, RRCPD bound to CPU 0, Src_Dist and P2PS bound to CPU 1.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 265.086 | 47.1616 | 2010 | 194 | 5695 |
| 5000 | 330.004 | 53.7201 | 2016 | 259 | 5700 |
| 10000 | 396.441 | 67.5512 | 3030 | 325 | 5710 |
| 20000 | 519.296 | 123.114 | 6007 | 387 | 5690 |
| 30000 | 619.963 | 65.7104 | 2006 | 398 | 5690 |
| 40000 | 581.038 | 83.3479 | 2459 | 386 | 5670 |
| 50000 | 701.862 | 97.0837 | 3065 | 389 | 5700 |
| 60000 | 787.507 | 102.317 | 2090 | 396 | 5680 |
| 70000 | 839.223 | 134.604 | 2085 | 398 | 5650 |
| 80000 | 879.065 | 160.113 | 2158 | 394 | 5650 |
| 90000 | 915.426 | 188.861 | 4325 | 393 | 5640 |
| 100000 | 1006.47 | 195.363 | 2983 | 402 | 5640 |
| 150000 | 1219.46 | 344.083 | 4076 | 397 | 5620 |
| 200000 | 1351.87 | 346.131 | 4207 | 396 | 5510 |
| 250000 | 1646.74 | 484.91 | 5602 | 404 | 5630 |
| 300000 | 1886.51 | 1446.53 | 36820 | 407 | 5610 |
| 350000 | 2074.07 | 1079.98 | 20025 | 400 | 5650 |
| 400000 | 3386.49 | 3600.84 | 53303 | 409 | 4901 |

## 5.5 Comparison of 2-Node and 3-Node Configurations

Plotting the data from the last four tables, we can see in Figure 6 below that the 3-node configurations always outperformed the 2-node configurations because the cpu (running both Src_Dist and P2PS) became the bottleneck in the 2-node configuration.

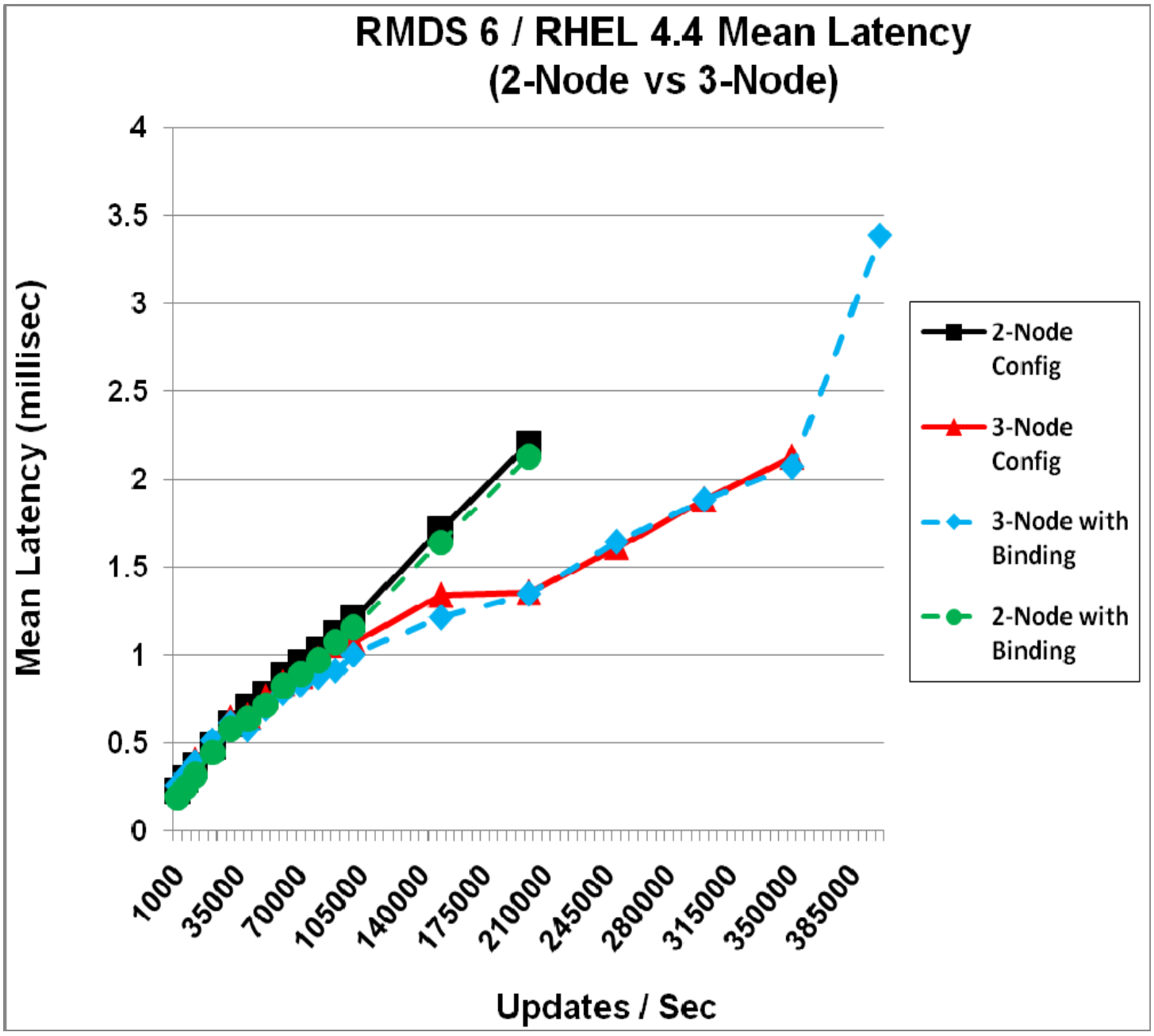


*Figure 6*

# 6. RHEL 4.4 versus RHEL 5.1 - Latency Test Results

## 6.1 RHEL 4.4 (2-Node Configuration) Latency

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 211.81 | 44.7495 | 1008 | 141 | 5670 |
| 5000 | 280.705 | 43.3033 | 1035 | 197 | 5688 |
| 10000 | 327.812 | 64.814 | 2043 | 253 | 5684 |
| 20000 | 454.867 | 50.9464 | 2005 | 327 | 5660 |
| 30000 | 557.348 | 48.9816 | 2053 | 384 | 5670 |
| 40000 | 585.486 | 58.9436 | 3060 | 354 | 5660 |
| 50000 | 670.418 | 69.541 | 2068 | 427 | 5710 |
| 60000 | 739.502 | 103.342 | 4051 | 379 | 5679 |
| 70000 | 800.903 | 101.526 | 2463 | 376 | 5640 |
| 80000 | 865.41 | 106.018 | 3015 | 426 | 5669 |
| 90000 | 856.357 | 132.657 | 4068 | 401 | 5660 |
| 100000 | 913.143 | 145.231 | 2086 | 372 | 5650 |
| 150000 | 1223.28 | 245.208 | 3584 | 343 | 5650 |
| 200000 | 1491.07 | 262.754 | 4925 | 339 | 5600 |
| 250000 | 1920.94 | 428.929 | 8146 | 354 | 5517 |

## 6.2 RHEL 4.4 (2-Node Configuration) Results – with Process & Interrupt Binding

This shows the results with irqbalance disabled and Ethernet interrupts bound to CPU 0, RRCPD bound to CPU 0, Src_Dist and P2PS bound to CPU 1.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 216.459 | 43.3781 | 1025 | 144 | 5670 |
| 5000 | 263.686 | 51.0862 | 2024 | 192 | 5722 |
| 10000 | 322.09 | 50.8939 | 2029 | 250 | 5720 |
| 20000 | 429.737 | 46.6239 | 2002 | 326 | 5686 |
| 30000 | 537.685 | 53.7293 | 2049 | 357 | 5670 |
| 40000 | 591.357 | 64.2492 | 2061 | 349 | 5699 |
| 50000 | 650.279 | 75.5374 | 2058 | 360 | 5694 |
| 60000 | 749.243 | 85.3279 | 2063 | 413 | 5700 |
| 70000 | 757.033 | 90.5869 | 2322 | 389 | 5692 |
| 80000 | 769.549 | 114.103 | 1868 | 345 | 5706 |
| 90000 | 844.348 | 145.532 | 2883 | 386 | 5680 |
| 100000 | 895.712 | 163.387 | 3006 | 369 | 5690 |
| 150000 | 1239.24 | 231.947 | 3929 | 362 | 5656 |
| 200000 | 1466.64 | 250.685 | 4161 | 376 | 5670 |
| 250000 | 1858.78 | 379.811 | 6114 | 337 | 5670 |
| 300000 | 2489.42 | 934.663 | 9407 | 333 | 2430 |

## 6.3 RHEL 5.1 (2-Node Configuration) Results

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 192.911 | 17.3959 | 415 | 158 | 5697 |
| 5000 | 248.313 | 18.6694 | 484 | 210 | 5697 |
| 10000 | 319.39 | 20.299 | 549 | 274 | 5718 |
| 20000 | 427.02 | 35.0676 | 670 | 327 | 5705 |
| 30000 | 499.835 | 51.1025 | 954 | 359 | 5692 |
| 40000 | 564.888 | 56.5912 | 1053 | 392 | 5692 |
| 50000 | 631.674 | 66.233 | 1241 | 415 | 5703 |
| 60000 | 746.547 | 91.0913 | 1440 | 435 | 5668 |
| 70000 | 795.799 | 95.585 | 1706 | 424 | 5647 |
| 80000 | 892.508 | 105.591 | 1877 | 422 | 5680 |
| 90000 | 985.734 | 124.735 | 2399 | 409 | 5670 |
| 100000 | 1059.52 | 157.573 | 2910 | 495 | 5671 |
| 150000 | 1335.41 | 316.368 | 4141 | 390 | 5644 |
| 200000 | 1627.07 | 713.797 | 9506 | 417 | 2738 |

## 6.4 RHEL 5.1 (2-Node Configuration) Results – with Process & Interrupt Binding

This shows the results with irqbalance disabled and Ethernet interrupts bound to CPU 0, RRCPD bound to CPU 0, Src_Dist and P2PS bound to CPU 1.

| Update Rate (per sec) | Mean (usec) | Std Deviation (usec) | Max Latency (usec) | Min Latency (usec) | # Samples |
|---|---|---|---|---|---|
| 1000 | 194.537 | 18.8991 | 428 | 153 | 5684 |
| 5000 | 249.091 | 18.4876 | 469 | 208 | 5679 |
| 10000 | 309.731 | 20.4177 | 557 | 268 | 5686 |
| 20000 | 420.08 | 32.7705 | 823 | 321 | 5652 |
| 30000 | 492.092 | 56.1817 | 887 | 345 | 5680 |
| 40000 | 529.98 | 74.6715 | 1080 | 370 | 5690 |
| 50000 | 614.742 | 83.904 | 1572 | 380 | 5690 |
| 60000 | 715.36 | 107.054 | 1473 | 380 | 5690 |
| 70000 | 764.807 | 116.501 | 1706 | 412 | 5690 |
| 80000 | 865.693 | 124.69 | 1774 | 399 | 5660 |
| 90000 | 953.288 | 147.279 | 2017 | 459 | 5670 |
| 100000 | 1035.64 | 167.995 | 2860 | 474 | 5650 |
| 150000 | 1286.48 | 306.798 | 3564 | 365 | 5670 |
| 200000 | 1464.03 | 334.487 | 2915 | 375 | 5620 |

## 6.5 Comparison of RHEL 4.4 and RHEL 5.1

Plotting the data from the last four tables, we can see in Figure 7 below that there is no statistically significant performance regression when going from RMDS 6 / RHEL 4.4 to RMDS 6 / RHEL 5.1. And the throughput is limited by the cpu (running both Src_Dist and P2PS) becoming the bottleneck in this 2-node configuration

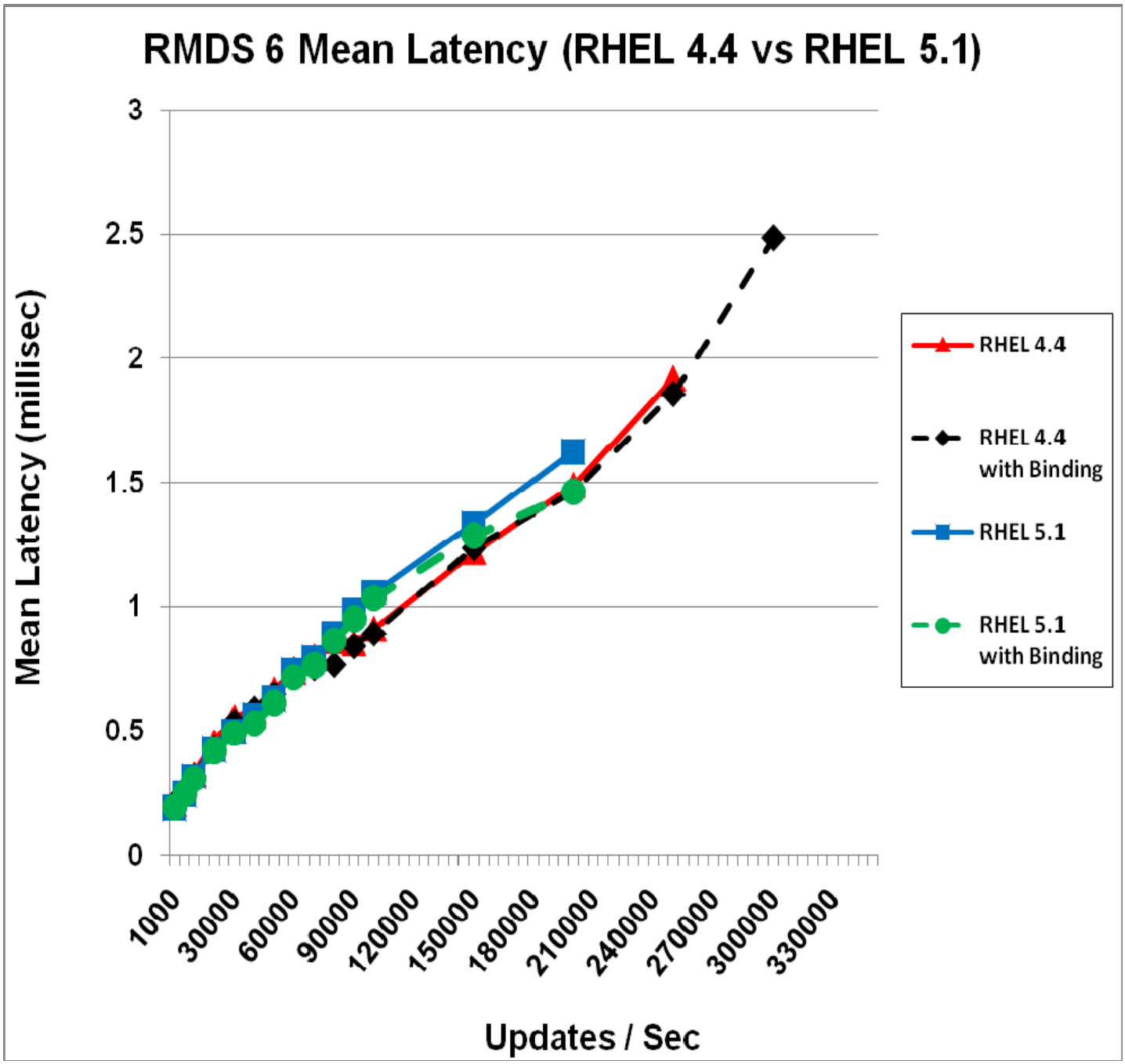


*Figure 7*

# 7. Summary & Conclusions

In the financial services industry, automated trading is a race – firms develop sophisticated strategies and attempt to beat the competition by analyzing and acting on market information faster than other traders. The quicker the systems, the better the financial results. For these financial services firms, one of the keys to maximizing portfolio performance is reducing data latency, or the time it takes for the market information to get to the trading applications. RMDS (Reuters Market Data System) is a high throughput, low latency platform for market data applications. This reference architecture is Phase I of a study of the performance, scalability, tuning, and optimization of RMDS running on RHEL (Red Hat Enterprise Linux).

All the performance tests described herein used the Reuters RMDS performance testbed. Typically these tests are run using configurations of 3 or more nodes – one node runs the load driver, the second node runs the RMDS infrastructure component that publishes market data (= source distributor), and a third node runs the RMDS infrastructure component that subscribes to market data (= Point-to-Point server = P2PS). Due to a shortage of hardware and being pressed for time, it was decided to run the latter two nodes on a single physical node with a loop-back ethernet connection to simulate the network hop. However, before deciding on using the 2-node configuration, a series of tests were run to position the relative performance of the 2-node and 3-node test configurations. As documented in this report, the 3-node configurations always outperformed the 2-node configurations because the cpu became the bottleneck in the 2-node configuration.

- With these results we are confident that real-world 3-node configurations would perform better than any 2-node configurations tested in the lab. With this assurance, we proceeded to use the 2-node configuration for the RHEL 4 versus RHEL 5 comparison.

Today, many of our existing customers use RMDS on RHEL 4. For many of these customers who are considering moving from RMDS 5 / RHEL 4 to RMDS 6 / RHEL 5, we wanted to ensure continued leadership performance. One of the primary goals of this study was to ensure that there was no performance regression when moving RMDS from RHEL 4 to RHEL 5. The results clearly show:

- In the configurations tested, there is no statistically significant performance regression when going from RMDS 6 / RHEL 4.4 to RMDS 6 / RHEL 5.1
- Process and Interrupt binding to cpu cores did not make any appreciable difference in performance
- Setting TSO to “on” and “off” did not make any appreciable difference in performance

The configurations employed for the tests in this report used 2-core Intel® Xeon® processors and a limited number of 1GigE NICs (Network Interface Cards). In follow-on testing, the goal is to use multi-core servers with a larger number of NICs. This will enable us to more fully explore the impact of binding instances of RMDS infrastructure processes and NIC interrupts to cpu cores in different topologies.

# 8. Next Steps (Tuning & Optimization)

There are several areas of RMDS/RHEL performance measurement, optimization and tuning that will be investigated in the next phase of this work.

## 8.1 Multi-core, Multi-NIC Configurations with Interrupt and Process Binding

The configurations employed for the tests in this report used 2-core Intel® Xeon® processors and a limited number of 1GigE NICs. These restrictions limited the insights that could have been gained by having multiple instances of RMDS infrastructure components - Src_Dist, P2PS, and RRCPD - and the binding of these processes and NIC interrupts to cpu cores in different topologies. In follow-on testing the goal is to use multi-core servers with a larger number of NICs. This will enable us to more fully explore the impact of binding processes and interrupts to cpu cores.

## 8.2 Update Throughput

The focus of this study was on measuring end-to-end latency. One of the next steps is to determine the maximum throughput that can be sustained by the system. The maximum throughput for the Src_Dist and P2PS are determined separately.

### 8.2.1 P2PS – Update Throughput

To avoid the Source Distributor becoming the limiting factor when testing the P2PS update throughput, in some cases two source servers (data simulator/Source Distributor) are configured to provide updates to the P2PS. Similarly in some cases multiple sink applications are connected to the P2PS where it is necessary in order to create sufficient load.

### 8.2.2 Source Distributor – Update Throughput

Reuters in-house testing has shown that the update throughput of a Source Distributor is more than that of a P2PS. So, to perform Source Distributor non-caching tests, you would need to run two or more P2PS nodes and split the load.

## 8.3 RMDS Infrastructure Topologies

### 8.3.1 Traditional Topology

In the traditional topology, each machine runs exactly one instance of RRCP and one instance of the P2PS or Src_Dist. All RRCP daemons communicate over the same network

address, and each machine has one network interface card (NIC) to the RMDS backbone. All RMDS "services" flow across this single network address and NIC, using either UDP broadcast or IP multicast. (RMDS "services" are the mechanism by which RMDS enables system administrators to manage content namespaces. Each service typically corresponds to a source of data, such as the Reuters Datafeed, an internal rate system, etc. A service is also a load-balancing/fault-tolerance domain.)

## 8.3.2 Stacked Topologies

The traditional topology has some limitations. While the RRCP daemon is multi-threaded and scales across multiple processors or cores, most P2PS and Src_Dist processing happens in a single thread, which limits performance in systems with many thread processors. The challenge is therefore to configure multiple instances of RMDS processes (= Src_Dist, P2PS, RRCPD) in so called "stacked" topologies to take advantage of multiple CPU cores in order to deliver better performance on multi-core architectures.

### 8.3.2.1 P2PS Multiplex Topology

In a multiplex topology multiple P2PS instances consume from a common RRCP daemon. This topology effectively co-locates several P2PS instances that would otherwise run on separate servers. In so doing, it utilizes the power of multiple multi-core processors and achieves economies of scale by leveraging a common RRCP daemon.

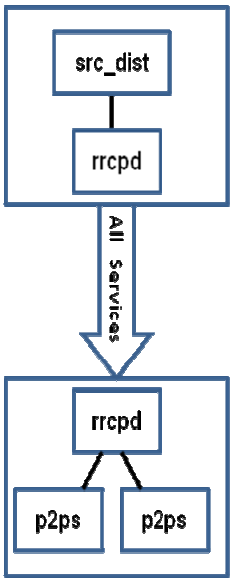


*Figure 8: P2PS Multiplex Topology*

### 8.3.2.2 P2PS Multipath Topology

Multiplexing more than two P2PS instances creates undesirable performance side effects. Another way to support multiple P2PS instances on a server is to run multiple RRCP daemons each bound to its own NIC and P2PS instance. A unique set of RMDS “services” is assigned to each network address = RRCP daemon. This creates a “multipath” topology, in

which the server supports parallel, independent data paths. In this topology, each P2PS instance handles a unique set of RMDS services, rather than the traditional model in which every P2PS handles all services.

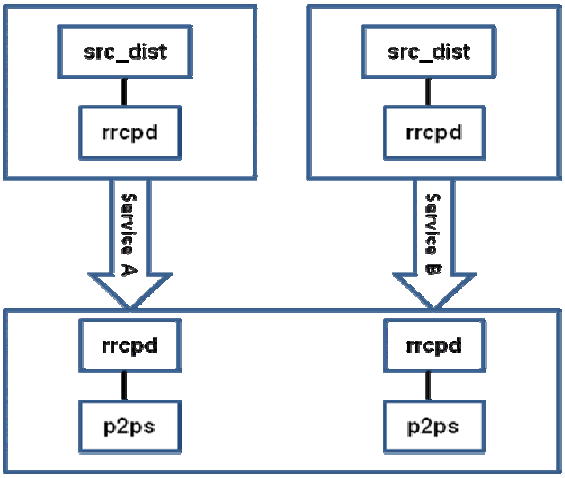


*Figure 9: P2PS Multipath Topology*

### 8.3.2.3 P2PS Multipath/Multiplex Topology

A preferred topology combines mutliplexing and multipathing RMDS services are split across multiple RRCP daemons. Each RRCP daemon binds to its own NIC and supports a pair of P2PS instances.

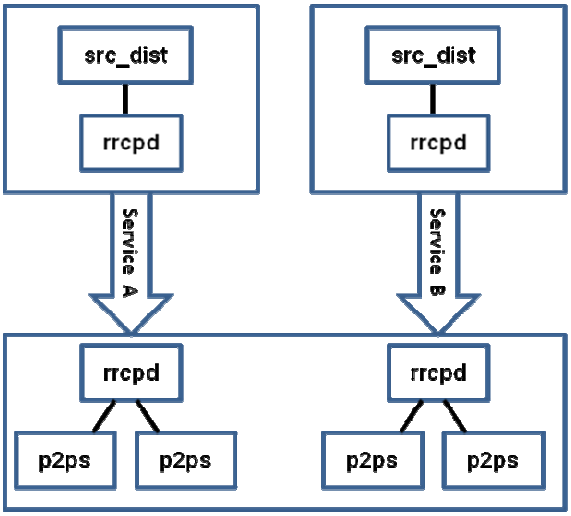


*Figure 10: P2PS Multipath/Multiplex Topology*

### 8.3.2.4 P2PS Fanout

The fanout capability of a P2PS machine is oriented toward environments in which many users are connected to the P2PS.

The P2PS "Producer 50/50" test is an extreme test of the fanout capability of a P2PS machine. It is oriented toward environments in which many users are connected to the P2PS

and users have a high degree of commonality in their watchlists (meaning that for most of the updates that the P2PS receives from the backbone, it must forward each update to many users).

To maximize fanout performance, a multiplex topology is chosen, in which multiple P2PS instances consume from a common RRCP daemon.

### 8.3.2.5 Src_Dist Multipath/Multiplex Topology

The performance challenge of a Src_Dist is throughput without fanout.

Multiplex topologies, in which multiple Src_Dist instances publish through a single RRCPD, improve throughput over standard topologies.

Just as in the P2PS case, a preferred topology for Src_Dist combines mutliplexing and multipathing. RMDS services are split across multiple RRCP daemons. Each RRCP daemon binds to its own NIC and supports two Src_Dist instances.

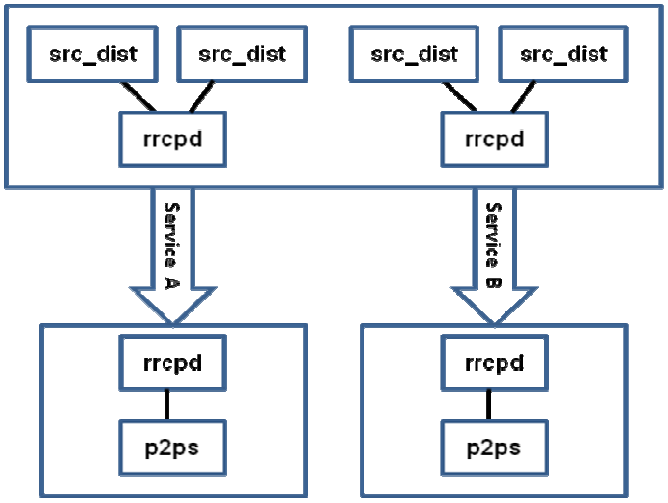


*Figure 11: Src_Dist Multipath/Multiplex Topology*

### 8.3.2.6 Stacked Topology for Latency Testing

To test latency through just two servers (one for the Src_Dist, one for the P2PS), STAC uses Src_Dist multipath/multiplex with the P2PS multipath/multiplex.

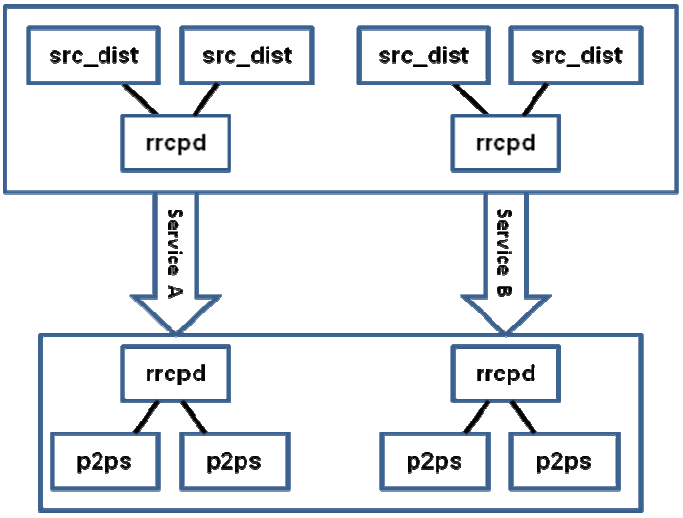


*Figure 12: Stacked Topology for Latency Testing*

## 8.4 10 GigE and Infiniband

As seen from recent STAC reports, faster networking technology like 10 GigE and Infiniband can enable significantly higher throughput and lower latencies. We will study the imact of these technologies in the next phase.

## 8.5 RealTime Kernel

Finally, we will tune RMDS running on the RealTime Kernel to further improve end-to-end latencies.

# 9. References


1. Reuters - RMDS Core Infrastructure 6.0 – Demo Tools, Issue 0.4, 5 September 2006.

2. Reuters – Reuters Market Data System 6.0 – Performance Test Procedure and Results, Issue 1.1, 8 June 2006.

3. Reuters – RMDS 6 Performance Testing, 8 June 2006.

4. STAC Report – BNT 10 GigE Switch and Chelsio NIC with RMDS 6 / RHEL 5.1 on Intel®-based IBM BladeCenter® Server, Issue 1.0, 18-March-2008.

5. STAC Report – RMDS 6 / RHEL 5.1 on a 45nm Intel® Xeon® Harpertown Processor, Issue 1.0, 26-February-2008.

6. STAC Report – RMDS 6 / RHEL 4.5 on a 16-core Intel® 'Caneland' Server, Issue 1.0, 14-September-2007.

7. STAC Report – RMDS 6 / RHEL 4.4 with Voltaire Infiniband and HP c-Class BladeSystem, Issue 1.0, 19-June-2007

# 10. Appendix I: Determining Steady State

The next four figures in this appendix clearly show that there is an initial period of between 35-50 seconds before the statistics enter a steady state. So, when processing the data the first 300-600 samples were discarded. This is because only after 35-50 seconds from launch does the test start producing samples with the required update rate.

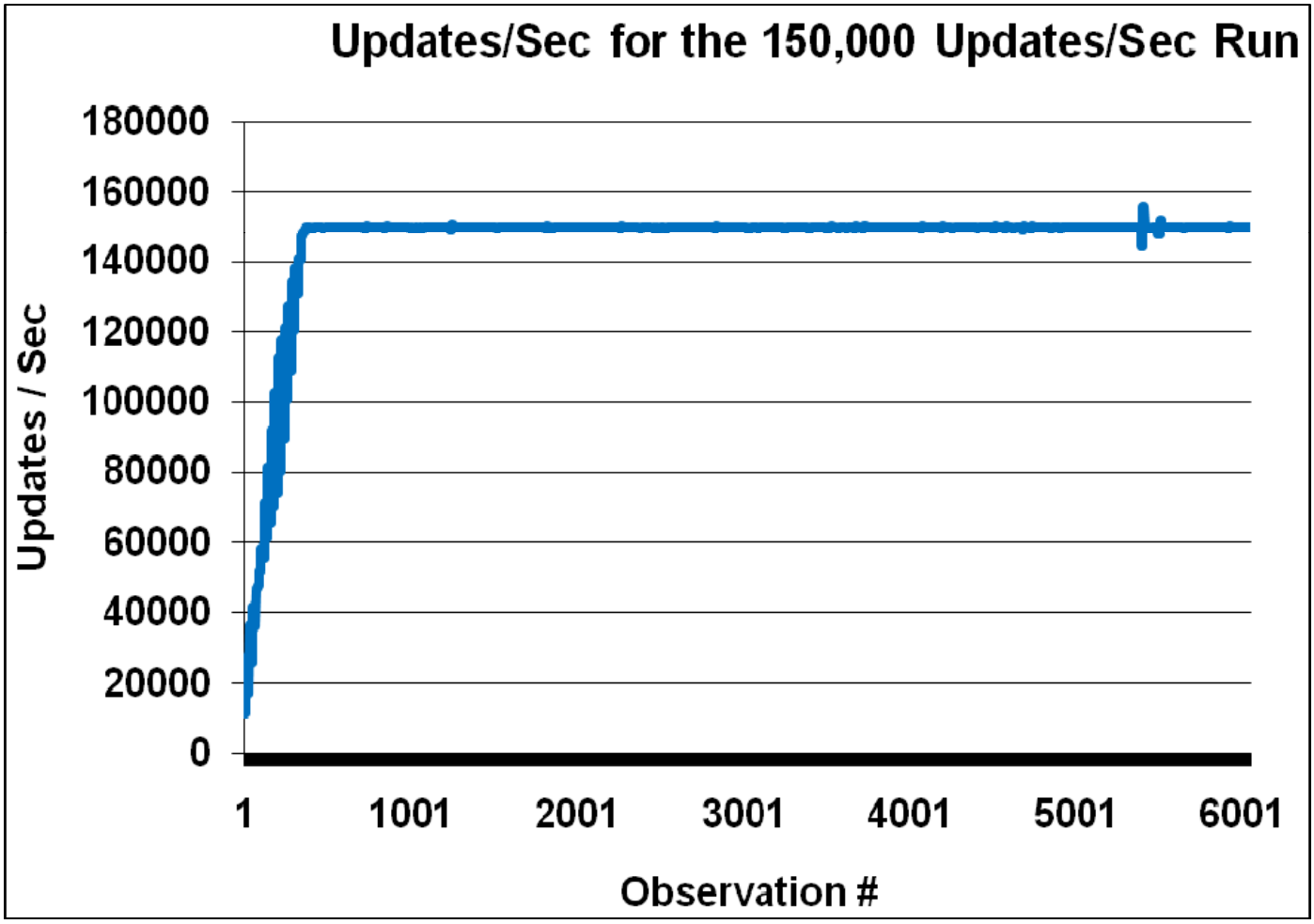


*Figure 13*

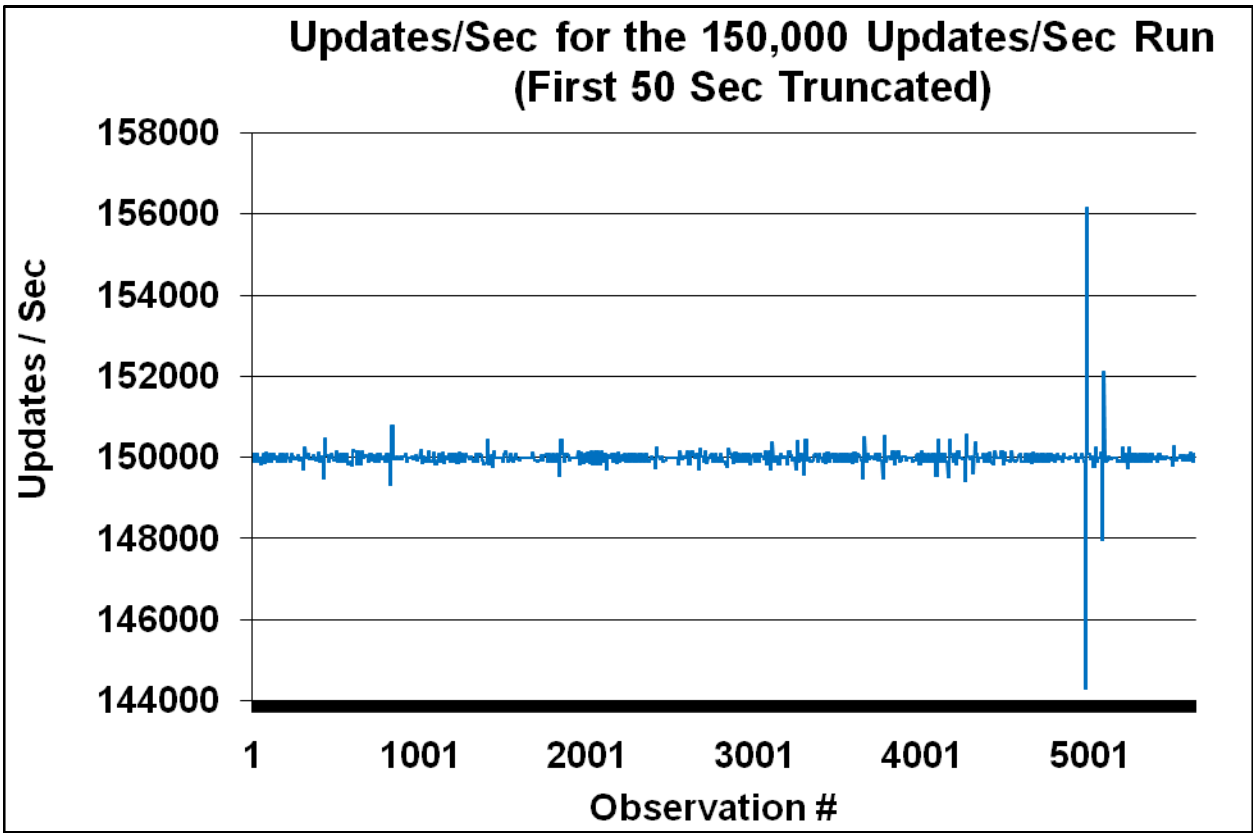
Updates/Sec for the 150,000 Updates/Sec Run
(First 50 Sec Truncated)
Updates / Sec
158000
156000
154000
152000
150000
148000
146000
144000
1
1001
2001
3001
4001
5001
Observation #

*Figure 14*

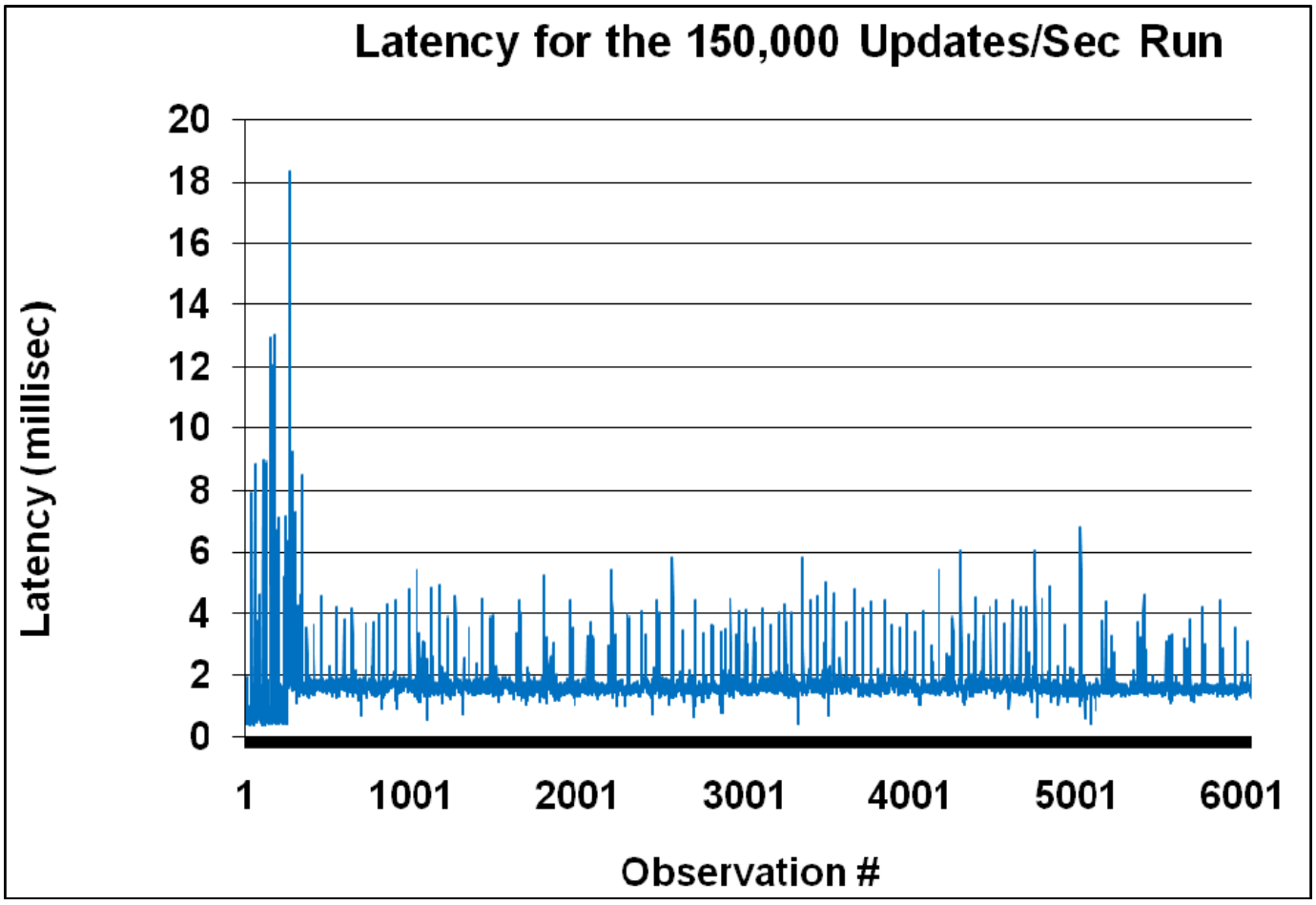
Latency for the 150,000 Updates/Sec Run
Latency (millisec)
20
18
16
14
12
10
8
6
4
2
0
1
1001
2001
3001
4001
5001
6001
Observation #

*Figure 15*

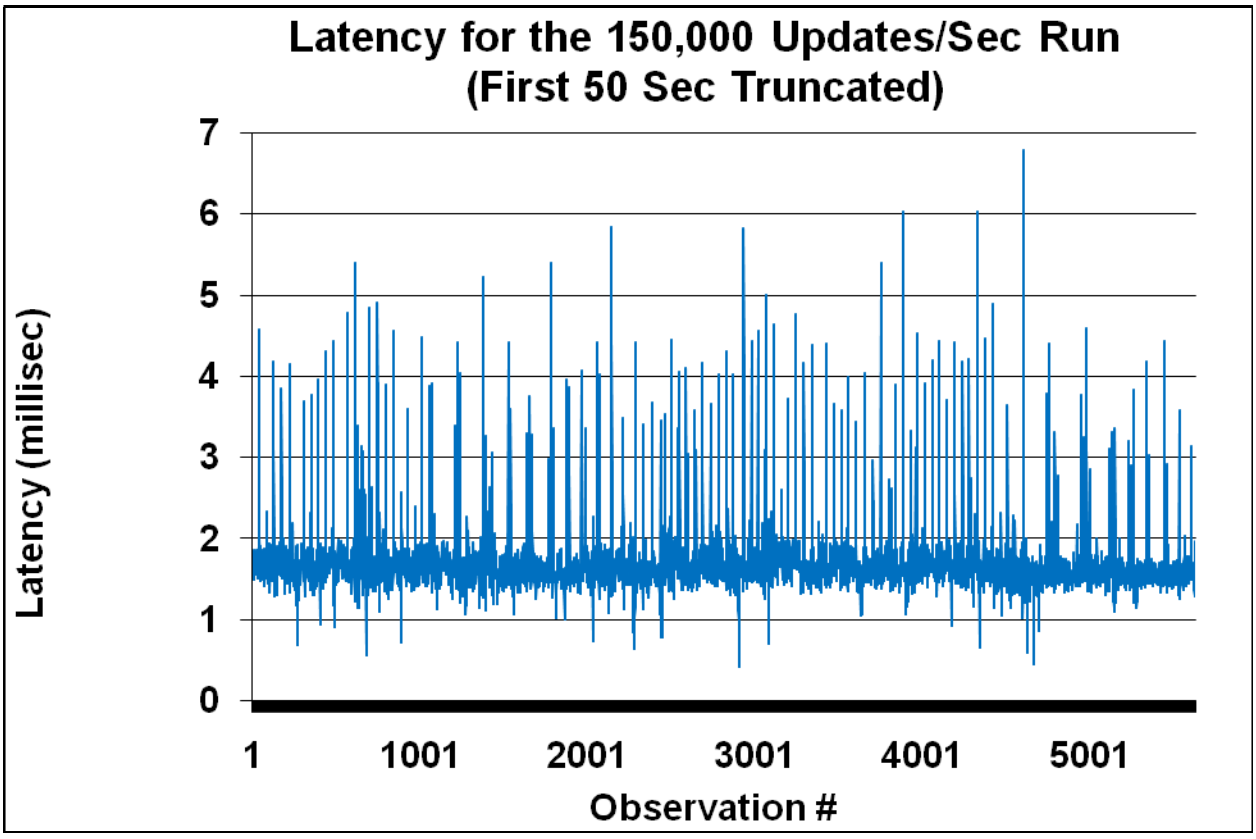
Latency for the 150,000 Updates/Sec Run
(First 50 Sec Truncated)
Latency (millisec)
7
6
5
4
3
2
1
0
1
1001
2001
3001
4001
5001
Observation #

*Figure 16*

# 11. Appendix II: Recent RMDS/RHEL Results from STAC Research

The Securities Technology Analysis Center (STAC®) is a provider of performance measurement services, tools and research to the securities industry. A summary of some recent STAC reports about RMDS/RHEL throughput and latency are included in the appendix. STAC reports are available at: http://www.STACresearch.com.

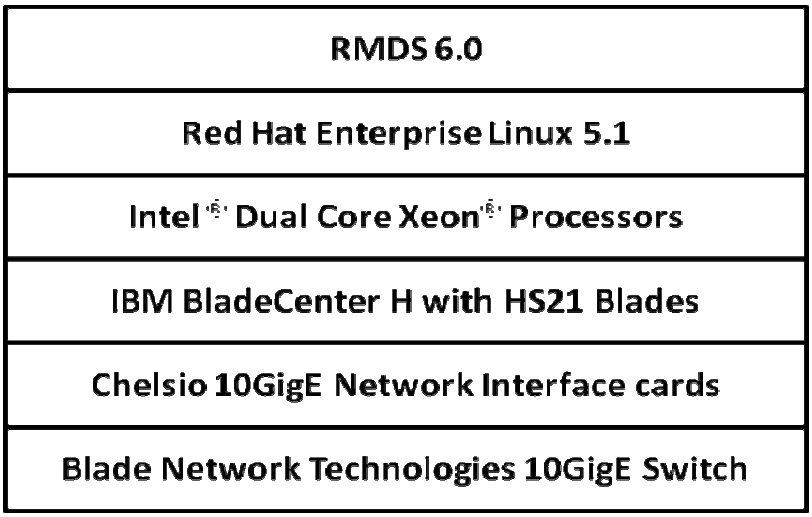


***Figure 17***

STAC, on 18-March-2008, reported that RMDS 6 / RHEL 5.1 running on IBM BladeCenter® H with HS21 Blades and using Blade Network Technologies (BNT) 10 Gigabit Ethernet Switch and Chelsio Communication's 10 Gigabit Ethernet Network Interface Card (NIC) achieved:

1. Lowest mean latency ever reported with RMDS
   - Less than 0.9 milliseconds of end-to-end infrastructure latency at up to 600,000 updates per second in the low-latency configuration of RMDS
2. Lowest standard deviation of latency ever reported with RMDS
   - Less than 0.5 milliseconds at rates up to 600,000 updates per second.
3. Very high output rate in the — “Producer 50/50” fanout test of a stacked P2PS
   - 5.8 million updates per second
   - 30% of this due to the TCP/IP Offload Engine (TOE) in the Chelsio NIC Blade Network Technologies 10GigE Switch

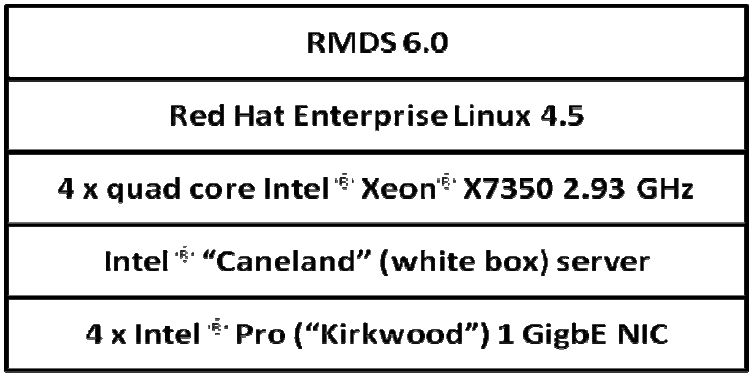


***Figure 18***

STAC, on 14-September-2007, reported that RMDS 6 / RHEL 4.5 running on 4 x quad core Intel® Xeon® with 4 x Intel® Pro 1 GigE NICs achieved:

1. Highest Source Distributor throughput to date on a single 4-socket or 2-socket server
    - 2.8 million updates per second
2. Highest Point-to-Point Server throughput to date on a single 4-socket or 2-socket server
    - 2.2 million updates per second through a single Point-to-Point Server