

Extreme Performance Platform for Real-Time Streaming Analytics

Achieve Massive Scalability on SPARC T7 with Oracle Stream Analytics

ORACLE WHITE PAPER | APRIL 2016



Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Table of Contents

Disclaimer	1
SPARC T7 and M7 Servers	2
Oracle Stream Analytics on SPARC T7	3
Oracle Stream Analytics Architecture	4
Data Flow	4
Installation and Configuration of Test Environment	4
Prerequisite	5
Single OSA Instance	5
Multiple Instance Configuration	6
Benchmark	7
Performance Tuning	8
JVM Tuning	8
Solaris Tuning	9
Collecting Throughput Data	9
Benchmark Results	9
Conclusion	11

SPARC T7 and M7 Servers

Oracle's SPARC T7 and M7 Servers built on SPARC M7 processors deliver advanced security, deep integration from applications to cloud, and world record performance. Forming the foundation of a new, open ecosystem, hardware and software are converged, delivering unmatched customer value.

Most Advanced Security - Increasing risks and cyber threats make IT security a high priority. Oracle's SPARC T7 and M7 Servers with always-on memory intrusion protection and comprehensive data encryption secure your data with no performance penalty. Security in silicon features and Oracle Solaris protect data in memory from unauthorized access and stop malware before it gets in.

Breakthrough Integration - By converging software functions into silicon, Oracle's SPARC T7 and M7 servers provide true integration from application to processor. Acceleration of SQL queries for Oracle Database In-memory processing provides outstanding efficiency for running OLTP and data analytics simultaneously.

Extreme Performance - Running the SPARC M7 processor, the fastest in the world, SPARC T7 and M7 servers deliver industry leading performance across a wide range of applications. With 20 world record benchmarks, these new servers are proven to be fast and efficient, and accomplish more in less time for your enterprise applications and cloud services.

Whether you are running your enterprise applications on a traditional IT infrastructure or implementing cloud services, Oracle's SPARC Servers can transform your business with advanced security, breakthrough integration, and extreme performance. Engineered to meet cyber threats head on, SPARC systems offer end-to-end data protection when you need it most. From scale-out to scale-up systems, SPARC Servers are the best for Oracle Database and Java applications and provide the greatest value for enterprise computing.

Oracle Stream Analytics on SPARC T7

Oracle Stream Analytics (OSA) is a platform that allows applications requiring event-driven architecture to filter, query and process events in real-time with low latency. It is built on industry standards including ANSI SQL, Spring DM and OSGi. It has proven to be a high performance complex event-processing engine that allows enterprises to maximize the value of high velocity data from various data sources in real-time. With the ability to process fast data, Stream Analytics is working greatly in a variety of industries. It performs real-time call detail record monitoring in telecommunications industry. It allows financial service providers to perform real-time risk analysis and financial transaction monitoring. Fraud detection applications can be easily developed on the OSA platform. OSA's integrated Oracle Spatial capabilities allow it to perform real-time spatial capabilities like mobile marketing. Due to its high performance, real-time processing nature, OSA can be adapted to a large number of use cases in every industry. At the heart of OSA is the built-in Oracle Continuous Query Language (Oracle CQL) engine. It provides the ability to perform filtering, correlation, aggregation, and pattern matching of the incoming real-time events while connecting them with the persistent data from data stores and in-memory caches. In this whitepaper, an application to detect DDoS attacks is taken as an example to perform performance tuning and benchmarking on Oracle SPARC T7-4 system. The throughput of the benchmark result is reviewed and compared to that of Oracle Exalogic Elastic Cloud X5-4 system.

Oracle Stream Analytics Architecture

Oracle Stream Analytics (OSA) adopts multi-layer software architecture. The Java Virtual Machine (JVM) provides most fundamental support at the lowest level. Above that is the OSGi framework, which manages the Java packages between software modules and deals with class versioning and loading. Spring Dynamic Modules lies above the OSGi framework, which is responsible for service instantiation and dependency injection. Above that comes the OSA server modules layer. This layer provides the core OSA functionality, including the CQL engine, server management and input/output data handling. The highest level in the architecture is the application layer.

Data Flow

A typical data flow through an OSA application starts from incoming event data streams. The data is converted and used by an adapter to create event objects that can be obtained by any component that registered to listen to the adapter. A channel is one of those components that can listen to adapters. Data goes through the channel all the way to the CQL processor component, which is able to efficiently process data using the query language (CQL). The output can be sent to downstream listeners.

Installation and Configuration of Test Environment

We use Oracle Analytics (OSA) and Solaris 11.3 64 bits for the benchmark. One socket of a T7 system is made to exhaust by running 200 OSA instances on it simultaneously. A T7-4 system has a total of 4 sockets. Each OSA instance has a dedicated load generator that simulates real-time incoming events and sends it to the OSA instance. The workflow is explained in detail in the Benchmark Section.

Prerequisite

There are a bunch of setup files required to be configured before deploying and running OSA instances.

1. A compatible Java version.
2. Ready-to-compile source code of the Application, written in Java and CQL, and it's best to be placed at `$OEP_HOME/`.
3. Required load generator files, or event generator files that act as input to the application, if required. OSA installation sets up a default load-generator at `$OEP_HOME/ocep_11.1/utlis/load-generator`. It could be customized to suit the specific requirements.

Single OSA Instance

Compiling the source code packaged into a 'wlevs.jar' file with the appropriate Java version and the required JVM arguments should start up an OSA instance. In this specific app, the 'defaultserver9002' is the directory that has the 'startwlevs.sh' script to start up the server. Running this script will deploy an OSA server which starts listening events at the specified receive port. These settings are in the 'config.xml' file under the 'config' folder.

The next step is to generate event data and send it over to the receiving App port.

Multiple Instance Configuration

For deploying multiple OSA instances with identical configuration, it's easier to follow the steps below for a smooth run – here, about 10 instances are configured to run:

1. Copy defaultserver directory into ten copies named as defaultserver1, defaultserver2... defaultserver10.
2. Once all the copies are made, modify the startwlevs.sh start-up script under each domain directory to make sure the loadgen.port and timesync.port are different across the 10 domains.
3. Modify the config.xml under defaultserver*/config directory accordingly to ensure each listens at a unique port.
4. Run “startwlevs.sh” in different terminals after the above configuration is done to deploy the 10 instances of “Dns Generation App”.
5. Setup the load generator. For each of the ten OSA instances, ten load generators should be created and run. Make sure each of the load generators are configured to send data at the corresponding OSA instance receive port. This will guarantee the data sent from different load generator will feed into different OSA instances. For more details about benchmark, see the subsequent sections.

Benchmark

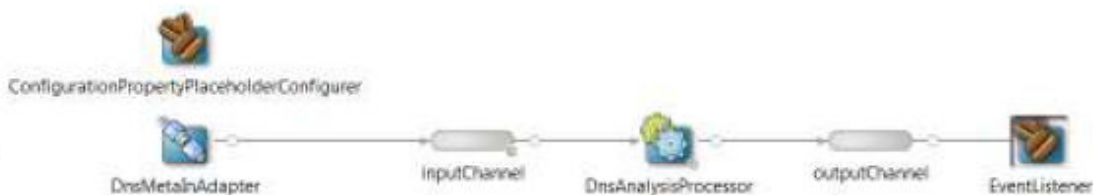
The DNS generator application was mainly laid out to detect malicious IP addresses that result in a Distributed Denial of Service (DDOS) attack on a system. It is accomplished by counting the number of UDP packets received per IP address and sorting the counters to figure out which IP sent the most packets.

To build a test scenario for the application, a dedicated load generator program is built. This program reads from a list of files, which in turn contains several randomized UDP packet information strings. The load generator parses these files line by line, formats these strings into the apt UDP format and sends them over the assigned socket. The load generation can be local or remote.

The OSA application instance is always in a listening mode. And, when it receives data on the socket it is configured to, it starts incrementing the packet counter. Different OSA instances are deployed with different sockets.

The packet counter is printed out in regular intervals as the throughput for testing purposes.

Here is the Event Processing Network (EPN) flow of the DNS generator app. It describes the high level components of the application and the communication flow.



Performance Tuning

JVM Tuning

Before benchmarking, we performed JVM tuning to the OSA to let OSA run in an optimal JVM environment. When tuning JVM, only one load generator and one instance are used. The goal is to let an OSA instance accept as much incoming data stream as possible at a unit time duration. The load generator is configured to send data from 8,000 to 50,000 events/sec to find out the peak value that an OSA instance can accept, which has turned out to be around 34,000 events/sec with a JVM flag configuration shown below:

```
JVM_ARGS='-server -Xms8G -Xmx8G -XX:SurvivorRatio=2 -  
XX:+UseCompressedStrings -XX:+OptimizeStringConcat -XX:+UseStringCache -  
XX:-UseISM -XX:MaxPermSize=4G -XX:NewSize=5G -  
XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass'
```

For the benchmark, about 200 instances were deployed on the T7 socket with a 1TB memory. To reduce the memory usage, the heap sizes are reduced to 2G for each of the JVM instance, thus with a configuration of:

```
JVM_ARGS='-server -Xms2G -Xmx2G -XX:SurvivorRatio=2 -  
XX:+UseCompressedStrings -XX:+OptimizeStringConcat -XX:+UseStringCache -  
XX:-UseISM -XX:MaxPermSize=1G -XX:NewSize=1G -  
XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass'
```

Solaris Tuning

Solaris SPARC comes with a feature that allows for the creation of processor sets, which are basically virtual processor groups. Processes can be bound to run exclusively on these sets.

Each T7 socket has 32 cores with 8 threads each. i.e., a total of 256 threads or virtual processors (or 'vcpu's). 200 of these processors are bound to each of the 200 OSA instances, thus, utilizing 25 cores of the T7 socket. Thus, all the execution of the OSA instance is restricted to that particular thread. At the same time, the other three sockets of the T7-4 i.e. about 750 threads are bound together into one processor set. On this processor set, the 200 load generators for the 200 OSA instances are run. The load generators need about ~2.5 vcpus per instance for an optimal performance.

Collecting Throughput Data

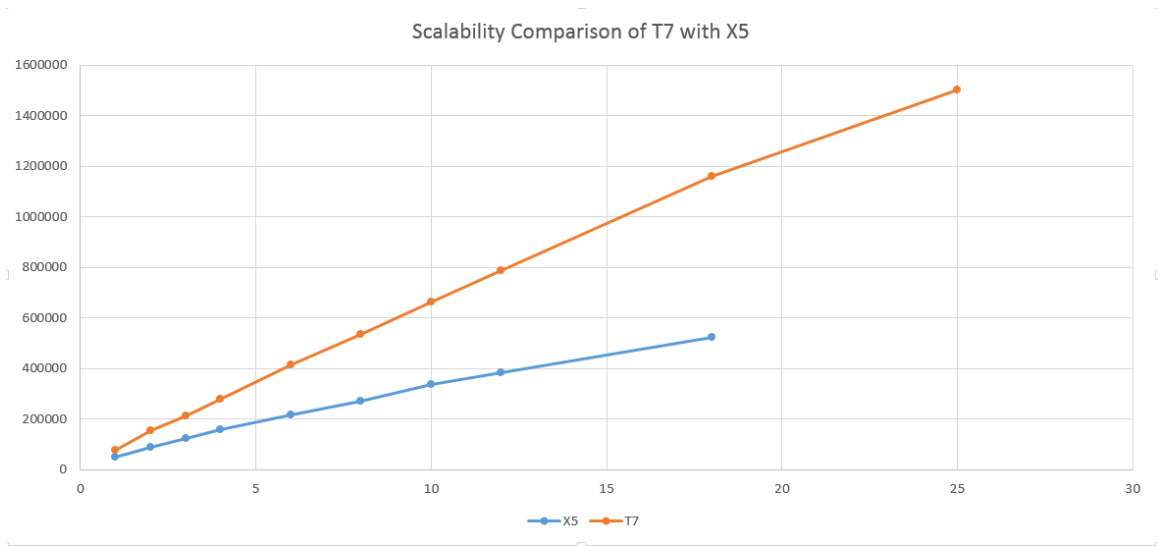
As the application keeps receiving the UDP packets, the number of packets received is collected in regular intervals. This number is averaged out and printed in batches, every 5 seconds for each of the load generators. The throughput for all the load generators is again combined to generate an average throughput per load generator.

Benchmark Results

The following table shows the data collected for different test configurations done at a core-by-core level. The maximum throughput achieved with the 200 OSA instances on one T7-4 socket is 1.505 million operations per second. The same tests were conducted on an Intel Xeon based X5-4 server, for a comparative standard.

The following table shows the performances of both the T7-4 and the X5-4 sockets.

Machine	Cores	Instances	Sending Rate	Throughput/Instance	CPU util(%)	JVM Heap	UDP buffer	Net throughput
X5	1	2	25000	24500	4.44	8G	1G	49000
	2	4	23000	22200	8.88	8G	1G	88800
	3	6	22000	20700	13.33	8G	1G	124,200
	4	8	20000	19500	17.77	8G	1G	160,000
	6	12	19000	18000	30	8G	1G	216,000
	8	16	18000	17000	35.5	8G	1G	272,000
	10	20	17500	16800	44.44	8G	1G	336,000
	12	24	17000	16200	53.33	8G	1G	384,000
	18	36	15000	14500	85	8G	1G	522,000
T7	1	8	10000	9682	2.65	2G	1G	77456
	2	16	10000	9663	4.06	2G	1G	154608
	3	24	9000	8920	6.09	2G	1G	214080
	4	32	9000	8750	10.62	2G	1G	280,000
	6	48	9000	8600	15.93	2G	1G	412,800
	8	64	9000	8369	21.25	2G	1G	535,616
	10	80	9000	8300	26.56	2G	1G	664,000
	12	96	9000	8196	33.01	2G	1G	786,816
	18	144	9000	8045	50.62	2G	1G	1,158,480
	25	200	9000	7527	70.3	2G	1G	1,502,000



As illustrated, the T7-4 clearly outperforms the X5-4 in terms of performance with a world record result running an Oracle Stream Analytics (OSA) benchmark on real world customer workload. A single SPARC M7 processor of a SPARC T7-4 server running OSA achieved a throughput result of 1.505 million ops/sec. The SPARC M7 processor achieved 2.9 times the throughput of an x86 Intel Xeon Processor E7-8895 v3 based server. Oracle Stream Analytics benefits from large number of SPARC M7 processor hardware strands or virtual CPUs and Oracle Solaris scheduling to increase throughput. End-to-end optimized Oracle Stream Analytics and Java on SPARC M7 processor provides low cost solution to customers to quickly process real time events.

Conclusion

This whitepaper reviews the Oracle Stream Analytics (OSA) architecture along with some core features that OSA are equipped with. The main focus of this paper is to provide readers an idea of how OSA performs on Oracle SPARC T-7 system. Note that the benchmark used in this whitepaper uses only one socket of the T7-4 machine. Comparing the data collected from Exalogic X5-4 machine, T7 performs much better with respect to performance using high volumes of incoming data coupled with good scalability. For enterprises that require applications with high throughput and scaling, OSA and T7 can provide a satisfying platform.







Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

-  blogs.oracle.com/oracle
-  facebook.com/oracle
-  twitter.com/oracle
-  oracle.com

Integrated Cloud Applications & Platform Services

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. This document is provided *for* information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0615

April 2016]



Oracle is committed to developing practices and products that help protect the environment