



Reuters Market Data System

RMDS 6.0

Sun Fire X4150 - Intel Xeon 3.16GHz Quad Core
(1GbE) Ethernet Infrastructure

DISCLAIMER: The test results and recommendations contained in this report are made available for informational purposes only. By issuing this report, Sun Microsystems, Inc. does not guarantee similar performance results. This report is provided for your internal use only and may not be redistributed or published in any form without the prior written consent of Sun Microsystems, Inc. All information contained herein is provided on an "AS-IS" BASIS WITHOUT WARRANTY OF ANY KIND. Obtaining repeatable, measurable performance results requires a controlled environment with specific hardware, software, network, and configuration in an isolated system. Adjusting any single element may yield different results. Additionally, test results at the component level may not be indicative of system level performance, or vice versa. Sun Microsystems, Inc. is pleased to provide this report for our customers. We appreciate that each organization has unique requirements, and therefore may find this information insufficient for its needs. Customers wishing to obtain custom analysis for their systems are encouraged to contact their local Sun Microsystems, Inc. representative.

Issue 3.1

Date of Issue: June 20, 2008

Contents

1 General Information	4
1.1 Objective.....	4
Results Summary	4
1.2 Testing Methodology.....	4
1.3 Software Versions.....	5
1.3.1RMDS.....	5
1.3.2RMDS Test Tool.....	5
1.3.3Operating Systems.....	5
1.4 Hardware.....	5
2 Preparation for Performance Test.....	5
2.1 Network.....	5
2.2 Hardware.....	5
2.3 Operating System Configuration.....	5
2.3.1Solaris Kernel Parameters.....	6
2.3.2TCP and UDP Buffers.....	6
2.3.3Intel NIC (e1000g) Driver Settings.....	6
2.4 RMDS Configuration.....	7
2.5 Miscellaneous Notes.....	7
3 Detailed Results.....	10
3.1 RSSL/RWF Update Throughput.....	10
3.1.1Standalone Source Distributor.....	10
3.1.2P2PS/LAN.....	10
3.2 Update Throughput via P2PS/POP.....	11
3.2.1RSSL/RWF.....	11
3.3 End-to-End RSSL/RWF Latency.....	12
3.3.1RRCP Backbone Results.....	12
3.3.2Rendezvous Backbone Results.....	13

© Sun Microsystems, Inc. 2008. All Rights Reserved.

Sun Microsystems, Inc. , by publishing this document, does not guarantee that any information contained herein is and will remain accurate or that use of the information will ensure correct and faultless operation of the relevant service or equipment. Sun Microsystems, Inc., its agents and employees, shall not be held liable to or through any user for any loss or damage whatsoever resulting from reliance on the information contained herein.

This document contains information proprietary to Sun Microsystems, Inc. and may not be reproduced, disclosed, or used in whole or part without the express written permission of Sun Microsystems, Inc.

Any Software, including but not limited to, the code, screen, structure, sequence, and organization thereof, and Documentation are protected by national copyright laws and international treaty provisions. This manual is subject to U.S. and other national export regulations.

1 General Information

1.1 Objective

The objective of this document is to report the performance test results for RMDS 6.0.3 on the Sun Microsystems x4150 system with Intel Xeon 3.16GHz quad core processors. The test procedures are described in Reuters RMDS 6.0 Performance Test Procedures and Results document.

The x4150 is a x86 architecture based on the Intel Xeon processor chip. The systems used in these tests had 2 processor chips with 4 cores per chip to execute applications in parallel.

The goal of these tests is to measure throughput and latency through RMDS 6.0 infrastructure components, specifically the Point-to-Pont Server (P2PS) and Source Distributor. The tests are grouped into three categories:

- Update throughput using RSSL/RWF data (see 3.1)
- Update throughput via P2PS/POP (see 3.2)
- End-to-end RSSL/RWF latency using embedded timestamp (see 3.3)

Results Summary

RMDS Components Configuration	RMDS Throughput & End-to-End Latency Using Ethernet Infrastructure
Maximum Throughput at <=1ms RMDS end-to-end Latency	500,000 updates/sec
P2PS-LAN, RRCP, Producer 50/50 Fanout (Cache Disabled)	3,050,200 updates/sec
P2PS-LAN, RVD, Producer 50/50 Fanout (Cache Disabled)	3,030,000 updates/sec
P2PS-POP, Producer 50/50 Fanout (Cache Enabled)	2,646,200 updates/sec
Src_dist, RRCP (Cache Disabled)	1,010,000 updates/sec
Src_dist, RVD (Cache Disabled)	970,000 updates/sec
P2PS-LAN, RRCP No Fanout (Cache Disabled)	876,000 updates/sec
P2PS-LAN, RVD No Fanout (Cache Disabled)	764,000 updates/sec
P2PS-POP, No Fanout (Cache Enabled)	460,000 updates/sec

1.2 Testing Methodology

For throughput testing, the **sink_driven_src** utility was used to generate update traffic, and the **rdmstestclient** utility was used to consume the updates. Level 1 data was used, with a Reuters Wire Format (RWF) update size of 74 bytes. Tests with no fanout of updates used a 100,000 item watchlist. The infrastructure was tuned for maximum throughput, and the update rate was increased until the CPU limit was reached with no errors reported. All tests were run at this final rate for at least 20 minutes. Where needed, and as noted, multiple Source Distributors or multiple P2PSs were used to create the load necessary to measure the component under test.

The embedded timestamp approach was used to calculate end-to-end latency for Level 1 (Quotes and Trades) data. RMDS 6.0 end-to-end update latency is measured by using **sink_driven_src** as the publisher and **rdmstestclient** as the subscriber. In the embedded timestamp approach, the publisher embeds timestamps into selected updates which the subscriber uses for latency calculations. In this scenario, the publisher and subscriber must be running on the same node for accurate timestamps.

1.3 Software Versions

1.3.1 RMDS

src_dist ver. mdh6.0.3.L1
p2ps ver. p2ps6.0.3.L1
rrcp as included in mdh6.0.3.L1 and p2ps6.0.3.L1
rvd 8.1

1.3.2 RMDS Test Tool

sink_driven_src (from MDH load above)
rmdstestclient (from P2PS load above)

1.3.3 Operating Systems

- **Solaris 10 8/07** (update 4)
- **Patches 125370-06 and 127112-11**

1.4 Hardware

- **Sun Fire X4150** - 2 X 3.16GHz quad-core Intel Xeon processors (total 8 cores)
- **Cisco Catalyst 4948** 1 Gbps Network Switch

2 Preparation for Performance Test

2.1 Network

All the performance tests were run with the machines connected to private networks via a 1 Gbps switch (Cisco Catalyst 4948). All the network cards and switch ports were set to Auto Negotiate.

2.2 Hardware

Three x4150 systems were available for these tests. For all tests, two x4150 systems were always assigned to the *src_dist* and *p2ps* RMDS components. The third x4150 system was used as a second *p2ps* system for the *src_dist* throughput tests, for the *p2ps/POP* in the POP tests, and as the *sink_driven_src/rmdstestclient* system for the latency tests. In all throughput tests, additional systems were used for the *sink_driven_src* and *rmdstestclient* tools..

2.3 Operating System Configuration

Solaris 10 8/07 (update 4) was installed on all systems. Two additional patches which are required to run with the 3.16GHz processors were added, 125370-06 and 127112-11. The e1000g network driver was used for all testing.

The network parameters set via *ndd* commands may be set from the command line or via a Service Management Facility (SMF) manifest and method file (*site/ndd*) available from:
<http://opensolaris.org/os/community/smf/manifests/>

For Solaris 10 update 3 and beyond, the *udp_smallest_anon_port* parameter should be set from the network-anon manifest and method files that are also available from the above [URL](#).

Further detail on tuning Solaris 10 and RMDS can be found at <http://www.sun.com/reuters> .

The RMDS 6.0.3 software used in this testing is the standard Reuters distribution for Sun x86 Solaris. As such it was compiled with optimization specific to the AMD Opteron processor, not to the Intel Xeon. Some performance gains may be realizable with a custom compile for the Intel architecture.

<i>rmids.cnf.template</i>	Configuration file	<i>\$RMDS_SW/config</i> <i>on mdh and p2ps/LAN systems</i>
<i>pop.cnf.template</i>	Configuration file for P2PS/POP tests	<i>\$RMDS_SW/config</i> <i>on p2ps/POP system only</i>

Any non-required changes (i.e., IP addresses, hostnames, etc). are noted below:

Config File	Parameter	Value
<i>rmids.cnf</i>	<i>*RRCP*udpRecvBufSize</i>	16384
<i>rmids.cnf</i>	<i>*RRCP*udpSendBufSize</i>	16384
<i>rmids.cnf</i>	<i>*RDFD*readBias</i>	60 (add this entry)
<i>rmids.cnf</i>	<i>*p2ps*maxOutputBuffers</i>	1000
<i>rmids.cnf</i>	<i>*p2ps*guaranteedOutputBuffers</i>	800

Values modified for **network latency** testing (per comment in the template file) are noted below:

Config File	Parameter	Value
<i>rmids.cnf</i>	<i>*IDN_RDF*rrmpFlushInterval</i>	0
<i>rmids.cnf</i>	<i>*RDFD*rrmpFlushInterval</i>	0
<i>rmids.cnf</i>	<i>*p2ps*tcpNoDelay</i>	True
<i>rmids.cnf</i>	<i>*p2ps*timedWrites</i>	False
<i>rmids.cnf</i>	<i>*p2ps*flushInterval</i>	0

2.5 Miscellaneous Notes

Any other significant deviations from the standard test procedures, or clarifications, are noted below (such as number/type of machines used, CPU binding policy, etc.):

For RMDS 6.0, the p2ps application is a multithreaded process where one thread does a substantial amount of work (this thread is referred to as the “high runner” below) with several other threads supporting it. When running with rrcpd in a LAN configuration, thread number 4 (of 5) is the high runner. When running with rvd in a LAN configuration, thread 7 (of 8) is the high runner. When running in a POP configuration using RSSL/RWF, thread 4 (of 4) is the high runner. Bindings of threads to specific processor cores were handled via the use of processor sets and explicit pbind commands. Typically, one strand was used for the high runner, for p2ps a second for the other p2ps threads, and in the LAN configurations one or two others were used for the transport daemon (rrcpd or rvd). There are test cases where use of more strands may be desirable, and it is expected that continued development of the p2ps application will also require changes to this strategy.

Test	Deviation	Comments
All	Multicast vs Broadcast	The standard <i>rmids.cnf.template</i> file does not enable multicast for RRCP, but does so for RV. RRCP tests run here with multicast are marked in the comments field. It should be assumed that all RV tests were run with multicast as opposed to broadcast for UDP networking.
src_dist	CPU binding and interrupts disabling and CPU scheduling class assignment	Except where noted, <i>src_dist</i> was bound to CPU (core) 2 via the use of a processor set. The respective transport daemon (rrcpd or rvd) was bound to CPUs (cores) 3 via another processor set to take advantage of its multithreaded capability. Interrupts were disabled on any

		CPU (core) that was running an application thread.
src_dist	p2ps configuration	For the src_dist cache disabled tests (with both rrcpd and rvd), two x4150 systems were configured to run p2ps processes. This was required as a single p2ps process does not have the performance to match that of the src_dist process. In this case, the watch list used by each (of two) rmdstestclient invocation was split with one instance using one half, the other using the second half per the RMDS Test Guide. For the cache enabled tests, a single p2ps was sufficient for the test.
p2ps (LAN/POP)	CPU binding and interrupts disabling and CPU scheduling class assignment	Except where noted, p2ps was bound to CPUs (cores) 4 (high runner thread) and 3 (all other threads) via the use of processor sets. The respective transport daemon (rrcpd or rvd) was bound to CPUs (cores) 2 and 3 via an additional processor set to take advantage of its multithreaded capability. Interrupts were disabled on any CPU core that was running an application thread. CPUs 6 and 7 were forced offline to limit cache contention.
p2ps fanout (LAN/POP)	rmdstestclient configuration	Six instances of rmdstestclient were used in each of the 3 producer 50/50 (fanout) tests. Two instances were run on each of 3 testbed systems. Three distinct networks, one to each of the 3 systems, were configured. A two network configuration was inadequate at least for all fanout tests as the message rate on each network exceeded the capacity of a 1Gb Ethernet LAN at the peak rate quoted in the tables below.
All	Solaris scheduler	Extensive testing has shown that for Solaris 10 6/01 (update 2) and later, p2ps and its respective transport daemon (at minimum) should be run under the FX scheduling class, not under the default timesharing class. For this testing, all RMDS components were run in the FX class with priorities explicitly set. The priocntl command is used to set the scheduling class.

3 Detailed Results

3.1 RSSL/RWF Update Throughput

- All the throughput numbers quoted here are for Level 1 data.
- The data file used in these tests has 1 update, with an update (data, not including header) size of 74 bytes in RWF.
- All of the tests with no fan-out used 100,000 item watchlist.
- ***sink_driven_src*** and ***rmdstestclient*** were used as the publisher and consumer of data.
- Where only a single number is shown, that is both the input and output rate values of the test.

3.1.1 Standalone Source Distributor

Configuration Option	Transport	Max Throughput	Comments
Cache Disabled	RRCP	1,010,000	Two systems were used for p2ps in the test bed. Multicast was explicitly enabled for this test.
Cache Enabled	RRCP	418,000	
Cache Disabled	Rendezvous	970,000	Two systems were used for p2ps in the test bed.
Cache Enabled	Rendezvous	400,000	

3.1.2 P2PS/LAN

Configuration Option	Mounts : Commonality	Transport	Max Throughput	Comments
Cache Disabled	No fan-out	RRCP	876,000	
Cache Enabled	No fan-out	RRCP	445,000	
Cache Disabled	100 mounts; Producer 50/50	RRCP	60,400 input 3,050,200 output	Six rmdstestclient instantiations were used with 3 VLANs. Multicast was explicitly enabled for this test.
Cache Disabled	No fan-out	Rendezvous	764,000	
Cache Enabled	No fan-out	Rendezvous	394,000	
Cache Disabled	100 mounts; Producer 50/50	Rendezvous	60,000 input 3,030,000 output	Six rmdstestclient instantiations were used with 3 VLANs.

3.2 Update Throughput via P2PS/POP

These tests were performed using a P2PS/POP with data provided by an upstream P2PS/LAN, with an RRCP transport.

3.2.1 RSSL/RWF

Configuration Option	Mounts : Commonality	Max Throughput	Comments
Cache Enabled	No fan-out	460,000	
Cache Enabled	100 mounts; Producer 50/50	52,400 input 2,646,200 output	

3.3 End-to-End RSSL/RWF 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 its API and the time the same update is received by the consuming application from its API, i.e. it includes both the latency contribution from the API and the core infrastructure components.

NOTES:

- Caching was disabled in both the Source Distributor and the P2PS during these tests.
- Optimized binaries of the RMDS infrastructure components were used.
- NTP was disabled on the tools node, as any drifts in time will affect the reported latency.
- The Solaris kernel variable hires_tick was set to 1 on (only) the test tools system (sink_driven_src and rmdstestclient) to provide a high resolution timer for latency calculations.
- Tests were run with 100,000 item watchlist and RWF data update size of 74 bytes [Data file (*sample.xml*) was used].
- Latency tests were run at each update rate for at least 6 minutes, up to near the maximum sustainable update rate for a given test setup. The RRCP and the RV test were terminated due to insufficient processor power remaining for the high runner thread of the p2ps process to take another step.
- Decode of data was turned on in these tests.

3.3.1 RRCP Backbone Results

Update Rate [74-byte RWF messages]	Mean Latency (millisec)	Std Deviation (millisec)	Maximum Latency (millisec)	Minimum Latency (millisec)	Number of Latency Points
1,000	0.241	0.008	0.268	0.224	4360
5,000	0.341	0.006	0.479	0.329	4180
10,000	0.430	0.010	0.663	0.404	4180
20,000	0.516	0.020	1.236	0.455	4180
30,000	0.559	0.038	0.890	0.487	4180
40,000	0.606	0.045	1.071	0.510	4150
50,000	0.631	0.093	2.555	0.509	4170
60,000	0.685	0.300	2.601	0.498	4290
70,000	0.651	0.115	2.527	0.493	4110
80,000	0.665	0.128	2.516	0.513	4340
90,000	0.676	0.149	2.604	0.515	4000
100,000	0.667	0.099	2.473	0.503	4150
150,000	0.721	0.143	2.433	0.512	4190
200,000	0.752	0.123	2.363	0.515	4180
250,000	0.802	0.145	2.299	0.524	4150
300,000	0.844	0.168	2.148	0.526	4180
350,000	0.886	0.186	2.439	0.539	4180
400,000	0.913	0.201	2.656	0.545	4160
450,000	0.962	0.247	2.829	0.530	4150
500,000	1.023	0.284	3.154	0.543	4130
550000	1.087	0.329	3.772	0.546	4190

600000	1.187	0.390	4.419	0.548	3470
650000	1.324	0.476	4.840	0.541	4160
700000	1.485	0.586	5.297	0.536	4120
750000	1.828	0.870	5.459	0.559	4110
800000	3.487	0.872	7.353	0.793	7490

3.3.2 Rendezvous Backbone Results

Update Rate [74-byte RWF messages]	Mean Latency (millisec)	Std Deviation (millisec)	Maximum Latency (millisec)	Minimum Latency (millisec)	Number of Latency Points
1,000	0.256	0.004	0.291	0.244	4450
5,000	0.362	0.008	0.470	0.346	4190
10,000	0.448	0.009	0.592	0.422	4400
20,000	0.516	0.017	0.816	0.474	4020
30,000	0.565	0.028	0.717	0.500	4110
40,000	0.604	0.036	0.722	0.523	4150
50,000	0.641	0.073	25.38	0.523	4300
60,000	0.667	0.043	0.790	0.544	4200
70,000	0.658	0.066	2.534	0.535	4150
80,000	0.675	0.172	2.565	0.533	4150
90,000	0.670	0.078	2.651	0.540	4180
100,000	0.685	0.115	2.521	0.529	4160
150,000	0.717	0.087	2.421	0.523	4180
200,000	0.757	0.108	2.402	0.535	4180
250,000	0.815	0.125	2.340	0.536	4180
300,000	0.844	0.140	2.283	0.548	4180
350,000	0.897	0.162	2.119	0.547	4180
400,000	0.933	0.181	2.184	0.552	4170
450,000	0.971	0.194	1.441	0.553	4150
500,000	1.010	0.217	1.908	0.553	4170
550000	1.030	0.228	1.882	0.553	4180
600000	1.125	0.285	2.270	0.564	4170
650000	1.324	0.384	2.776	0.558	4170
700000	1.454	0.439	2.716	0.506	4160