FUSION MIDDLEWARE
SOA SUITE

Oracle Event Processing Extreme Performance on Sparc T5

An Oracle Event Processing (OEP) Whitepaper

ORACLE WHITE PAPER | AUGUST 2014

Table of Contents

Introduction	2
OEP Architecture	2
Server Architecture	2
Data Flow	2
Test Environment	3
JDK version	3
OEP Installation and Test Environment Configuration	3
Multiple Instance Configuration	3
Load Generator Configuration	3
Benchmark	4
Benchmark App	4
Performance Tuning	5
JVM Tuning	5
Solaris Tuning	5
Test Methodology	6
Benchmark Result	6
Comparison with Exalogic	8
Conclusion	8

Introduction

Oracle Event Processing (OEP) is a platform that allows applications requiring event-driven architecture to filter, query and process events in real-time with low latency. OEP 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, OEP 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 OEP platform. OEPs integrated Oracle Spatial capabilities allow it to perform real-time spatial capabilities like mobile marketing. Due to its high performance, real-time processing nature, OEP can be adapted to an large number of use cases in every industry.

At the heart of OEP 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, we will take Signal Generation sample application as an example to perform performance tuning and benchmarking on Oracle SPARC T5 system. We will review the throughput and latency from the benchmark result and compare the result with Oracle Exalogic Elastic Cloud X2-2.

OEP Architecture

Server Architecture

OEP 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 OEP server modules layer. This layer provides the core OEP 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 OEP 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.

Test Environment

TEST SERVER SPECS

System	Value
Physical CPU	4
Core / Physical CPU	16
Virtual CPU / Core	8
Cache Per Processor	
share 8 MB, 8 banked, Level 3 Cache and 128 KB Level 2 unified cache	
Main Memory	
1 TB (64 x 16 GB 1066 MHz DDR3 DIMMs)	

JDK version

Hotspot jdk1.6.0 with update 45 was utilized during the test.

OEP Installation and Test Environment Configuration

We use OEP Solaris 11.1.1.7.0 64 bits in the benchmark. During this whitepaper, we will try maximizing the server load by running ten OEP instances at the same time on two T-5 physical CPUs. All those instances are configured be in a cluster as for easy deployment of the Signal Generation App. Each OEP instance will have one load generator feeding the data simulating real-time incoming events. The detailed workflow will be explained in Benchmark Section.

Multiple Instance Configuration

Once the OEP installation is done, we need to follow the steps below to duplicate the “defaultserver” directory under $OEP_HOME/user_projects/domains/ into ten different copies:

1) Copy defaultserver directory into ten copies named as defaultserver1, defaultserver2… defaultserver10.

2) Once all the copies are made, we need to modify the startwlevs.sh start-up script under each domain directory. Make sure the loadgen.port and timesyc.port are different across the 10 domains.

3) Modify the config.xml under defaultserver*/config directory accordingly. Make sure all the NetIO sslNetIO ports are different across those domains as well.

4) In order to make all domains running within a cluster, we will need to modify the <cluster> section in the config.xml file. Make sure use different <server-name> in each file. Use the same <groups> name across all the domains.

Once above configuration is done, you will need to start up all the domains by running “startwlevs.sh” in different terminals and deploy the “Signal Generation App” onto the cluster.

Now, you’ve created a ten instances cluster with a Signal Generation App running on it. The next step will have to setup the load generator in order to feed data to the application during benchmark.

Load Generator Configuration

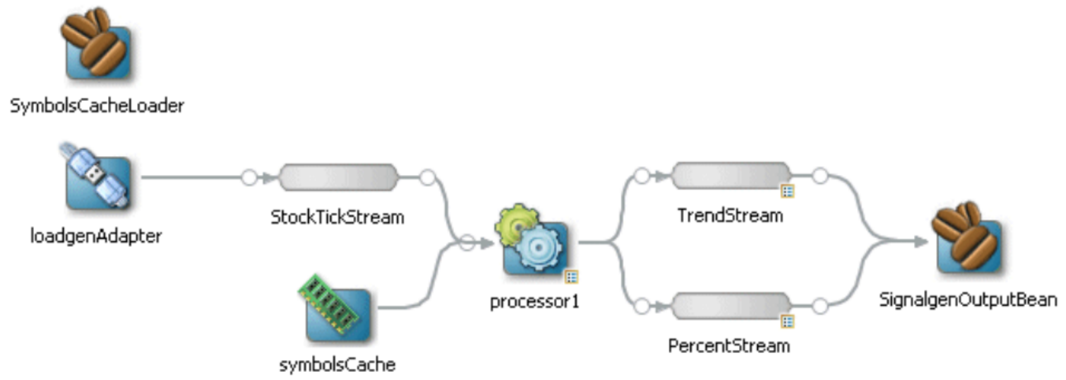
The OEP installation comes with a load generator tool. It’s located in $OEP_HOME/ocep_11.1/utils/load-generator. However, we need to tweak a bit to create a customized property file for the load generator. Since we have ten OEP instances in total, we will need to setup ten load generators correspondingly. Thus, we will have ten customized property files and ten data csv files. Make sure the test.port in each property file is corresponding to each loadgen.port in startwlevs.sh script in each domain directory. This will guarantee the data sent from different load generator will feed into different OEP instances. More details about benchmark will be explained in next sections.

Benchmark

Benchmark App

The Signal Generation Application gives an idea how OEP is involved in applications that are designed to perform analytics upon data from real-time stock market. The workload generator will acting like a real-world stock market that sends real-time stock price to the application through OEP. Signal Generation will verify if the price of certain stock has fluctuated more than 2%, additionally, it also detects if there is a trend occurring to any stock. More specifically, if more than 3 successive prices fluctuate more than 2% in the same way, Signal Generation will consider it a trend.

Below is the Event Processing Network (EPN) of Signal Generation app. It describes the high level components that make up the whole application and how they are work together with each other.



Signal Generation Application Event Processing Network (EPN)

The loadgenAdapter reads the data sent from OEP load generator through TCP/IP socket. Each load generator can establish one or more socket connections to the OEP instance. The data read from the loadgenAdapter will be further transporting to the CQL processor for processing. Another data source for the processor to process is from the symbolsCache. The SymbolsCacheLoader loads the stock symbol cache from a local disk cache file in this sample application case. The cache file contains 300 stock symbols that are monitored by the CQL processor. The CQL processor will filter out symbols from the loadgenAdapter that don't have a corresponding match in the cache file. The input data file of load generator is containing 1470 distinct stock symbols. Below is the core CQL used in the Signal Generation App.

```
<view id="S" schema="symbol lastPrice" ordering-constraint="PARTITION_ORDERED" partition-expression="T.symbol">
  <![CDATA[
    RStream(select T.symbol, T.lastPrice from StockTickStream[now] as T, SymbolsRelation as R where T.symbol = R.symbol)
  ]]>
</view>

<query id="perc" ordering-constraint="PARTITION_ORDERED" partition-expression="symbol">
  <![CDATA[
      select symbol, lastPrice, percLastPrice
      from S MATCH_RECOGNIZE (
        PARTITION BY symbol
        MEASURES
          B.symbol as symbol,
          B.lastPrice as lastPrice,
          100*(B.lastPrice - A.lastPrice)/A.lastPrice as percLastPrice
        ALL MATCHES
        PATTERN (A B)
        DEFINE
          B AS (100*(B.lastPrice - A.lastPrice)/A.lastPrice > 2.0
                or   100*(B.lastPrice - A.lastPrice)/A.lastPrice < -2.0
               )
      ) as T
 ]]>
</query>
```

Signal Generation CQL Queries

More detailed explanation of the CQL queries can be found in the whitepaper Oracle Complex Event Processing Exalogic Performance Study published in September 2011.

Performance Tuning

JVM Tuning

Before benchmarking, we performed JVM tuning to the OEP in order to let OEP run in an optimal JVM environment. When tuning JVM, only one load generator and one instance are used. The goal is to let an OEP instance accept as much incoming data stream as possible at a unit time duration. The load generator is configured to send data from 70,000 to 350,000 events/sec in order to find out the peak value that an OEP instance can accept, which in our case is around 300,000 events/sec. Please note that during the performance tuning stage, we use test.ticksPerPacket=1, which means each packet sent through OEP is containing exactly one stock symbol. However, during the benchmark stage, we use test.ticksPerPacket=10 in order to create a clearer comparison between the result from T5 and Exalogic as the benchmark done Exalogic machine is using test.ticksPerPacket=10. Please also note that a load generator can be configured to have one or more connections to an OEP instance. During our tuning process, we found 4 connection each load generator will yield the best throughput.

As the tuning result, the following JVM flags were applied to the OEP instance before doing the benchmark.

JVM Flag
-XX:NewSize=1g
-Xms1536m
-Xmx1536m
-XX:MaxPermSize=512m
-XX:+CMSConcurrentMTEnabled
-XX:+UseConcMarkSweepGC

JVM Flags Applied

Solaris Tuning

Solaris SPARC comes with a feature that allows you to create a processor set, which you can use to create a virtual processor group and then specify certain processes to be using the processors in only that group. We created 2 processor sets, each of which contains 128 virtual processors. This way, each processor set is working exactly as 1 physical T-5 CPU. Later during the benchmark, we will bind the OEP instance to the two-processor sets, in order to simulate running in a T5-2 system. The purpose to create a simulated two CPU system is for leaving the rest of the CPUs for workload generator usage. That way we can guarantee a better-isolated test environment for the OEP instances.

Test Methodology

In order to correspond with prior performance tests run on Exalogic machine, we performed the test scaling the connection number per load generator from 1 to 8; each load generator set to send 50,000 events/second. Please note that during the benchmarking, we use test.ticksPerPacket=10, which means each packet will contain ten stock symbols. This is going to affect the throughput if you count events/second as the measure unit.

We conducted the benchmark by running ten OEP instances at the same time, each of which is fed data by one load generator. The ten OEP instances are divided into two groups, each of which is executing under a processor set. This makes the CPU load distribute evenly across the two processor sets. As we mentioned before, each processor set can be taken as a single T-5 CPU, so the benchmark is conducted using 2 T-5 CPUs. The load generators are configured to use the rest of the CPUs in T5-4 in order to better isolate the test environment of OEP instances.

The following steps are taken to collect the data:

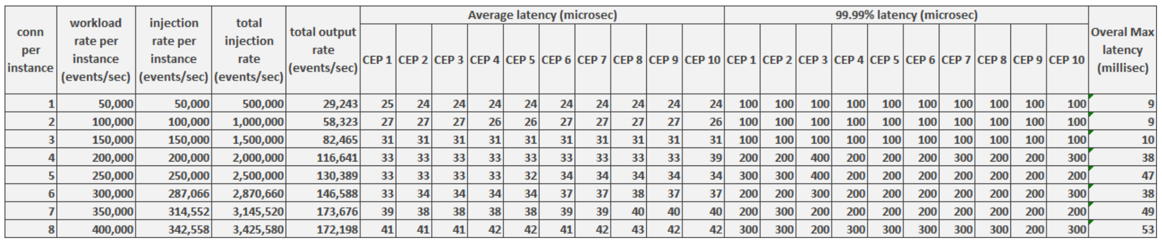
1) Start up all ten OEP instances and conduct a warm up test run for all instances at the rate of 50,000 events/second per load generator. Each load generator is configured to have one connection.

2) Perform 8 connection-scaling runs of all OEP instances, each of which is a 10-minute run. The connection per load generator scales from 1 to 8. The highest workload injection rate is 4,000,000 events/sec (8 * 50,000 * 10).

3) For each run, the injection rate, output rate and latency data were collected.

Benchmark Result

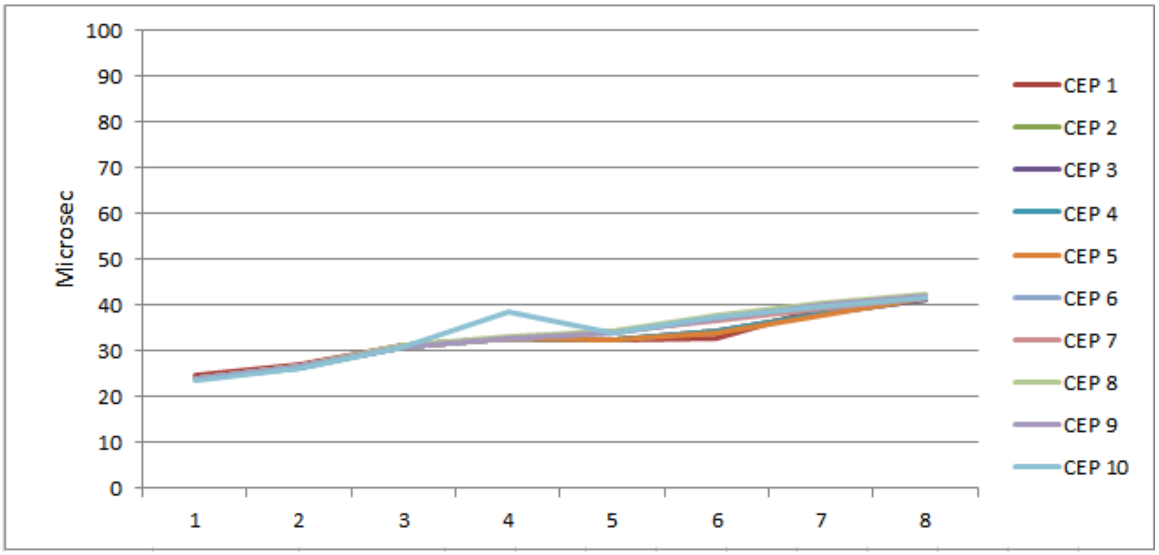
The following table shows the data collected during all test runs. Each row stands for one test run. The workload rate column indicates the load that load generator tries to inject to OEP. The injection rate column indicates the workload that OEP can accept and can be considered as throughput. The total injection rate is the product of number of OEP instances multiplied by the injection rate per instance. We also collected the average latency, 99.99% latency and the Average Max Latency for your reference.



conn per instance	workload rate per instance (events/sec)	injection rate per instance (events/sec)	total injection rate (events/sec)	total output rate (events/sec)	Average latency (microsec)										99.99% latency (microsec)										Overal Max latency (millisec)
					CEP 1	CEP 2	CEP 3	CEP 4	CEP 5	CEP 6	CEP 7	CEP 8	CEP 9	CEP 10	CEP 1	CEP 2	CEP 3	CEP 4	CEP 5	CEP 6	CEP 7	CEP 8	CEP 9	CEP 10	
1	50,000	50,000	500,000	29,243	25	24	24	24	24	24	24	24	24	24	100	100	100	100	100	100	100	100	100	100	9
2	100,000	100,000	1,000,000	58,323	27	27	27	26	26	27	27	27	27	26	100	100	100	100	100	100	100	100	100	100	9
3	150,000	150,000	1,500,000	82,465	31	31	31	31	31	31	31	31	31	31	100	100	100	100	100	100	100	100	100	100	10
4	200,000	200,000	2,000,000	116,641	33	33	33	33	33	33	33	33	33	39	200	200	400	200	200	200	300	200	200	300	38
5	250,000	250,000	2,500,000	130,389	33	33	33	33	32	34	34	34	34	34	300	300	400	200	200	200	200	200	200	200	47
6	300,000	287,066	2,870,660	146,588	33	34	34	34	34	37	37	38	37	37	200	200	300	200	200	200	200	200	200	300	38
7	350,000	314,552	3,145,520	173,676	39	38	38	38	38	39	39	40	40	40	200	300	200	200	200	200	200	200	200	200	49
8	400,000	342,558	3,425,580	172,198	41	41	41	42	42	41	42	43	42	42	300	300	200	300	300	300	300	300	200	300	53

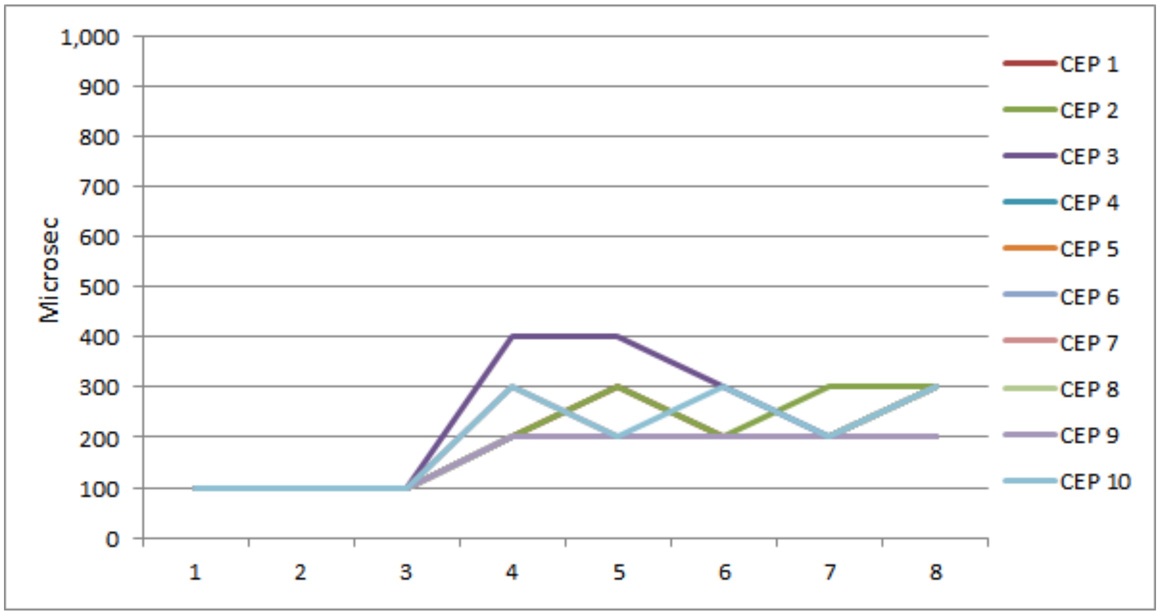
Connection Scaling Benchmark Result

The following chart shows the average latency of ten OEP instances when performing benchmark scaling the connection per load generator from 1 to 8. As you can see from the chart, the average latency of each OEP instance is staying at a low level even though the throughput has been increase from 500,000 events/sec to nearly 4,000,000 events/sec in total. This indicates that the OEP is equipped with a very high performance in terms of low latency at high throughput when running on SPARC T5.



Average Latency at 50,000 events / second / connection

The next chart shows the 99.99 percent latency when performing test run from 1 to 8 connections per instance. We can see that even with 4,000,000 events per second workload rate, the OEP can still maintain 99.99 percent latency lower than 400 microseconds.



With the data collected from all test runs, we can see that OEP has the ability of maintaining extremely low latency while providing the processing ability on a very high volume incoming data stream. In addition, each T-5 CPU is configured with five OEP instances running at the same time. According to the data, T-5 was doing very well in terms of keeping multiple instances running at a good performance, which indicates great scalability of the T-5 CPU.

Comparison with Exalogic

Comparing the data collected in this whitepaper with the data from Exalogic X2-2 tests, we can draw the conclusion that when receiving the same amount incoming streaming data, OEP running on Sparc T-5 is an equally viable solution as Exalogic X2-2 in terms of keeping a low latency. The highest throughput that the Exalogic test performs was 1,000,000 events per second per node with an average latency around 32 microseconds and the overall max latency of 34.5 milliseconds. In our test, when receiving 1,000,000 events per second data, the average latency is only around 26 microseconds. In this test, we scaled to a higher throughput pushing the number of OEP instances and the total throughput to a much higher level than the test did on Exalogic machine. We scaled the number of OEP instances to 10 and the total workload rate reached 4,000,000 in our benchmark proving that OEP on Sparc scales extremely well.

Conclusion

This whitepaper reviews the OEP architecture along with some core features that OEP are equipped with. The main focus of this paper is to provide readers an idea of how OEP performs on Oracle SPARC T-5 machine. Please note that the benchmark conducted in this whitepaper is only using half of a T5-4 machine. Comparing to the data collected from the test done on Exalogic X2-2 machine, T5 performs slightly better in terms of keeping the latency low while receiving high volumes of incoming data. Meanwhile, the T5 machine also shows great scalability when running 10 OEP instances at the same time. For enterprises that require applications with high throughput and low latency performance, OEP and T5 can provide a satisfying platform. For more information regarding to the test done Exalogic machine, please refer to Oracle Complex Event Processing Exalogic Performance Study solution.

Author: Bin Liu

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200