Accelerate Oracle Database using Oracle TimesTen as an Application-Tier Cache

Using Oracle Real Application Clusters 19c, Oracle TimesTen Application-Tier Database Cache 18c, Intel Optane Persistent Memory, and Dell EMC Enterprise Infrastructure

June 2021

H18733

Reference Architecture Guide

Abstract

This document showcases how Oracle TimesTen, used as an application-tier cache with a backend Oracle RAC database, can accelerate performance and efficiency in specific application scenarios when deployed on a Dell EMC infrastructure. It also demonstrates the advantages of using Intel Optane PMem in an Oracle TimesTen In-Memory Database environment.

Dell Technologies Solutions

Copyright

Contents

Executive summary

Introduction

Oracle Real Application Clusters (RAC) provides scalable performance, rich functionality, and high availability. It is the database chosen by thousands of organizations around the world.

Oracle TimesTen In-Memory Database (TimesTen) is an industry-leading in-memory relational database that provides breakthrough performance, low latency, and high availability to the most demanding applications.

In general, combining RAC with TimesTen—deployed as an application-tier database cache (TimesTen cache)—gives customers the best of both worlds. This especially applies in scenarios where long running queries, or a heavy Data Manipulation Language (DML) workload, running on a subset of Oracle data, depletes many of the Oracle RAC resources. Heavy DML workloads can lead to a lack of Oracle resources for other database operations and can have an impact on the overall database response times.

In this solution, we deploy a TimesTen cache in front of a RAC database to offload processing (both queries and DML) from the RAC database. We measure how the deployment impacts the overall database performance and the resource utilization of the RAC database. The study includes end-to-end Dell EMC enterprise IT infrastructure—PowerEdge servers, Dell EMC Networking switches, and Dell EMC Enterprise shared-data storage arrays.

The study also includes the use of Intel Optane persistent memory (PMem) to increase the capacity of the TimesTen cache, enabling far greater data volumes to be cached within a single server than in a DRAM-only configuration. We measured the performance impact of running TimesTen on PMem in Memory Mode (PMem-MM) when configured as a database cache in front of a RAC database.

Use cases overview, results, and key observations

The study and benchmarking in this guide were based on a schema, dataset, and application that simulate the kind of processing performed in a telco Home Location Register (HLR) or Home Subscriber Server (HSS) environment. The schema has four tables— SUBSCRIBER, ACCESS_INFO, SPECIAL_FACILITY and CALL_FORWARDING—arranged in a parent-child hierarchy with SUBSCRIBER as the root table.

In this study, the total dataset generated in the RAC database consisted of data for 2,500,000,000 subscribers (2,500M subscribers). To assess the impact of TimesTen caching with a backend RAC database schema, the two use cases we tested targeted a subset of the total dataset: 400,000,000 subscribers (400M subscribers) for Use Case 1 and 1,800,000,000 subscribers (1,800M subscribers) for Use Case 2.

In Use Case 1, the 400M subscribers were cached in TimesTen running on 768 GB of DRAM-only modules. In Use Case 2, we added 3 TB of Intel PMem to the TimesTen server system, and loaded 1,800M subscribers into TimesTen cache. During Use Case 2, 768 GB of DRAM modules acted as the front-end cache to PMem. These DRAM modules were transparent to TimesTen.

For all tests, the purpose-built HLR application also generated the benchmarking workload. The HLR workload simulated an Online Transaction Processing (OLTP) workload environment with a read/write I/O distribution ratio of 90/10.

For each use case, we ran the HLR benchmark workload directly against the RAC database in order to establish a baseline and measured various performance metrics at the HLR benchmark application, database, and operating system levels. We then cached the use case data (sub)set into the TimesTen cache and ran the HLR workload against the cache. We measured the same performance metrics for both the TimesTen system and the RAC database so that we could compare them. The RAC database performance metrics during the TimesTen tests included the TimesTen replication agent asynchronously propagating the transaction updates and committing them to the backend RAC database. For more details on replication, refer to TimesTen cache AWT replication overview.

Use Case 1 - Small (400M subscribers) TimesTen cache on DRAM

During Use Case 1 testing, we observed the following results:

- During the baseline tests, the RAC database scaled well in terms of transactions per second (TPS) performance, achieving 24 times more TPS at the peak workload thread count of 64 than at the starting workload thread count of 2. However, at higher workloads there was a noticeable increase in average transaction latency, especially for DML operations.
- Comparing the seven HLR transaction latencies at the maximum workload throughput used during the testing, TimesTen's average transaction response time across the seven transactions was 37 times faster than the average transaction response time delivered by the RAC database.
- By offloading the RAC database (HLR) transactions to TimesTen cache:
  - The RAC node's CPU utilization went down from 12 percent to 8 percent—a delta of 4 percent.
  - The RAC database's 'log file sync' wait events were eliminated. The wait events reduced from 65.9 percent of DB Time[1] in the baseline test down to 5.4 percent of DB Time during the TimesTen tests.
  - The RAC database's CPU utilization time (DB CPU[2]) increased from 41.6 percent DB time to 76.9 percent DB time—a delta of 35 percent.

Use Case 2 - Large (1,800M subscribers) TimesTen cache on PMem

During Use Case 2 testing, we observed the following results:

- During the baseline tests, the RAC database scaled well in terms of TPS performance, achieving 41 times more TPS at the peak workload thread count of 64 than at the starting workload thread count of 2.

[1] DB Time is a timed metric in Oracle AWR report that captures the total time spent by Oracle user processes either actively working or actively waiting in a database call or operation.

[2] DB CPU is a timed metric in Oracle AWR report that captures the DB Time spent on 'actively working'; that is, using CPU to perform database work rather than waiting on other resources.

- Comparing the seven HLR transaction latencies, at the same maximum workload throughput as for Use Case 1, the TimesTen average transaction response time across the seven transactions was 35 times faster than the average transaction response time delivered by the RAC database.
- By offloading the RAC database (HLR) transactions to TimesTen:
  - The RAC node's CPU utilization went down from 19 percent to 10 percent—a delta of 9 percent.
  - The RAC database's 'log file sync' wait events were eliminated. The wait events reduced from 61.2 percent of DB Time in the baseline test to 2.8 percent of DB Time during the TimesTen tests.
  - The RAC database's CPU utilization time (DB CPU) increased from 31.8 percent DB time to 50.0 percent DB time—a delta of 18 percent.

Note: The replication lags, latencies, and the maximum throughput or TPS rate we observed in this study are strictly applicable to our setup in the lab and are NOT indicative of the maximum performance capability of the individual hardware and software products, including the schema, dataset, and the workload used in the solution. Performance results will vary based on the deployed environment.

The two use case results showed that, overall, the two-node RAC cluster, configured as the baseline setup, scaled well.

When we deployed a TimesTen cache in front of the baseline RAC setup, TimesTen could offload query and DML transactions from the backend RAC database. This is reflected in the RAC node's CPU utilization and a reduction in the RAC 'log file sync' wait events. With the wait events significantly reduced, the RAC database's CPU utilization efficiency also improved which is reflected by the increased DB CPU metric in the AWR report. This demonstrates that a TimesTen cache can help to improve RAC database consolidation and provide better return on RAC infrastructure investment.

In addition, these results show that when TimesTen is deployed as a cache, it greatly improves the database transaction response times while asynchronously persisting the transactions in the backend RAC database.

Using Intel PMem in Use Case 2 demonstrated that PMem can not only provide greater capacity than DRAM-only configurations for TimesTen within a single PowerEdge server, but it can do so without compromising the overall database performance.

Multiple independent TimesTen servers can be deployed to cache mutually exclusive partitions and to accelerate different datasets of a backend RAC database.

Customers can confidently use this reference architecture, based on their database size and needs, to deploy multiple independent TimesTen cache servers on multiple PowerEdge servers populated with PMem to offload and accelerate their backend RAC databases.

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email or provide your comments by completing our documentation survey.

Authors: Naveen Iyengar (Dell Technologies), Chris Jenkins (Oracle)

Contributor: Reed Tucker (Dell Technologies)

Note: The Dell Technologies Oracle Info Hub provides links to additional documentation for other Oracle solutions.

Solution architecture and components overview

Architecture overview

As shown in the figure below, the architecture that we tested included a dedicated Dell EMC PowerEdge server as the (TimesTen) application or client server, two dedicated PowerEdge servers for the two-node RAC database cluster, and a Dell EMC all-flash enterprise storage array for storing the Oracle data and HLR benchmarking schema. We used two 25 GbE top-of-rack (ToR) switches for all the database traffic between the application server, Oracle RAC nodes, and the shared storage array. We used two 1 GbE Dell EMC Networking switches for any management traffic.

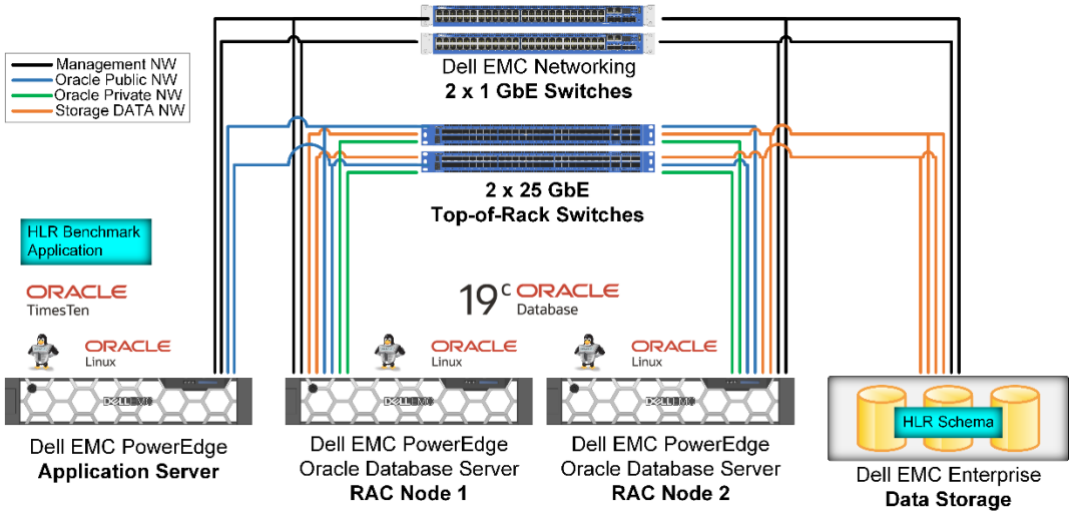

Figure 1. Solution physical architecture overview

TimesTen cache application server tier

We used a two-socket PowerEdge server running two Intel Xeon Scalable 28 core processors and Oracle Linux 8.3 as the application server. Two 25 GbE network ports on two different adapters were Linux-network-bonded to carry the database traffic and connect to the backend RAC database servers. We installed the HLR benchmark application and TimesTen on this server. We configured dedicated local SAS-SSDs within the application server for the TimesTen cache checkpoint files and transaction logs.

We used two different memory configurations. During Use Case 1, the application server had 768 GB of DRAM available as the main memory for TimesTen. For Use Case 2, we added 3 TB of Intel Optane PMem (Memory Mode) modules that were available as the main memory for TimesTen. In this use case, the 768 GB of DRAM in the server acted as PMem's front-end cache; this was transparent to the operating system and applications.

Two-node RAC servers

We used two 2-socket PowerEdge servers as the two-node RAC servers, with each running Oracle Linux 7.8. Each RAC node had two dedicated 25 GbE network ports

(Linux-network-bonded) for the Oracle public traffic, two dedicated 25 GbE network ports for the two Oracle private interconnect networks and two dedicated 25 GbE network ports (Linux-network-bonded) for the shared-storage data traffic. Each database server had 768 GB of physical memory and two Intel Xeon Scalable 18 core processors. We installed a two-node Oracle Grid Infrastructure 19c and RAC database on these two servers.

TimesTen cache AWT replication overview

In this study, we configured and tested the TimesTen cache in asynchronous write-through (AWT) mode. In this mode, the propagation of transactions from the TimesTen cache to the backend Oracle Database is asynchronous and is done in batches. When the application completes a transaction and issues a commit, the commit executes and completes in TimesTen and the transaction is then protected by TimesTen's native persistence mechanisms.

When the transaction commits in TimesTen, it becomes a candidate for propagation to the backend database. The TimesTen replication mechanism will capture the transaction from the TimesTen log stream (usually from the in-memory log buffer) and add it to the current batch. When sufficient transactions have accumulated (approximately 256 KB of data), the batch is applied to the backend database and committed there.

The propagation and the commit-apply process are parallelized and use dependency tracking to maintain the correct ordering in terms of both DML operations and commits. The replication propagation is real-time and the delay between a transaction committing in TimesTen and committing in the backend database is typically a few milliseconds.

Note: TimesTen cache also supports synchronous write-through (SWT) mode, which provides a tighter synchronization between the TimesTen cache and the Oracle database but at a considerable cost in terms of DML performance. Most customers prefer to use AWT to maximize performance. If the TimesTen cache is implemented as a HA pair using TimesTen replication (recommended for production configurations), then updates originating in the cache are fully protected against loss due to failures.

Dell EMC shared storage array

We used a four-node Dell EMC software-defined, all-flash, SAS-SSDs-based enterprise array as the shared Oracle Database storage. Each storage node had two dedicated 25 GbE ports for data traffic. On the shared storage array, we created eight volumes for Oracle DATA (includes HLR benchmarking schema and TimesTen cache management objects), four volumes for Oracle REDO logs, three volumes for Oracle OCR-Voting Disks, and six volumes for Oracle TEMP, and presented them to the two RAC nodes.

25 GbE Networking Switches

Two 25 GbE top-of-rack switches provided the networking. For security, performance, and as a best practice, on the two switches, we separated the network traffic by configuring different VLANs for the different network functions. As an example, we configured the management traffic to use VLAN ID 100, Oracle public network to use VLAN ID 101, two Oracle private networks to use VLAN IDs 102 and 103, and the two storage data networks to use VLAN IDs 104 and 105. For performance, we configured jumbo frames on the switch ports used for the Oracle public, private, and storage data traffic.

For details on hardware and software, refer to Hardware and software configuration details.

Use cases, test methodologies, and results

The main goal of this study was to deploy TimesTen on a separate application server in front of a RAC database to offload processing of both queries and DML operations on a subset of dataset residing on the backend RAC database. During the workload tests, we wanted to measure the impact of TimesTen on the overall database performance and on the resource utilization of the RAC database.

In order to measure the impact of using TimesTen as a database cache to a backend RAC database, we created two use cases:

- **Use Case 1: Small TimesTen cache on DRAM-only**—In this use case the TimesTen server had system memory using DRAM-only modules. Only a relatively small subset of the RAC dataset was loaded into the TimesTen cache running on this DRAM-only configuration.
- **Use Case 2: Large TimesTen cache on PMem-MM**—In this use case the TimesTen server had system memory using Intel PMem and DRAM (acted as front-end cache to PMem) modules. In general, with its larger capacity modules compared to DRAM modules, PMem enables loading larger RAC datasets into TimesTen within a single server when compared to DRAM-only configurations. In this study, PMem was configured to run in Memory Mode (PMem-MM) in the PowerEdge server's BIOS.

HLR benchmark schema overview

To test the database environment, we created an Oracle Database schema using a purpose-built benchmarking application and workload generator called HLR that simulates a Home Location Register (HLR) type of schema, which is typically seen in the Telco industry.

The HLR schema consists of four tables - `SUBSCRIBER`, `ACCESS_INFO`, `SPECIAL_FACILITY` and `CALL_FORWARDING`. `ACCESS_INFO` is a child of `SUBSCRIBER`, as is `SPECIAL_FACILITY`. `CALL_FORWARDING` is a child of `SPECIAL_FACILITY`. All tables are keyed off the `subscriber_id (S_ID column)`.

The dataset that we created in the RAC database for our testing consisted of 2.5 billion `SUBSCRIBER` rows, 12.5 billion `ACCESS_INFO` rows, 12.5 billion `SPECIAL_FACILITY` rows, and 250 million `CALL_FORWARDING` rows. The total number of rows in the `CALL_FORWARDING` table varies slightly as the tests run. This is because the benchmark application performs some INSERT and DELETE database operations against this table, although these operations only make up a small fraction of the overall workload. We created two partitions on all tables and indexes based on the value of `S_ID` such that each partition contained exactly half the number of rows.

The following table shows the number of `SUBSCRIBER` rows that were targeted for each use case and the corresponding size in GB when loaded in the respective databases.

Table 1. Use Cases target schema sizes in terms of SUBSCRIBER rows and in GB

Use case	Target SUBSCRIBER rows	Target schema and data size in	
		Oracle database	TimesTen cache (Physical memory size and type)
Total schema →	2,500 Million	2,062 GB	N/A
Use Case 1 (Small)	400 Million	330 GB	578 GB (768 GB DRAM-only)
Use Case 2 (Large)	1,800 Million	1,485 GB	2,447 GB (3 TB PMem-MM)

HLR benchmark workload overview

The HLR benchmarking application was also used to generate the benchmarking workload against the databases. The workload generated by the HLR benchmarking application simulates an OLTP workload. In terms of SQL operations, the read/write distribution ratio generated by the HLR workload was 90 percent reads (select) and 10 percent writes (insert/update/delete).

The benchmark consists of seven different business transactions, each composed of one or more database operations (statements). The following table shows the different transactions and their percentage compositions of the overall workload.

Table 2. HLR workload transactions and their percentage compositions

Transaction name	Percentage of workload	Type of transaction	
		Query/DML	Read (R) / Write (W)
GetBasicSubscriberData	35%	Query	100% R
GetAccessData	35%	Query	100% R
GetNewDestination	10%	Query	100% R
InsertCallForwarding	2%	Both	50% R + 50% W
DeleteCallForwarding	2%	Both	50% R + 50% W
UpdateLocation	14%	Both	50% R + 50% W
UpdateSubscriberData	2%	DML	100% W
Total	100%		89% / 11%

Test methodology overview

The HLR benchmark application was installed on, and always ran on, the application (TimesTen) server. To assess the impact of caching the RAC database in TimesTen, for both use cases, we first ran the HLR workload directly against the backend RAC database and then ran the same HLR workload against the TimesTen cache with updates propagating to the RAC database from the TimesTen cache asynchronously.

Each workload run contained multiple iterations. Each iteration had a varying and increasing number of application threads (thread counts (TC)) or concurrent database connections that the HLR workload generated. For each iteration, we set the ramp-up time to two minutes, the measurement time to five minutes, and the ramp-down time to one minute, so that each iteration's total duration was eight minutes. During each iteration's measurement period, the HLR application captured the total transactions

throughput measured as Transactions per Second (TPS) and the latency (in microseconds) of each of the seven HLR transactions.

We determined the size of the HLR schema to target for each use case by calculating the number of SUBSCRIBER rows that we could explicitly load into the TimesTen cache. We targeted the same number of SUBSCRIBER rows during the respective use case's baseline tests where we ran the HLR benchmark directly against the RAC database.

For the baseline runs directly against the RAC database, we ran two separate instances of the benchmark application, each targeting one of the RAC instances and a disjointed set of data that matched the partitioning scheme of the HLR schema tables and indexes. This was to avoid potential RAC data block ping-pong effects[3] which can be detrimental for OLTP-style workloads. Similarly, during the TimesTen benchmarking runs, we configured the TimesTen replication agents to connect to only one RAC instance. This was also done to avoid the potential backend RAC block ping-pong effects.

The TimesTen replication agent asynchronously batches and propagates the committed updates in the TimesTen cache tables to the cached HLR schema tables in the backend RAC database. We used a purpose-built script to measure the replication lag of the propagation of the committed updates to the backend RAC database. During the initial trial and error tests, using the measured replication lag, we then determined the maximum throughput or TPS rate at which the replication lag was not constantly increasing. This was indicative of the fact that the backend RAC database could keep up with the replication flow that the TimesTen agent pushes. We capped both the TimesTen and the baseline RAC-only tests to this same maximum throughput or TPS value during our final measured runs.

Note: The replication lags, latencies, and the maximum throughput or TPS rate we observed in this study are strictly applicable to our setup in the lab and are NOT indicative of the maximum performance capability of the individual hardware and software products, including the schema, dataset, and workload used in the solution. Performance results will vary based on the deployed environment.

For more details on TimesTen replication, refer to TimesTen cache AWT replication overview. For more specific details about the use cases, see the following sections.

Use Case 1: Small TimesTen cache on DRAM

The goal of this use case was to study the performance and resource utilization impact of a small TimesTen cache when caching relatively a small subset of the backend RAC database.

Configuration overview

The following figure shows the dataset configuration for Use Case 1.

[3] Scenario where there is increased interconnect I/O because the data blocks that each RAC instance requires exist in the other instance's buffer cache.

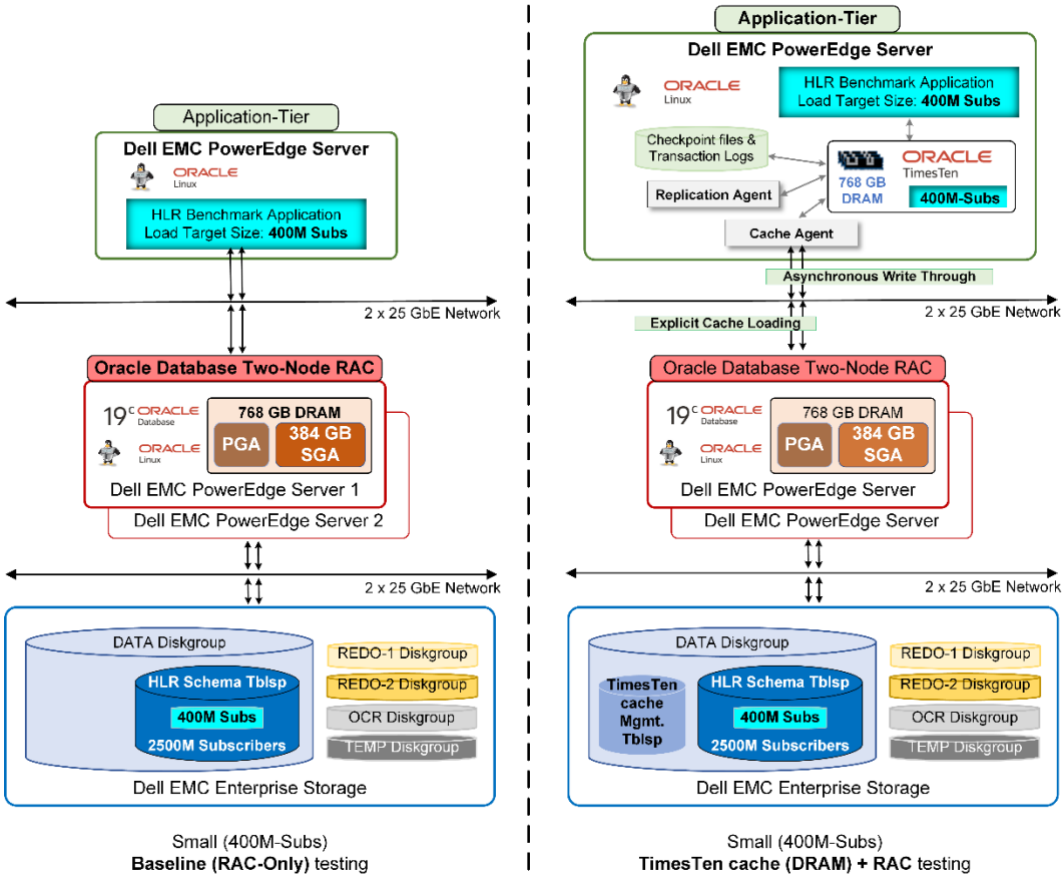

Figure 2. Use Case 1: Small (400M Subs) dataset configuration overview

In Use Case 1, we tested with two configurations as shown by the two adjacent columns in the figure above. The left column in the figure shows the baseline configuration and the right column in the figure shows the TimesTen test configuration.

In this use case, the two-socket PowerEdge server used as the application (TimesTen) server hosts both the HLR benchmark application and TimesTen. It has a total physical memory capacity of 768 GB using DRAM-only modules. Given this main memory size, we could explicitly load a maximum of 400 million SUBSCRIBER rows (400M subscribers) into the TimesTen cache. Therefore, we set the target workload data size for both the test configurations in this use case to 400M subscribers. 400M subscribers is only sixteen percent of the total HLR schema size of 2,500 million SUBSCRIBER rows in the backend RAC database; therefore, this use case represents the scenario in which a relatively small subset of the RAC database is cached in the TimesTen cache.

Each of the two-node RAC servers has 768 GB of physical memory. We set the Oracle database's System Global Area (SGA) to 384 GB. For details on the RAC and TimesTen database configuration, refer to Appendix: Configuration details.

Test methodology

During the baseline tests, we ran the HLR benchmarking workload directly against the backend two-node RAC database setup with a target workload size of 400M subscribers. During each of the baseline tests, we ran seven workload iterations with increasing HLR

application thread counts of 2, 4, 8, 16, 32, 48, and 64. We repeated the baseline tests three times and reported the averages of the three runs.

Note: During the baseline tests, the TimesTen database and cache were disabled on the application server.

During the TimesTen tests, we first explicitly loaded 400M subscribers into the TimesTen cache running on 768 GB of DRAM and configured in Asynchronous Writethrough (AWT) caching mode. We then ran the HLR benchmarking workload against the TimesTen cache limiting the target throughput to the maximum TPS rate achieved against RAC.

Note: As described in section Test methodology overview the limited target TPS rate for the TimesTen tests were nearly the same as the highest TPS achieved by the baseline tests.

Performance results

The figure below shows the results of the baseline tests generated by the HLR application for the workload target size of 400M subscribers that was run directly against the RAC database. The TPS values are plotted relative to the TPS throughput achieved at workload thread count 2 iteration. All latency values are plotted relative to the `InsertCallForwarding` (ICF) transaction latency observed at workload thread count 2 iteration.

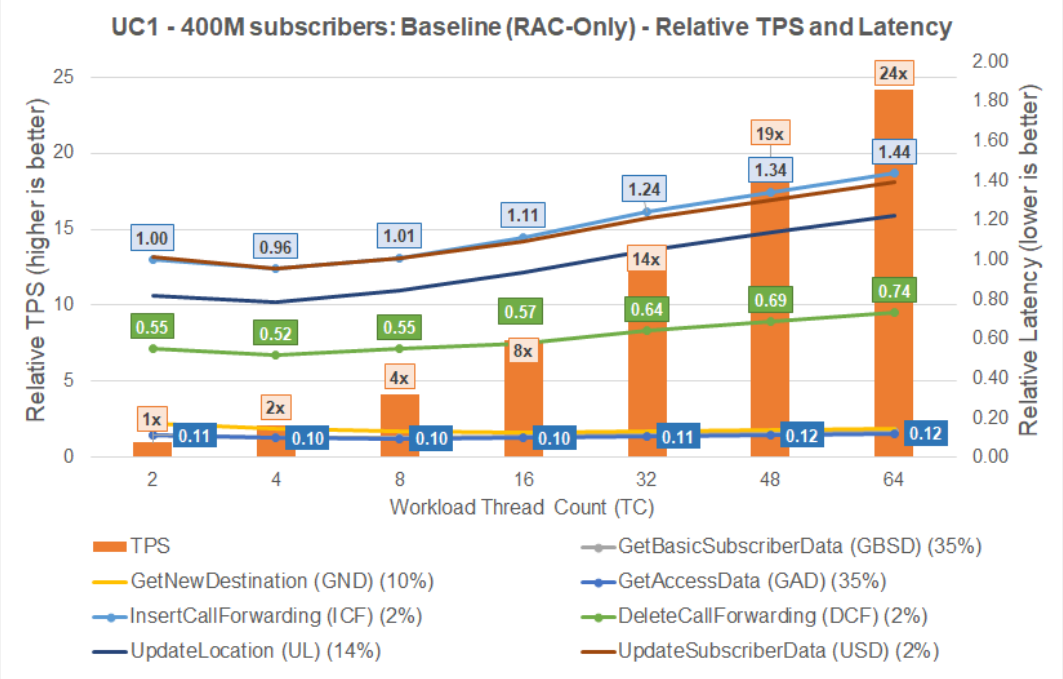

Figure 3. Use Case 1 - 400M subscribers - Baseline (RAC-Only) - Relative TPS and Latency performance graph

As shown in the figure above, the RAC database and backend infrastructure scaled well in terms of TPS, delivering 24x more TPS at the peak workload thread count of 64 when compared to the TPS at the starting workload thread count of 2. However, the latency of DML transactions noticeably increased as the workload increased. This is reflected in the figure above by the relative transaction latency values which increased by up to 44

percent for the two write intensive transactions—`InsertCallForwarding` and `UpdateSubscriberData`.

The figure below compares the application latencies for each of the seven HLR transactions between the baseline tests and the TimesTen cache tests.

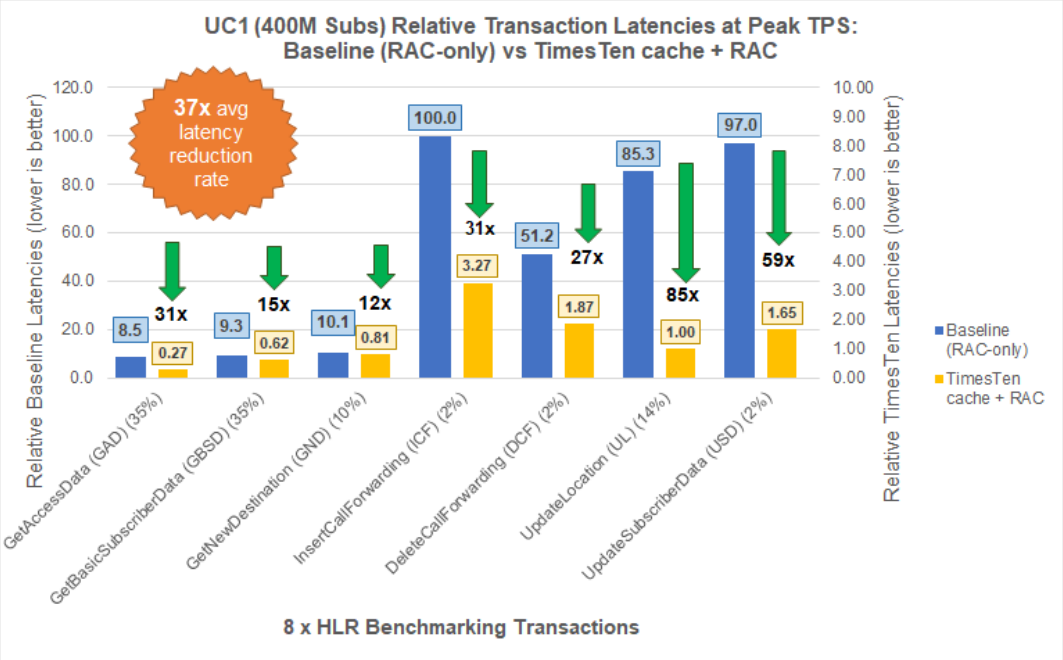

Figure 4. Use Case 1 (400M subscribers) - Average Latencies: Baseline (RAC-only) vs TimesTen cache with RAC

To reiterate the test methodology, the above transaction latencies were captured for both the baseline and the TimesTen tests when they both delivered near about the same peak TPS throughput performance. All latencies in the figure above are relative to the baseline `InsertCallForwarding` (ICF) transaction latency. As shown in the figure above, in Use Case 1 where we tested with a relatively small dataset size (400M subscribers), the TimesTen cache could deliver the same throughput as the RAC database but with an average of 37x better transaction response time across the seven different mixed types of query and DML transactions.

Part of the study was also to determine how a TimesTen cache impacts the resource utilization of the backend RAC database and its infrastructure.

The following figure shows the CPU utilization of RAC node2 as reported by the Oracle Automatic Workload Repository (AWR) report that we captured during both the baseline and the TimesTen cache tests.

Host CPU 12.3% Utilization - Use Case1 - Baseline (RAC-only)

CPUs	Cores	Sockets	Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
72	36	2	0.80	15.55	8.8	3.5	0.0	87.1

Host CPU 8% Utilization - Use Case 1 - TimesTen cache (DRAM) + RAC

CPUs	Cores	Sockets	Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
72	36	2	3.10	1.30	5.2	2.8	0.1	91.7

Figure 5. RAC node2 CPU utilization - Use Case 1 - Baseline vs TimesTen cache tests

As shown in the AWR report, the RAC node2's CPU utilization went from 12.3 percent (8.8 %User + 3.5 %System) during the baseline tests down to 8 percent (5.2 %User + 2.8 %System) during the TimesTen tests—a delta of 4 percent. Though not significant for our particular test setup, this definitely provides a data point that the TimesTen cache helped to offload queries and reduce the RAC node's CPU utilization. Therefore, TimesTen cache can help to improve database consolidation on fewer servers and provide better return on RAC infrastructure investment.

Similarly, when we compared the 'Top 10 Foreground Events by Total Wait Time' from the AWR reports of the two test cases—baseline vs TimesTen—we observed that during the TimesTen cache tests the 'log file sync' wait events that were observed during the baseline tests were nearly eliminated (65.9% DB time in baseline vs 5.4% DB time in TimesTen tests), as shown in the comparative figure below.

Top 10 Foreground Events by Total Wait Time Use Case 1 - Baseline (RAC-only)

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
log file sync	67,219,662	52.4K	778.90us	65.9	Commit
DB CPU		33K		41.6	
SQL*Net message to client	4.5E+08	517.8	1.14us	.7	Network
SQL*Net break/reset to client	5,597,923	239.1	42.71us	.3	Application
cursor: pin S	123,438	131.4	1.06ms	.2	Concurrency
gc current grant busy	353,382	25.1	70.92us	.0	Cluster
db file scattered read	39,369	23.9	607.61us	.0	User I/O
db file sequential read	23,978	11.6	483.26us	.0	User I/O

Top 10 Foreground Events by Total Wait Time Use Case 1 - TimesTen cache (DRAM) + RAC

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
DB CPU		2713.8		76.9	
db file scattered read	319,109	259.8	814.25us	7.4	User I/O
db file sequential read	345,803	237.9	687.83us	6.7	User I/O
log file sync	325,736	189.4	581.52us	5.4	Commit
library cache: mutex X	178,410	142.6	799.46us	4.0	Concurrency
library cache: bucket mutex X	7,630	11.5	1.50ms	.3	Concurrency
cursor: pin S	7,574	8.1	1.07ms	.2	Concurrency
SQL*Net more data from client	286,486	6.5	22.79us	.2	Network

Figure 6. Foreground Wait Events - Use Case 1 - Baseline vs TimesTen cache tests

Also, the RAC database spent more time using the CPU (which is a good thing) than waiting on other resources during the TimesTen tests (DB CPU = 76.9% DB time) when compared to its CPU utilization during the baseline tests (DB CPU = 41.6% DB time)—a delta of 35 percent. This finding provides another proof point that running a TimesTen cache contributes to better total system capacity and utilization of the backend RAC database.

Note: The application latencies reported by the HLR application in case of the TimesTen tests, is from the transaction commits to the local TimesTen cache running on DRAM. The RAC database performance metrics reported during the TimesTen tests are from the TimesTen replication agent propagating the transaction updates and committing them to the backend RAC database.

Use Case 2: Large TimesTen cache on PMem

The goal of this use case was to study the performance and resource utilization impact of a large TimesTen cache when caching a relatively large subset of the backend RAC database. We also wanted to study the impact of using Intel Optane PMem as the main memory device for caching the backend RAC database on the TimesTen cache running on it.

Configuration overview

The following figure shows the two configurations—1800M subscribers Baseline and 1800M subscribers TimesTen cache + RAC—that we tested in Use Case 2.

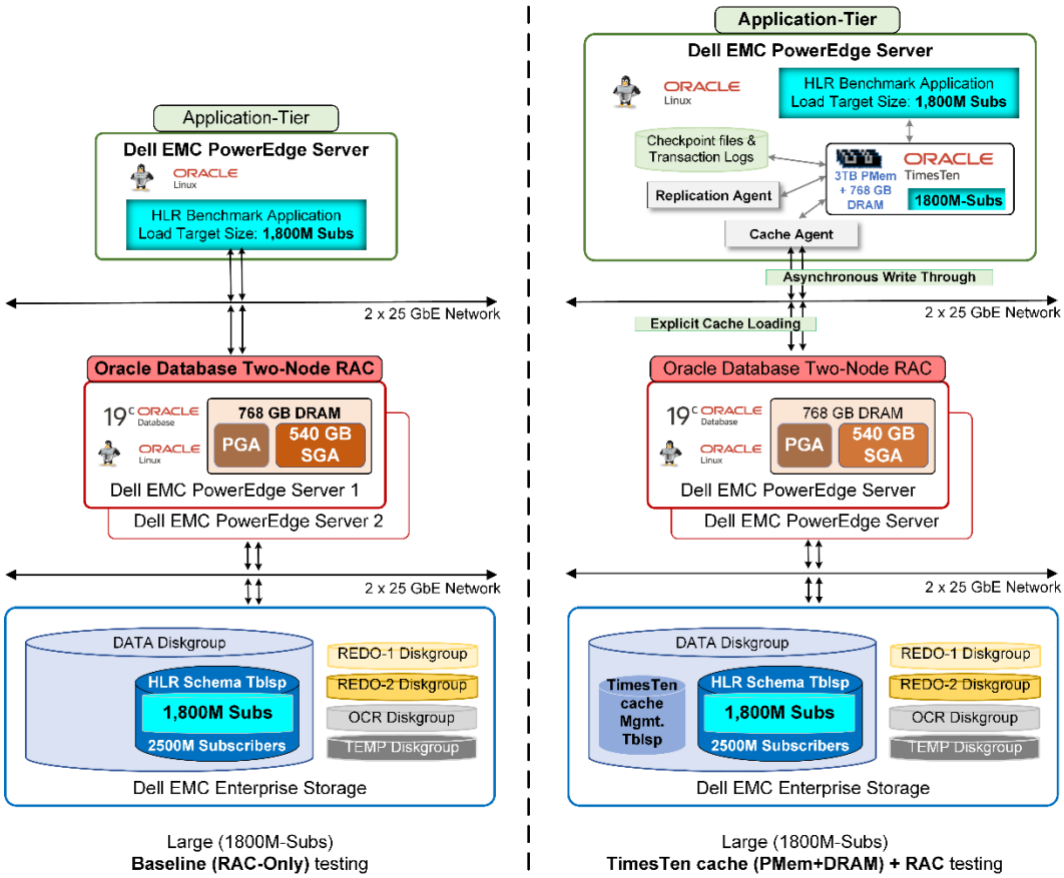

Figure 7. Use Case 2: Large (1800M subscribers) dataset configuration overview

In this use case, we added 3 TB capacity of PMem to run in Memory Mode to the 768 GB of DRAM that existed in the application (TimesTen) server. The PMem modules' 3 TB capacity was used as the main memory by the operating system and the application (TimesTen) while the 768 GB of DRAM that was used by the PMem as the front-end cache was transparent to the operating system and the application. Given this 3 TB main memory size, we could explicitly load a maximum of 1,800 million SUBSCRIBER rows (1,800M Subs) into the TimesTen cache. Therefore, we set the target workload data size

for both the test configurations in this use case to 1,800M subscribers. 1,800M subscribers is 72 percent of the total HLR schema size of 2,500 million SUBSCRIBER rows in the backend RAC database; therefore, this use case represents the scenario in which a relatively large subset of RAC database is cached in TimesTen.

All hardware and software configurations used for the HLR benchmark application, TimesTen, and the backend RAC database in Use Case 1 remained virtually identical for Use Case 2 with two exceptions: We increased the RAC instance SGA to 540 GB, and we increased the HugePages size appropriately. For details on the RAC and TimesTen database configuration, refer to Appendix: Configuration details.

Test methodology

During the baseline tests, we ran the HLR benchmarking workload directly against the backend two-node RAC database setup with a target workload size of 1,800M subscribers. During each of the baseline tests, we ran seven workload iterations with increasing HLR application thread counts of 2, 4, 8, 16, 32, 48, and 64. We repeated the baseline tests three times and reported the averages of the three runs.

Note: During the baseline tests, the TimesTen database and cache were disabled on the application server.

During the TimesTen tests, we first explicitly loaded 1,800M subscribers into the TimesTen cache running on 3 TB of PMem and configured in Asynchronous Writethrough (AWT) caching mode. We then ran the HLR benchmarking workload against the TimesTen cache limiting the target throughput to the maximum TPS rate achieved against the RAC database.

Note: As described in section Test methodology overview the limited target TPS rate for the TimesTen tests were nearly the same as the highest TPS achieved by the baseline tests.

Performance results

The figure below shows the results of the baseline test generated by the HLR application for the workload target size of 1,800M subscribers that we ran directly against the RAC database. The TPS values are plotted relative to the TPS throughput achieved at workload thread count 2 iteration. All latency values are plotted relative to the `UpdateSubscriberData` (USD) transaction latency observed at workload thread count 2 iteration.

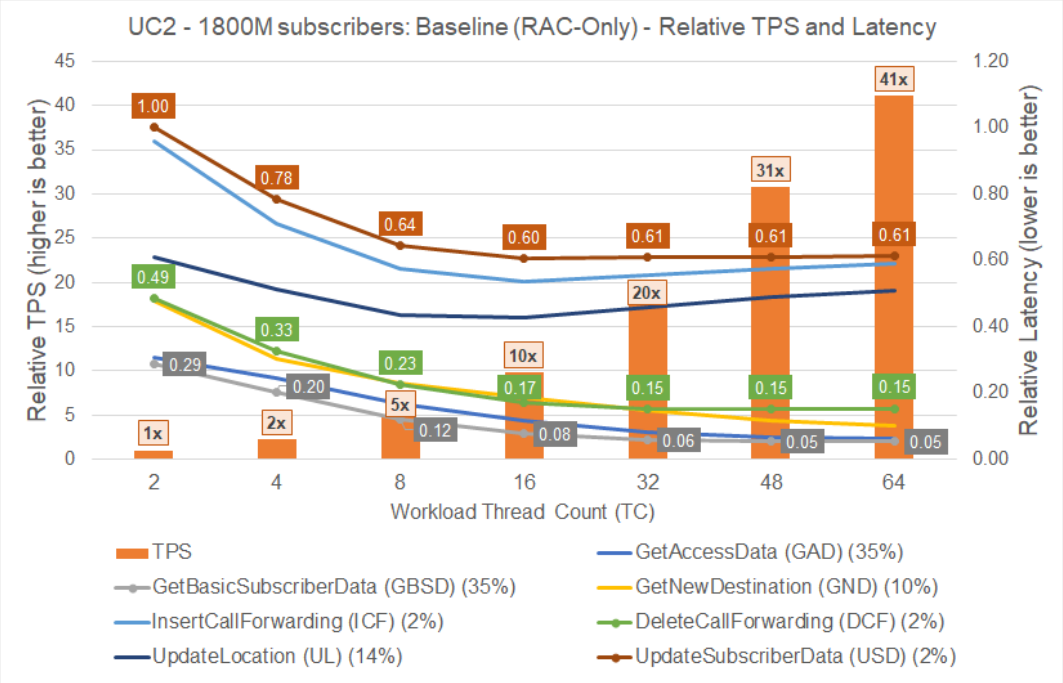

Figure 8. Use Case 2 - 1800M subscribers - Baseline (RAC-only) - Relative TPS and Latency performance graph

As shown in the figure above, the RAC database and backend Dell EMC infrastructure scaled well in terms of TPS, delivering 41x more TPS at the peak workload thread count of 64 compared to the TPS at the starting workload thread count of 2. However, the HLR transaction latencies began at a high level during the initial iterations and decreased as the test progressed. Latency levels stabilized beginning with the iteration running with workload thread count of 16 and remained at the same latency level until the end of the test. The latency level variation occurred because it took few initial iterations for the RAC buffer cache to get warmed up. During this time, more read and write I/O operations to the backend storage caused higher I/O latencies while processing the larger target dataset size of 1,800M subscribers.

The figure below compares the application latencies for each of the seven HLR transactions during the baseline tests and the TimesTen cache tests.

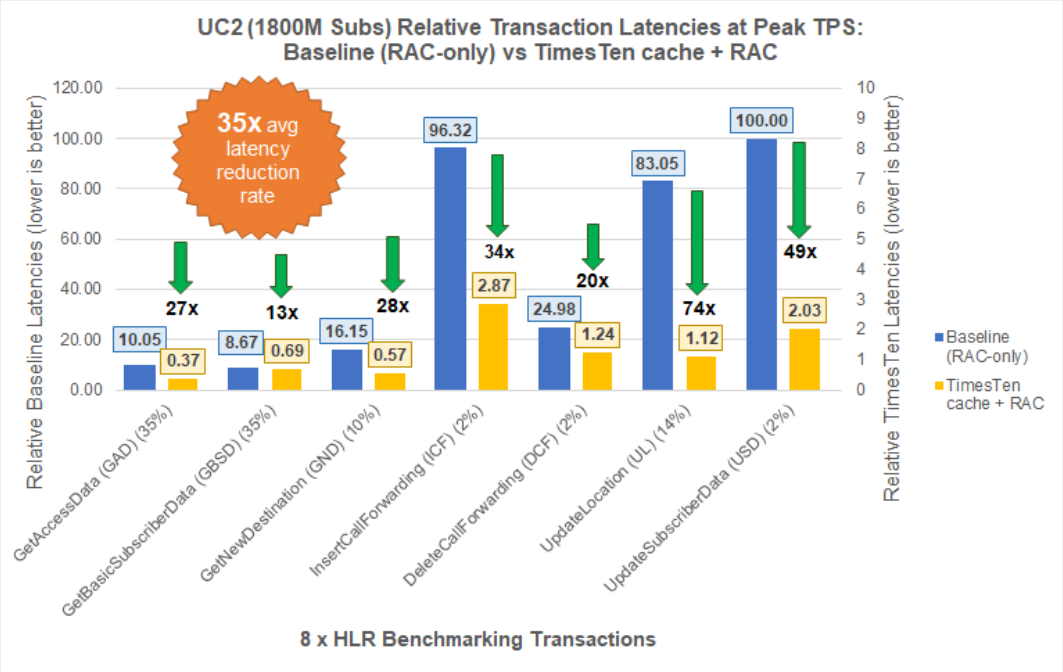

Figure 9. **Use Case 2 (1800M subscribers) - Average Latencies**: Baseline (RAC-only) vs TimesTen cache with RAC

The relative transaction latencies reported in the figure above were captured for both the baseline and the TimesTen tests when they both delivered nearly the same peak TPS throughput performance. All latency values in the figure above are relative to the baseline `UpdateSubscriberData` (USD) transaction latency. As shown in the figure above, in Use Case 2, in which we tested with a relatively large dataset size (1,800M subscribers), the TimesTen cache could deliver the same throughput as the RAC database but with an average of 35 times better transaction response time across the seven different mixed types of query and DML transactions.

Note: As mentioned in Use Case 1, it is important to note that the latencies reported for the TimesTen tests in Use Case 2 are from the transaction commits to the local TimesTen cache running on 3 TB of PMem in Memory Mode (with 768 GB of DRAM as front-end cache to PMem).

An additional purpose of this use case was to determine how a TimesTen cache impacts the resource utilization of the backend RAC database and its infrastructure.

The following figure shows the CPU utilization of RAC node2 captured using the Oracle OSWatcher utility during the duration of both the baseline and the TimesTen tests.

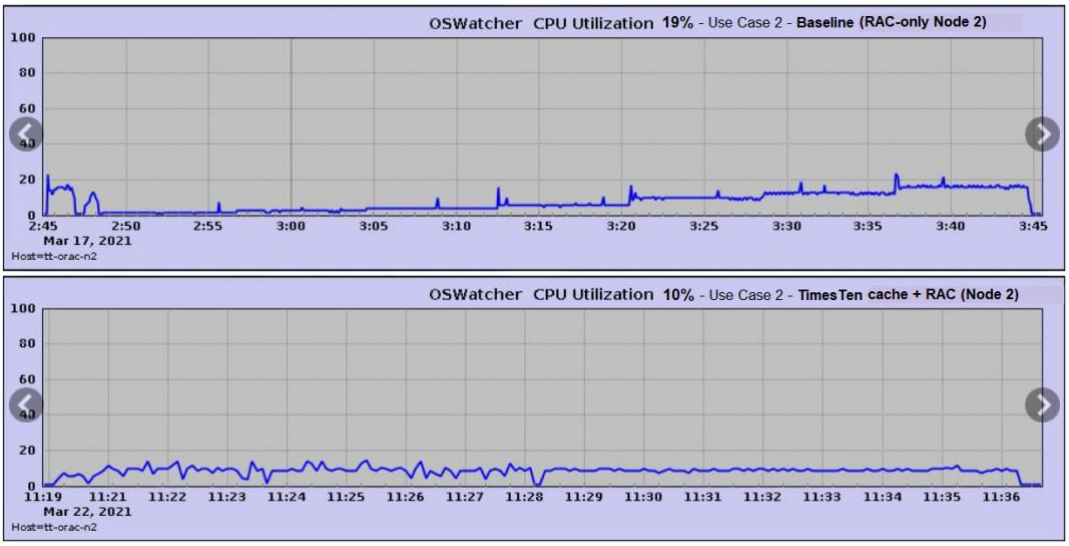

Figure 10. RAC node2 CPU utilization - Use Case 2 - Baseline vs TimesTen cache tests

As shown in the figure, the RAC node2's CPU utilization was around 10 percent (bottom graph) throughout the duration of the TimesTen tests as compared to roughly 19 percent for the baseline tests (top graph) during the peak workload thread count of 64. Therefore, the CPU utilization of the RAC node was reduced—a delta of about 9 percent in our particular test setup—during TimesTen tests compared to the baseline tests during their peak TPS performance.

Similarly, when we compared the 'Top 10 Foreground Events by Total Wait Time' from the AWR reports of the two test cases—baseline vs TimesTen—we observed that during the TimesTen cache tests the 'log file sync' wait events that were observed during the baseline tests were nearly eliminated (61.2% DB Time in baseline vs 2.8% DB Time in TimesTen tests), as shown in the comparative figure below.

Top 10 Foreground Events by Total Wait Time Use Case 2 - **Baseline** - Oracle RAC-only

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
log file sync	27,684,776	18.8K	678.41us	61.2	Commit
DB CPU		9760.9		31.8	
db file sequential read	11,197,892	4236.6	378.33us	13.8	User I/O
SQL*Net message to client	1.9E+08	49.5	256.44ns	.2	Network
cursor: pin S	26,665	28.2	1.06ms	.1	Concurrency
SQL*Net break/reset to client	465,995	17.2	37.00us	.1	Application

Top 10 Foreground Events by Total Wait Time Use Case 2 - **TimesTen Cache +** Oracle RAC

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
DB CPU		2532.5		50.0	
db file sequential read	6,018,418	2439.2	405.29us	48.2	User I/O
log file sync	304,624	142.8	468.76us	2.8	Commit
gc current block busy	517,246	93.2	180.13us	1.8	Cluster
library cache: mutex X	84,171	46.3	549.89us	.9	Concurrency
gc current block 2-way	123,250	22.2	180.32us	.4	Cluster

Figure 11. Foreground Wait Events - Use Case 2 - Baseline vs TimesTen cache tests

Also, the RAC database's CPU consumption time (DB CPU) improved during the TimesTen tests (DB CPU = 50.0% DB time) when compared to its CPU consumption time during the baseline tests (DB CPU = 31.8% DB time)—a delta of 18 percent. This is an

indication that the database spent more time processing the queries than waiting on other resources or events. This finding shows that deploying a TimesTen cache helps to increase the backend RAC database's infrastructure usage efficiency.

Conclusion

From the testing results of the two use cases, we observed that the two-node RAC cluster, configured as the baseline setup on Dell EMC infrastructure, scaled well in terms of TPS throughput performance when tested with an increasing HLR OLTP benchmarking workload. However, during the baseline RAC-only tests, some performance impact was observed at higher throughputs due to increased latency for DML operations caused by an increase in 'log file sync' wait events.

When we deployed TimesTen on a Dell EMC PowerEdge server in front of the baseline RAC setup, the TimesTen cache was not only able to offload query and DML transactions from the backend RAC database, but it was also able to deliver the same TPS throughput with up to 37 times improvement in transaction response times. Note that these response times were delivered with the transactions initially being persisted to the local TimesTen server's memory (DRAM and/or PMem) and disks before being asynchronously propagated to the RAC database. During the TimesTen tests, it was also observed that the backend RAC database node's CPU utilization went down during the two use cases. The 'log file sync' wait events observed during the baseline RAC testing were significantly reduced. During the TimesTen tests, the RAC database resource utilization went up with the DB CPU (as reported by AWR) delta values increasing by 35 and 18 percent during Use Case 1 and 2, respectively, as compared to the baseline RAC tests. This testing shows that TimesTen deployed as a cache helps optimize the utilization of the backend database hardware resources.

Therefore, in environments where some RAC database queries or DML transactions are saturating the infrastructure utilization and impeding the overall performance, customers can take advantage of deploying a TimesTen cache on PowerEdge servers to offload those transactions. This will help to accelerate the overall database response times and to optimize the utilization of the Dell EMC infrastructure, thus yielding a better return on investment.

We also demonstrated that using Intel PMem, in conjunction with DRAM modules as front-end cache, can help to provide increased capacity for TimesTen. This enables a much larger volume of data to be cached within a single PowerEdge server. PMem can not only provide greater capacity than DRAM-only configurations but Use Case 2 results showed that it can do so without compromising the overall database performance. During their initial purchasing and capacity planning, customers can definitely consider using Intel PMem supported by PowerEdge servers for their TimesTen memory needs.

Note: The replication lags, latencies, and the maximum throughput or TPS rate we observed in this study are strictly applicable to our setup in the lab and are NOT indicative of the maximum performance capability of the individual hardware and software products, including the schema, dataset, and the workload used in the solution. Performance results will vary based on the deployed environment.

References

Dell EMC documentation

The following Dell EMC documentation provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell EMC representative.

- Dell Technologies Ready Solutions for Oracle
- Dell EMC PowerEdge Servers
- Dell EMC Data Storage
- Dell EMC Networking
- Dell EMC PowerFlex - software-defined storage solutions
- *Dell EMC Intel Optane PMem User's Guide*

Oracle documentation

The following Oracle documentation provides additional and relevant information.

- *Oracle TimesTen Application-Tier Database Cache Overview*
- *TimesTen Application-Tier Database Cache User's Guide*
- *Oracle Database 19c Real Application Clusters Installation Guide for Linux and UNIX*

Intel documentation

The following Intel documentation provides additional and relevant information.

- *Intel Optane Persistent Memory - Start Up Guide*

Appendix: Configuration details

Hardware and software configuration details

Table 3. One Oracle TimesTen cache server details

Component	Description
Server	1 x Dell EMC PowerEdge R740xd
CPU	2 x Intel Xeon Platinum 8280L 2.7 GHz, 28C/56T
Memory	12 x 64 GB 2666 MHz RDIMMs [768 GB] *(Cache to PMem in UC2)*
	12 x 256 GB Intel Optane PMem [3 TB] *(UC2 only)*
rNDC/Integrated NIC	1 x Broadcom DP 25 GbE rNDC
NIC (PCIe Slot 1)	1 x Broadcom BCM57414 DP 25 GbE Adapter, Full Height
NIC (PCIe Slot 2)	1 x Broadcom BCM5720 1 GbE Adapter, Full Height
Local Disks Controller	PERC H740p mini (Embedded)
Local Disks Drives	[OS] 4 TB = 4 x 2 TB (RAID10) SAS-SSDs [TimesTen Logs] 14 TB = 16 x (RAID10) 1.8 TB SAS-SSDs
Operating system	Oracle Linux 8.3 running UEK R6 (5.4.17-2036.100.6.1.el8uek.x86_64)

Table 4. Two Oracle RAC database server details

Component	Description
Server	2 x Dell EMC PowerFlex (R640) Compute-only nodes
Components per PowerFlex Compute-only node	
CPU	2 x Intel® Xeon® Gold 6254 3.1 GHz, 18C/36T
Memory	12 x 64 GB 2666 MHz RDIMMs [768 GB]
rNDC/Integrated NIC	Mellanox 25 GbE 2P ConnectX-4 LX rNDC
NIC (PCIe Slot 1)	Broadcom 1 GbE DP adapter LP
NIC (PCIe Slot 2)	Mellanox 25 GbE 2P ConnectX-4 LX SFP Low Profile Adapter
NIC (PCIe Slot 3)	Mellanox 25 GbE 2P ConnectX-4 LX SFP Low Profile Adapter
Local Disks Controller	H730P mini (embedded)
Local Disks Drives	(OS) 2 x (RAID1) 1.8 TB SAS-SSDs
Operating system	Oracle Linux 7.9 running UEK R5 (4.14.35-2025.402.2.1.el7uek.x86_64)
PowerFlex SDC rpm	EMC-ScaleIO-sdc-3.5-1000.175.el7.x86_64.rpm

Table 5. Dell EMC shared data storage details - PowerFlex HCI (storage) nodes

Component	Description
Servers	4 x Dell EMC PowerFlex (R640) HCI nodes

Component	Description
Components per PowerFlex HCI node	
CPU	2 x Intel® Xeon® Gold 6254 3.1 GHz, 18C/36T
Memory	12 x 32 GB 2666 MHz RDIMMs [384 GB]
rNDC/Integrated NIC	QP iNDC: Intel 2P X710 (10 GbE) + Intel 2P i350 (1 GbE)
NIC (PCIe Slot 2)	Mellanox 25 GbE 2P ConnectX-4 LX SFP Low Profile Adapter
NIC (PCIe Slot 3)	Mellanox 25 GbE 2P ConnectX-4 LX SFP Low Profile Adapter
Local Disks Controller	(Operating system) BOSS-S1 (DATA) HBA330 mini (embedded)
Local Disks Drives	(Operating system) 2 x (RAID1) 240 GB M.2 SATA-SSDs (DATA) 10 x 1.8 TB SAS-SSDs
Host operating system	VMware ESXI 6.7.0, 16316930
PowerFlex HCI operating system	RCM 3.5.3.1

Table 6. Top-of-rack (ToR) switch details used for data traffic

Component	Description
ToR Switches	2 x Cisco Nexus 9000
Operating system/Firmware	7.0(3)I7(3)

Operating system settings

Oracle RAC nodes

We applied the following operating system settings and best practices:

- We installed the required RPMs and set the operating system and kernel parameters using:
  - `oracle-database-preinstall-19c-1.0-2.el7.x86_64.rpm`
  - `/etc/sysctl.d/99-powerflex-emc-sysctl.conf`
- We set the I/O Schedule to `noop` (default)
- We stopped and disabled the Linux system tuning service `tuned`.
- We set the HugePages size based on the Oracle SGA size set during the two use cases as follows:
  - Use Case 1: `sga_max_size` = 384 GB; `vm.nr_hugepages` = 196623
  - Use Case 2: `sga_max_size` = 540 GB; `vm.nr_hugepages` = 276494
- We created, on each Oracle ASM disk, a single GPT partition spanning the entire volume with 1 MB offset alignment, using the following commands:
  - `$> parted -s /dev/sciniX mklabel gpt`
  - `$> parted -s /dev/sciniX mkpart primary 2048s 100%`

- We set Oracle ASM disk ownership and permissions in the UDEV rules file `/etc/udev/rules.d/66-oracle-asmdevices.rules`. Below is an example of the UDEV rule set for one ASM disk (PowerFlex volume):

```
KERNEL=="scini*", SUBSYSTEM=="block",
PROGRAM="/opt/emc/scaleio/sdc/bin/drv_cfg --query_block_device_id --
block_device /dev/%k", RESULT=="5506082f2c48a3ef-
c62408e300000008",SYMLINK+="oraudev/disks/OCR1", OWNER="grid",
GROUP="asmadmin", MODE="0660"
```

Note: `/dev/oracleasm/disks` PATH is reserved for ASMLib. Setting this path in UDEV rule `SYMLINK+="oracleasm/disks/"` will cause the Oracle Grid installation to fail.

Oracle TimesTen cache server

We applied the following operating system settings and best practices:

- Kernel parameter settings:
  - `kernel.shmall = 1073741824`
  - `kernel.shmmax = 4398046511104`
  - `kernel.shmmni = 4096`
  - `kernel.sem = 250 32000 100 128`
  - `fs.file-max = 6815744`
  - `vm.overcommit_memory = 2`
  - `vm.overcommit_ratio = 100`
- We set HugePages size based on the main memory capacity available to the TimesTen cache during the two use cases as follows:
  - Use Case 1: DRAM-only = 768 GB; `vm.nr_hugepages` = 350000
  - Use Case 2: PMem = 3 TB; `vm.nr_hugepages` = 1433600

The following table shows the Virtual Disks (VD) that were created using PERC H740p:

Table 7. TimesTen server Virtual Disk (VD) settings

Purpose	Disk size	RAID	Logical selector size	Stripe element size	Write cache policy	Read cache policy	Disk cache
Operating system	4 x 2 TB	RAID 10	512 B*	256 KB*	Write Back*	Read Ahead*	Default* (Enabled)
TimesTen Checkpoint files and logs	16 x 1.8 TB	RAID 10	512 B*	256 KB*	Write Through**	No Read Ahead**	Default* (Enabled)

*Default property set by H740p
**Set as part of enabling FastPath feature in H740p. FastPath improves application performance by delivering high I/O per second (IOPS) for SSDs.

- We created a single GPT partition and XFS filesystem on the TimesTen virtual disk and mounted it on `/ttdb`. The following example shows the `/etc/fstab` entry for the TimesTen checkpoint files and logs mount point (`/ttdb`):

```
# cat /etc/fstab
/dev/mapper/ol-root     /                   xfs     defaults        0 0
UUID=3acfd0b0-e815-4bb1-83ff-9dff684b8dcc   /boot    xfs     defaults        0 0
UUID=4BC6-7AFC          /boot/efi            vfat    umask=0077,shortname=winnt 0 2
/dev/mapper/ol-home     /home                xfs     defaults        0 0
/dev/mapper/ol-swap     none                 swap    defaults        0 0
/dev/sdb1               /ttdb                xfs     defaults        0 0
```

Dell EMC storage volumes and settings

The following table shows the PowerFlex HCI storage nodes settings and the shared volumes that we created on it for RAC database usage:

Table 8. Dell EMC PowerFlex HCI storage nodes and volume settings

Storage-level settings			
Storage Pool	Capacity: 69.82 TB	Layout: Medium Granularity	
SDS	Performance Profile: High		
Storage volumes design			
Oracle purpose	**Size (GB)**	**Type**	**Compression**
DATA	8 x 1 TB = 8 TB	Thin	No
OCR / VD	3 x 48 GB = 144 GB	Thin	No
REDO1	2 x 48 GB = 96 GB	Thin	No
REDO2	2 x 48 GB = 96 GB	Thin	No
TEMP	6 x 512 GB = 3 TB	Thin	No

Oracle Database configuration

GI and Database patches and release updates

In this study, we tested with Oracle Grid Infrastructure 19.7 and Oracle RAC 19.7 Release Updates. The following table provides the details of the base images, patches, and the release updates (RU) that we applied:

Table 9. Oracle GI 19c and Oracle RAC 19c software stack versions

Stack name	**Patch number**	**Purpose**	**Oracle Support Document ID**
Oracle GI	19.3.0.0 base	Oracle GI 19c base software	
	6880880	OPatch Upgrade	274526.1
	30189609	Passwordless SSH fix	30189609.8
	30899722	Oracle GI 19.7 Release Update	30899722.8
Oracle RAC	19.3.0.0 base	Oracle RAC 19c base software	
	6880880	OPatch Upgrade	274526.1
	30899722	Oracle RAC 19.7 Release Update	30899722.8
	30805684	Oracle JavaVM Component RU 19.7	30805684.8

ASM disk groups

The following table shows the Oracle ASM disk groups (DG) that we created using our recommended best practices.

Table 10. Oracle ASM disk group (DG) settings

DG type	DG name	DG ASM disks	Total capacity	Striping	AU size
OCR_VD	+OCR_DG	3 x 48 GB	144 GB	Coarse	4 MB
DATA	+DATA_DG	8 x 1 TB	8 TB	Coarse	4 MB
REDO	+REDO1_DG	2 x 48 GB	96 GB	*Fine*	4 MB
REDO	+REDO2_DG	2 x 48 GB	96 GB	*Fine*	4 MB
TEMP	+TEMP_DG	6 x 512 GB	3 TB	*Fine*	4 MB

REDO log groups design

The following table shows the Oracle REDO log groups that we created according to our recommended best practices.

Table 11. Oracle REDO log groups design

REDO log group #	Disk group location	REDO log size	Thread (instance) number
1	+REDO1_DG	15 GB	1
2	+REDO2_DG	15 GB	1
3	+REDO1_DG	15 GB	1
4	+REDO2_DG	15 GB	1
5	+REDO1_DG	15 GB	2
6	+REDO2_DG	15 GB	2
7	+REDO1_DG	15 GB	2
8	+REDO2_DG	15 GB	2

Oracle TimesTen Database settings

We used the TimesTen In-Memory Database software version 18.1.4.9.0 in this study.

The TimesTen database settings were as follows:

```
[ODBC Data Sources]
sampledb=TimesTen 18.1 Driver
sampledbcs=TimesTen 18.1 Client Driver

[sampledb]
DataStore=/ttdb/db/sampledb
PermSize=2818000
# For 400M case PermSize=620000
TempSize=16384
LogBufMB=32768
```

```
LogFileSize=32768
ReplicationParallelism=8
LogBufParallelism=16
CacheAWTParallelism=31
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
PrivateCommands=1
CkptReadThreads=4
ReplicationParallelismBufferMB=1024
OracleNetServiceName=TTORACDB2
WaitForConnect=0
RecoveryThreads=16
IncludeInCore=1

[sampledbcs]
TTC_SERVER_DSN=SAMPLEDB
TTC_SERVER=r740xd-ttcache/6625
ConnectionCharacterSet=AL32UTF8
```

We determined the optimal values for some of these parameters, such as `LogBufParallelism`, `ReplicationParallelism` and `CacheAWTParallelism`, during the initial trial-and-error tuning runs.

HLR benchmarking usage and application settings

The HLR benchmark application is a Java application that interacts with the database using JDBC. The application is highly tuned to deliver optimal performance in accordance with database programming best practices. It also implements comprehensive performance measurements and reports a wide variety of metrics such as throughput (TPS), minimum, maximum, average, and 90th percentile transaction response times.

The online help for the application is as follows:

```
Usage:

java HLR [options]

     -c[lient]            connect in client mode (TimesTen only)
     -mysql               use MySQL rather than TimesTen
     -ora[cle]            use Oracle rather than TimesTen
     -sub[scribers] n     number of prepopulated subscribers
     -minsub[scriber] n   minimum subscriber id to target
     -maxsub[scriber] n   maximum subscriber id to target
     -h[ost] n            host for MySQL or Oracle}
     -p[ort] n            port number for MySQL or Oracle}
     -s[id] n             SID for Oracle
     -uid n               user name for TimesTen or Oracle
     -pwd n               password for TimesTen or Oracle
     -r[ate] n            number of ops/sec to run
     -t[hreads] n         number of threads to use
     -ti[me] n            time limit; run for n seconds
     -d[etail]            print per-thread measurements
```

```
    -tr[ace]              turn on jdbc tracing
    dsn=<mydsn>           specifies DSN for target database (TimesTen only)
```

Examples:

(TimesTen) `java HLR -o 1000000 -ti 300 -t 8 -uid scott -pwd tiger dsn=sampledb`

(MySQL) `java HLR -mysql -host host1 -port 12345 -t 8`

(Oracle) `java HLR -ora -host host1 -p 1521 -s orcl -uid scott -pwd tiger -t 8`

The actual command lines and parameters used for each scenario in this study are shown below. We varied the number of threads for the two use cases, as described in Use cases, test methodologies, and results.

Here is an example of the HLR command that we ran during the baseline 400M subscribers RAC-only tests:

```
java HLR -oracle -host tt-orac-n1 -port 1521
         -sid ttoracdb.domain.com -uid appuser -pwd appuser
         -sub 2500000000 -minsub 1 -maxsub 200000000
         -threads 32 -rup 120 -rdown 60 -time 300
         -csv ora400_1.csv

java HLR -oracle -host tt-orac-n2 -port 1521
         -sid ttoracdb.domain.com -uid appuser -pwd appuser
         -sub 2500000000 -minsub 1250000001 -maxsub 1450000000
         -threads 32 -rup 120 -rdown 60 -time 300
         -csv ora400_2.csv
```

Here is an example of the HLR command that we ran during the baseline 1,800M Subs RAC-only tests:

```
java HLR -oracle -host tt-orac-n1 -port 1521
         -sid ttoracdb.domain.com -uid appuser -pwd appuser
         -sub 2500000000 -minsub 1 -maxsub 900000000
         -threads 1 -rup 120 -rdown 60 -time 300
         -csv ora1800_1.csv

java HLR -oracle -host tt-orac-n2 -port 1521
         -sid ttoracdb.domain.com -uid appuser -pwd appuser
         -sub 2500000000 -minsub 1250000001 -maxsub 2150000000
         -threads 1 -rup 120 -rdown 60 -time 300
         -csv ora1800_2.csv
```

Here is an example of HLR command that we ran during the 400M subscribers TimesTen tests:

```
java HLR -rate 220000 -sub 400000000 -uid appuser -pwd appuser
         -threads 4 -rup 120 -rdown 60 -time 300 -csva tt400.csv
         dsn=sampledb
```

Here is an example of the HLR command that we ran during the 1,800M subscribers TimesTen tests:

```
java HLR -rate 220000 -sub 1800000000 -uid appuser -pwd appuser
         -threads 4 -rup 120 -rdown 60 -time 300 -csva tt1800.csv
         dsn=sampledb
```