

# Oracle TimesTen In-Memory Database High Availability

Using TimesTen Replication to deliver High Availability and Disaster Recovery


ORACLE WHITE PAPER | OCTOBER 2015





## Table of Contents

Introduction	1
Oracle TimesTen In-Memory Database Overview	1
High Availability Overview	1
What is High Availability?	1
High Availability versus Disaster Recovery	2
Data Replication Concepts and Terminology	2
Transactional versus Snapshot Replication	3
Continuous versus Timed Replication	3
Replication element	3
Single master versus Multi master	3
Asynchronous versus Synchronous Replication	4
Single Stream versus Parallel Streams	4
Replication Scheme or Topology	5
TimesTen Data Replication	5
Active/Standby Pair Replication Functionality	5
Read-only Subscribers	6
Failure handling and recovery	7
Parallel Replication	8
A/S Pair and Application-Tier Database Cache	8
Failure handling and recovery	9
A/S Pair with Cache and Read-only Subscribers	9
Automatic Client Failover	9



Online Software Upgrade	10
Clusterware Management of A/S Pair Replication	10
A/S Pair Replication Advanced Features	11
Synchronous (RETURN TWOSAFE) Replication	11
Excluding Database Objects from Replication	11
Data Compression	11
Parallel Replication – No Commit Order Enforcement Optimization	12
Disaster Recovery	12
A/S Pair Replication Performance	12
Example Use Cases	15
Location Services for Mobile Networks	15
VOIP Application Server	16
Conclusion	19
Appendix A – Example Benchmark Details	20
Test Systems	20
Database Parameters (SYS.ODBC.INI settings)	20
TPTBM Table and Workload	20
Session Management Table and Workload	21



## Introduction

The Oracle TimesTen In-Memory Database family provides extreme levels of performance for today's high volume mission critical applications. As well as high performance, most mission critical applications also require high levels of availability and/or data protection; TimesTen provides rich and robust functionality to meet these requirements. This paper describes the sophisticated High Availability and Disaster Recovery capabilities that are an intrinsic part of TimesTen and describes how they may be utilized in order to build applications, systems and services that not only deliver extreme performance but also the highest levels of availability and data protection.

The information in this paper is based on the TimesTen 11.2.2 release.

## Oracle TimesTen In-Memory Database Overview

The Oracle TimesTen In-Memory Database (TimesTen) is a Relational In-Memory Database (IMDB). When it was launched in 1998 it was the first commercially available in-memory RDBMS. TimesTen has been designed from the ground up for high performance - low, consistent response times and high throughput, high concurrency and scalability - as well as flexibility, robustness and data protection through persistence and high availability.

TimesTen supports standard SQL and standard database APIs making high performance database access available to a wide variety of applications without the need for special tooling or learning new development skills. In addition TimesTen is designed to require little DBA oversight making it a good fit for embedding within ISV or OEM applications and solutions. TimesTen can be deployed as:

- » A self contained standalone database, with or without High Availability – **Oracle TimesTen In-Memory Database**
- » A high performance SQL based relational cache for the Oracle Database, with or without High Availability – **Oracle TimesTen Application-Tier Database Cache**

The TimesTen Application-Tier Database Cache includes all of the functionality of the TimesTen In-Memory Database so it is possible to have both cache and non-cache tables residing in the same TimesTen database concurrently.

This paper focuses on the features and functionalities provided by TimesTen that enable high availability. Except where indicated otherwise, all aspects of TimesTen's High Availability and Disaster Recovery capabilities apply equally to both **Oracle TimesTen In-Memory Database** and **Oracle TimesTen Application-Tier Database Cache**.


For more information on TimesTen visit the [TimesTen product center](#) on [Oracle Technology Network](#).

## High Availability Overview

What is High Availability?

The subject of system availability, and high availability (HA), is a broad and interesting one but it is far beyond the scope of this paper to give it a comprehensive treatment. Here we focus on a few key points as the subject applies to databases.

Availability is a characteristic of a system describing the duration for which the system is operational within a defined time period (typically one year –  $365 \times 24 = 8760$  hours or 525,600 minutes). So if a system is described as having



99.999% availability then this means that in any one year the system will be operational for at least 525,594 minutes and 45 seconds ( $0.99999 \times 525,600$  minutes). Put another way, the maximum time that the system can be non-operational in any given year is 5 minutes and 15 seconds ( $525,600$  minutes – 525,594 minutes and 45 seconds ).

The degree of availability required for any given system is largely driven by business and operational considerations and is usually enshrined in a Service Level Agreement (SLA). There is no strict definition of high availability but it is commonly taken to mean any system with availability better than 99.9% (three nines) with the gold standard for high availability generally being considered to be five nines, or 99.999. In SLAs, availability may be defined as maximum permitted downtime per month or week rather than per year in which case 99.999% availability equates to a maximum downtime of 26 seconds per month or 6 seconds per week.

Availability, and the notion of high availability, may be applied both at a system level and at the level of individual components and services that go to make up the system. For example in a large enterprise application there may be many components; server hardware, databases, application servers, network infrastructure etc. The availability of each of these components and services will determine the overall availability of the system.

There are various ways in which a database service can be made highly available; some common techniques are shared-everything clusters, shared nothing clusters and data replication. A detailed description and discussion of these techniques and their pros and cons is beyond the scope of this paper but there is a great deal of information available in the literature for those who are interested.

### High Availability versus Disaster Recovery

A topic that is often related to high availability is that of Disaster Recovery (DR). Making a system (or a database) highly available means being able to continue operation, with all data accessible (and typically also updateable), in the face of the failure of a single component such as a server, a network switch, a disk drive etc. Disaster Recovery focuses on allowing business operations to continue (usually after some initial interruption to service) following a catastrophic failure such as the failure of an entire data center.

From a database perspective DR typically means:


- » Maintaining a complete copy of the primary operational database at one or more remote locations and keeping these copies synchronized with the primary database
- » Providing mechanisms and resources that enable one of the remote copies to be rapidly brought online as a fully functional primary database usable by applications in the event that some 'disaster' befalls the site hosting the primary database
- » Providing mechanisms and resources to revert the database back to the original configuration, without any data loss, once the 'disaster' has been rectified

TimesTen's HA functionality may also be utilized to provide a DR capability if required.

### Data Replication Concepts and Terminology

TimesTen provides High Availability and Disaster Recovery by way of data replication, specifically real-time transactional data replication.

**Data replication** refers to features and mechanisms that enable multiple copies of data (ranging from an entire database to a subset of some tables) to be kept synchronized with each other. These copies are typically stored in different databases, hosted on different servers, possibly in different data centers and maybe located in different continents. Although data replication technology is often used to provide HA and/or DR, it also has a number of other uses (which we shall not be discussing in this paper).



Before we move on to a detailed examination of the HA and DR functionality that is available in TimesTen it will be helpful to define a few key terms and concepts that relate to data replication technology in general.

### Transactional versus Snapshot Replication

Transactional data replication works by capturing changes that occur at one copy of the data then propagating them to and applying them at each of the other copies. The changes are captured at the level of the individual database transactions executed by the application(s) and transactional boundaries and integrity are preserved throughout the replication pipeline ensuring that all copies of the data are transactionally consistent within themselves at all times. Typically, transactional replication operates continuously and in real-time or semi real-time.

Snapshot replication typically taking an occasional 'snapshot' of the data at one of the copies and then propagating this to and applying it at the other copies, usually with no specific guarantees of transactional consistency. Snapshot replication often involves copying the entire (replicated) data set each time rather than just deltas (changes) and is usually time based (once an hour, once a day etc.) rather than continuous.

TimesTen replication is **transactional**.

### Continuous versus Timed Replication

Continuous replication runs all the time and endeavors to propagate changes as quickly as possible after they have occurred at the source copy. Timed replication occurs based on some defined time interval.

Generally transactional replication is continuous while snapshot based replication is timed.

TimesTen replication is **continuous**.

### Replication element

A replication element (element) is the unit of data for which replication may be defined. Most products/technologies provide either table level or database level replication (or both). Some products may allow for replication of subsets of table rows and/or columns.

TimesTen replication supports both **database** level elements and **table** level elements though only database level elements are commonly used.


### Single master versus Multi master

In replication terminology, the master copy of a replication element is the copy at which application updates occur. Replication technologies may support 'single master' replication - whereby for each element one copy is always the master and the other copies are 'slaves' or 'subscribers' (they do not allow direct updates to the element by the application) – or they may support 'multi master' replication – all copies of an element are directly updateable by the application.

Multi master replication seems very attractive as it provides the greatest flexibility for system design and operation but in fact it brings with it a number of challenging issues:

- » How to reconcile conflicting updates that arrive at an element copy due to transactions executed at other copies of the element?
- » How to ensure data consistency across all element copies in the presence of replication conflicts?
- » How to ensure consistent recoverability after a failure since there is not necessarily any authoritative copy of any element?

There are mechanisms that can mitigate these issues for the specific case where there are just 2 copies of each element ( $N = 2$ ), and careful design of the application (making it 'replication aware') can also mitigate the issues to some degree. But there is still the need for tradeoffs even with  $N = 2$  and these mechanisms cannot generally solve



the problems adequately for cases where  $N > 3$ . As a result, multi-master replication is very challenging to deploy and operate successfully and reliably regardless of the product/technology used and in most cases single master replication is a superior choice.

TimesTen replication supports both **single master** and **multi master** replication but only single master is commonly used.

### Asynchronous versus Synchronous Replication

Replication of database changes typically involves the following steps:

1. Capture the changes at the master element (usually this happens after the application transaction that made the changes commits)
2. Propagate the changes to the subscriber element(s)
3. Apply the changes at the subscriber element(s)
4. Acknowledge completion back to the master element

With asynchronous replication, all these steps happen independently of the application transaction that made the changes; the application transaction executes and then commits, the commit call returns as soon as the changes are committed in the local database and then the replication steps all occur in the background.

With synchronous replication, the application commit call blocks until all these steps have completed; it only returns control to the application once the transaction has been processed at some/all of the subscriber elements.

Asynchronous replication typically gives much better response time and throughput than synchronous replication. However, with asynchronous replication the master element (where the application updates are happening) is always 'ahead' of the subscribers.

If there is a failure and the application processing is moved to one of the subscribers then either some data loss must be accepted or some (complex) mechanism must be devised to retrospectively apply the delayed transactions after the failed (originally master) element has been recovered. Generally this latter option is not feasible and so use of asynchronous replication generally implies the acceptance of some degree of data loss if a master element fails and application processing is moved to a (promoted) subscriber.

Synchronous replication avoids these issues at the cost of increased response time and lower throughput.


The choice between asynchronous and synchronous replication is usually driven by business and application requirements.

TimesTen replication provides both **asynchronous** and **synchronous** options with the **ability to control this at the individual transaction level if so desired**.

### Single Stream versus Parallel Streams

Normally a single replication 'pipeline' (capture, propagate, apply) is used between a master element and each of its subscriber elements. This ensures that operation and transaction ordering (commit ordering) is maintained for each replication flow but it also places a limit on the maximum throughput that can be achieved by replication. In many cases significantly higher replication throughput can be achieved by the use of multiple parallel replication pipelines between the master element and each subscriber element but this brings with it some complications:

- » How to ensure correct operation ordering when there are operation dependencies between transactions in different pipelines?
- » How to ensure correct commit ordering when multiple transaction streams are being applied in parallel at a subscriber?



Note that for many workloads strict commit ordering across replication streams is actually not required to ensure data consistency.

These are difficult challenges to overcome while still preserving the performance benefits of parallel replication.

TimesTen replication supports both **single stream** and **parallel stream** replication. The parallel stream implementation solves these problems by tracking inter-transaction operation and commit dependencies at the master element and encoding this information into the replicated transaction stream. This dependency information is then used by the apply mechanism at the subscribers to ensure correctness while transactions are being applied in parallel. In addition, TimesTen provides an option to disable the enforcement of commit ordering during apply in order to realize further substantial performance uplift for workloads that do not require strict commit ordering.

### Replication Scheme or Topology

The replication scheme, sometimes referred to as the replication topology, is the complete definition of the replicated environment; machines, databases, elements, network links, single/multi master, async/sync etc. All replication products provide mechanisms to define and deploy the scheme, to monitor and control its operation and to recover from various kinds of failures that can occur.

TimesTen provides **comprehensive tools (utilities and APIs)** to define, deploy and administer replicated environments.

## TimesTen Data Replication

TimesTen has supported data replication almost since its first release and has a long track record of providing robust and high performance HA and DR capabilities.

TimesTen supports two distinct replication mechanisms; Active/Standby Pair Replication (A/S pair) and Classic Replication. Of these, A/S Pair is the more modern, and recommended, choice; Classic is primarily maintained in order to support existing customer deployments where migration to A/S Pair is not viable. Both mechanisms use TCP/IP (IPv4 or IPv6) as their transport mechanism. This paper focuses on Active/Standby Pair Replication.

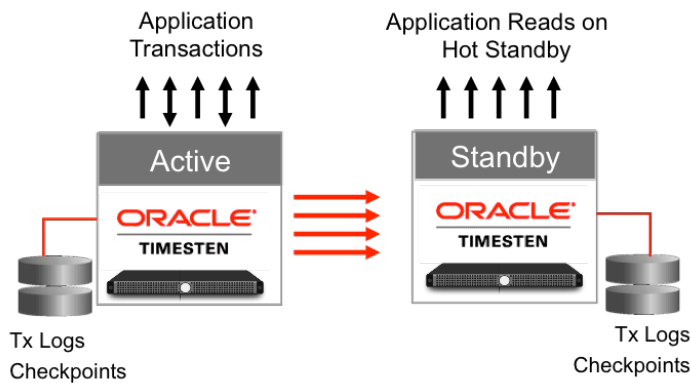
### Active/Standby Pair Replication Functionality

The core of an A/S Pair replication scheme consists of two databases deployed on different machines and connected by a LAN grade network. At any given moment in time, one of the databases will be designated the active database; this is the only database where applications can perform DML. The other database is designated the standby database; applications may connect to the standby and may execute queries there but they may not execute any DML (if they try they will get an error indicating that those operations are not permitted on the replication standby). See Figure 1.

A/S Pair replication is whole database replication (the replication element is the database) but it is possible to exclude specific database objects from replication (see the Advanced Features section later in this paper). All changes to the persistent data of replicated objects that occur at the active are replicated to the standby and applied there thus keeping the active and standby databases synchronized.

Replication captures changes by mining the TimesTen database transaction log stream at the active; with correct configuration the capture process occurs completely in memory with no need to read any data from disk.





**Figure 1: A TimesTen A/S Pair**

The default, and most commonly deployed, replication mode for A/S Pair is asynchronous (a synchronous option is also available - see the Advanced Features section later in this paper). In the event of some failure of the standby database, its host machine or the network between the active and the standby, applications can continue to execute transactions unimpeded at the active. The changes that occur while the standby is not accessible remain queued in the active database's transaction logs in memory and on disk. Once the failure has been repaired and the standby database is again up and accessible from the active then the accumulated changes will be propagated across and the databases will automatically re-synchronize, at which point the accumulated transaction logs will be freed.

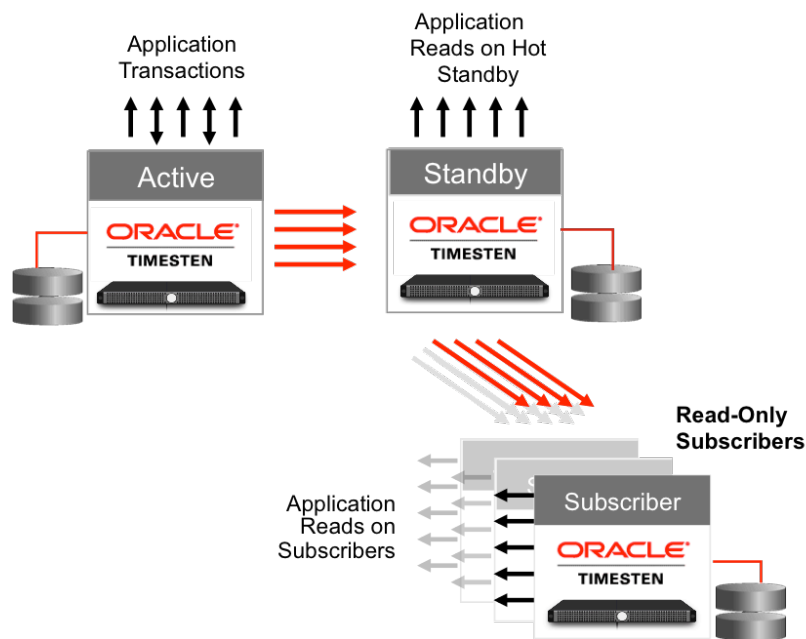
In the event of a prolonged outage the replication flow from active to standby can be set to a 'stopped' state either by an administrator command or automatically when a predefined threshold is reached. This prevents excessive accumulation of transaction log files at the active. When the standby system is again operational and accessible the standby database is resynchronized by being re-created using a network clone operation (called a 'duplicate' operation) from the active database.

In the event of a failure affecting the active database, a database failover can be initiated. The standby database can be rapidly (within milliseconds) promoted to the active role and applications can continue to operate against the promoted standby (including DML execution). When the failure has been rectified, the old active database can be recovered as the new standby.

TimesTen also provides a mechanism to perform a controlled (non failure) database role reversal (a switchover).

### Read-only Subscribers

In addition to the active and standby databases, a TimesTen A/S Pair replication scheme may also include one or more (up to a maximum of 127) read-only subscriber databases. Each of the subscribers is a complete database copy (apart from any excluded objects) and is maintained in sync by replication. As the name suggests, these copies are read-only; applications can run queries but any attempt to execute DML will return an error. See Figure 2.



**Figure 2: A TimesTen A/S Pair with read-only Subscribers**

Whereas the active and standby need to be connected by a LAN grade network, it is supported for one or more of the read-only subscriber databases to be connected to the A/S Pair over a lower capacity/higher latency network such as a WAN. This facilitates using read-only subscribers to implement a DR strategy for TimesTen (see the Advanced Features section for more details).

The other use for read-only subscribers is for local and/or remote reader farms; if the application requires huge capacity for queries but less so for insert/update/delete then this is a good way to provide read capacity at scale.

#### Failure handling and recovery

When read-only Subscriber(s) are configured it is the job of the Standby database to propagate changes to them. Transactional changes originate at the Active, are propagated to and applied at the Standby and then onward propagated to, and applied, at the read-only subscribers. Replication tracks, by way of control information exchanged between, and stored in, every copy which transactions have been sent to, applied at and acknowledged by every copy.

In the event that a read-only subscriber fails, changes destined for that subscriber are queued in the transaction logs on the Standby. When the subscriber has been recovered and is up and contactable again, the queued changes will propagate across automatically. If the subscriber will be down for a prolonged period then the same mechanisms are available for recovery as for the core A/S Pair.

In the event that the Active fails and the Standby is promoted to be active then the promoted Standby (now the Active) will continue to be the propagator of changes to the read-only subscribers. No data is lost and the interruption to replication processing is only as long as it takes to decide to promote the Standby and execute the necessary API calls / administrator commands to do so. When the failed Active is recovered (as the new Standby) and has fully re-synchronized with the Active, responsibility for propagating changes to the read-only subscribers is automatically and seamlessly passed back to the new Standby.

In the event that the Standby fails then as soon as the Active is aware of the failure it will take over the propagation of changes to the read-only subscribers. No data is lost and the interruption to replication processing is only as long as it takes to decide that the Standby has failed and to execute the necessary API calls / administrator commands to mark the Standby as failed. When the Standby has been recovered and has fully re-synchronized with the Active, responsibility for propagating changes to the read-only subscribers is automatically and seamlessly passed back to the Standby.

### Parallel Replication

By default TimesTen replication propagates changes from a source to a target using a single capture/propagate/apply pipeline (single stream replication). Although this can provide quite high performance, for write intensive workloads the throughput of a single replication stream may impose a limit on the overall processing capacity of the system. TimesTen also supports parallel replication in order to increase replication throughput and to leverage the processing power of modern multi-core systems and high bandwidth low-latency networks.

When configured for parallel replication there are multiple capture/propagate/apply pipelines between each master element and each subscriber element, and transactional changes are replicated through all these pipelines concurrently. In order to ensure correctness TimesTen detects and tracks any dependencies between the replicated transactions. This information is encoded as part of the replication stream and is used at the target when applying the transactions. This allows TimesTen to ensure that transactions are always applied in the correct order, even if they arrive via different replication streams.

### A/S Pair and Application-Tier Database Cache

If TimesTen is being used as a cache for the Oracle Database (TimesTen Application-Tier Database Cache) then certain types of cache table – READONLY with AUTOREFRESH and ASYNCHRONOUS WRITETHROUGH - can also be protected by TimesTen replication. See Figure 3.

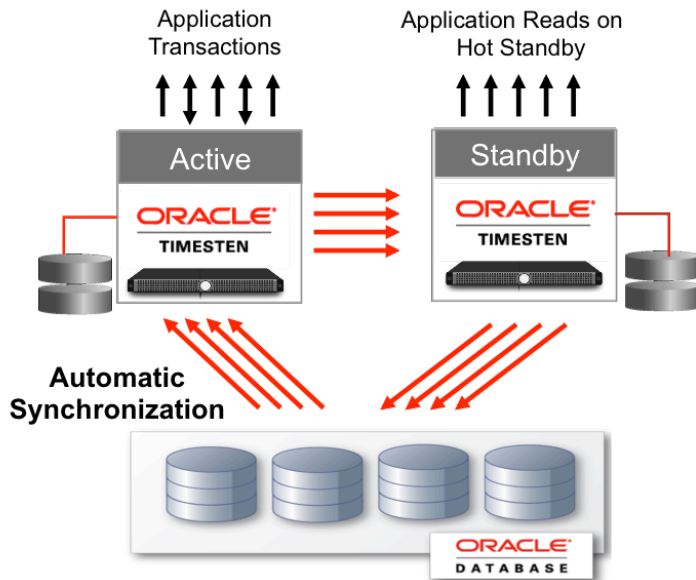



Figure 3: A TimesTen A/S Pair with Cache



In this configuration, changes to tables cached in READONLY AUTOREFRESH cache groups are refreshed from the Oracle database to the active TimesTen database, and from there replication propagates them to the standby and onwards to any defined subscribers. For tables cached in AWT cache groups, changes occurring at the active TimesTen database are propagated by replication to the standby TimesTen database and then onwards to the Oracle database and any defined subscribers.

### **Failure handling and recovery**

The integration between TimesTen replication and TimesTen cache functionality is seamless and automatic. The transfer of responsibility for cache AUTOREFRESH and AWT propagation in the event of any failure is handled seamlessly and automatically, with no data loss, in a manner analogous to that used for read-only subscriber propagation.

### **A/S Pair with Cache and Read-only Subscribers**

This is also a supported configuration. Cache tables and their contents are present as regular tables in the read-only subscribers.

### **Automatic Client Failover**

The TimesTen database APIs (ODBC, JDBC, OCI, Pro\*C, ODP.NET) support two connectivity modes, direct-linked and client/server.

In order to use direct-linked connectivity, the application must run in the same machine as the TimesTen database. Direct-linked connectivity provides a lightweight, non-IPC connectivity mechanism that provides the very highest levels of performance.

Client/server mode follows a more traditional architecture; the TimesTen client (application) communicates with a server process that acts as a proxy for all database operations. The client sends database requests to the server, which executes them on the client's behalf and then returns the results to the client. The server process is a specialized direct-linked application that runs on the same machine as the TimesTen database, while the client itself may be local or (more commonly) remote. Each database interaction requires one or more round trips between the client and the server.

Client/server supports various connectivity options. For remote clients TCP/IP networking (IPv4 and IPv6) is supported. For local clients (client and server co-located on the same machine) there is a choice of TCP/IP, Unix domain sockets or a proprietary shared memory based IPC mechanism.

When using client/server mode over TCP/IP to access a TimesTen database replicated using A/S Pair Replication, the TimesTen client can be configured to provide automatic client failover. A client connection configured for automatic failover always connects to the currently active database; if a database switchover occurs and the standby is promoted to the active role then client connections are migrated across to the new active. Only the connection is migrated; any in progress transactions are terminated, any open cursors are closed and any prepared statements are freed. After the connection failover has completed the application must re-prepare its statements and then restart any queries or transactions.

Applications can detect when a failover has occurred either by detecting and handling the specific error codes that are generated in a failover situation or (preferred) by registering a failover callback which will allow them to be proactively notified when a failover occurs so that they may take the necessary recovery actions.

Automatic client failover is implemented completely within the TimesTen client, server and replication mechanism and does not require the use of any external agent or virtual IP addresses.

## Online Software Upgrade

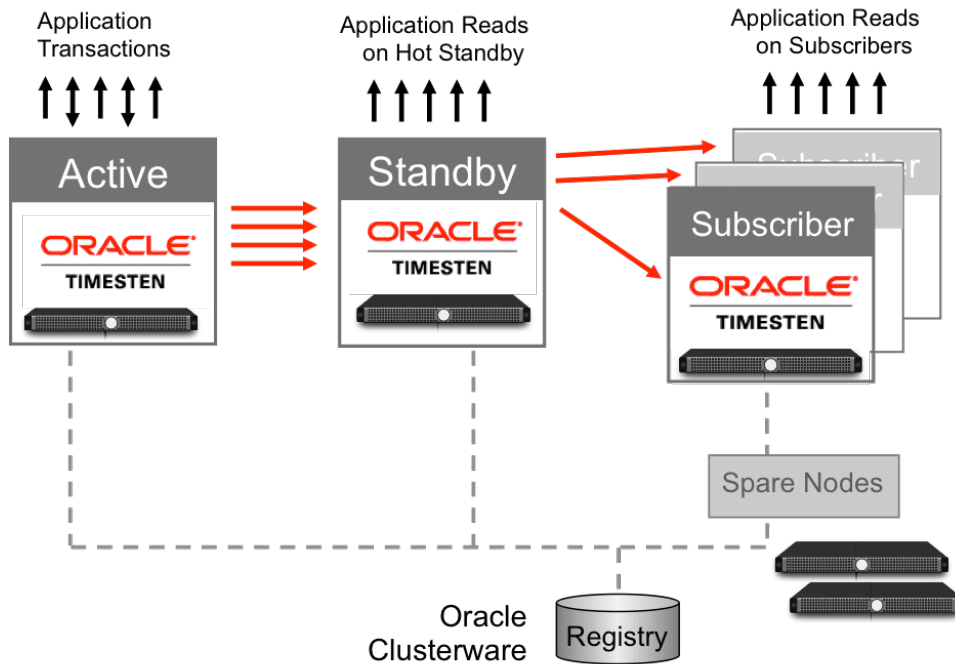
TimesTen replication supports interoperability across different software versions (but they must be the same hardware platform and OS). This provides an online (rolling) software upgrade capability for the TimesTen software.

In a replicated environment it is possible to update the software for each database individually while all the others remain online and functional. In this way it is possible to perform a software upgrade while the database continues to provide service to the applications.

## Clusterware Management of A/S Pair Replication

One of the challenges when operating a sophisticated database replication setup is monitoring for failures and ensuring that the correct failover and recovery actions are executed, as rapidly as possible, when a failure does occur in order to ensure maximum availability and minimal service interruption.

Oracle has sophisticated cluster management software, Oracle Clusterware, which provides the underlying infrastructure for the Oracle RAC cluster database. TimesTen has a deep integration with Oracle Clusterware and it can be used to provide a complete management solution for TimesTen A/S pair replication. See Figure 4. Although the Clusterware technology used is the same as that which underpins Oracle RAC, the Clusterware installation used for TimesTen is completely independent of that used for Oracle RAC in the database tier and indeed there is no requirement for Oracle RAC to be present.



**Figure 4: TimesTen A/S Pair with Clusterware**

The TimesTen Clusterware integration provides the following capabilities:

- » Support for TimesTen In-Memory Database and TimesTen Application-Tier Database Cache
- » Support for read-only subscribers in addition to the active and standby
- » Automated rollout of the replicated environment

- » Automated monitoring for and reaction to failures
- » Automatic database failover and recovery
- » Automated database backups (full and incremental)
- » Support for spare hardware that can be brought into play automatically
- » Application failover management for applications using direct-linked connectivity

Use of Clusterware is not mandatory; TimesTen provides all the necessary utilities and APIs to allow customers to implement their own HA management infrastructure or to integrate TimesTen into an existing HA infrastructure.

## A/S Pair Replication Advanced Features

### Synchronous (RETURN TWOSAFE) Replication

By default TimesTen replication operates in asynchronous mode. This provides excellent performance and a high degree of data protection. However, in the event of a failure of the active database and a subsequent database switchover, the use of asynchronous replication may expose the application to some degree of data loss. If data loss cannot be tolerated then you can choose to configure TimesTen replication in RETURN TWOSAFE mode.

RETURN TWOSAFE is a fully synchronous replication mode built on the foundations of the asynchronous replication transport. With RETURN TWOSAFE, the capture, propagate, apply and acknowledgement stages of replication occur while the application is blocked within its commit call. The commit call only returns to the application when the transaction has been safely applied and committed at both the Active and Standby databases (the commit happens first at the Standby) or when replication has determined that some problem has occurred (in which case an error is returned to the application). These commit calls are in-memory commits (logging to disk happens outside of the transaction execution path) and so response time is not impacted by the need to wait for disk I/O. This double memory commit protects the committed data from any single point of failure.

Use of RETURN TWOSAFE allows you to guarantee that there will be no data loss in the case of any single failure; the downside is that the performance of RETURN TWOSAFE is lower than that for asynchronous replication; throughput is lower and response times are higher.

TimesTen is very flexible however, rather than simply turning on RETURN TWOSAFE for all transactions (which is of course possible) you can opt to enable this feature to be used 'on demand'. In this case it is up to the application itself to indicate, for each transaction that it executes, if it requires the protection of RETURN TWOSAFE replication. Any transactions where this option is not requested will be replicated using the normal asynchronous mechanism. In this way the application designer can balance the conflicting requirements of performance and data protection.

### Excluding Database Objects from Replication

Normally A/S Pair Replication replicates the entire contents of the database. Sometimes this may not be desirable so an option is provided that allows you to selectively exclude specified tables, cache groups and sequences from replication.

### Data Compression

TimesTen replication provides an option to use software data compression to compress the replicated data stream during transmission over the network. The compression algorithm is a lightweight one (so it does not add much CPU overhead) that typically achieves compression ratios in the order of 2:1. The primary use for this feature is when you are using replication to provide Disaster Recovery between data centers and the network between them is a high latency, limited bandwidth network such as a WAN.

## Parallel Replication – No Commit Order Enforcement Optimization

In addition to data (operation order) dependencies TimesTen parallel replication, by default, also enforces commit order dependencies. This means that regardless of any data dependencies between transactions they will always be committed at the target in the exact same order as they committed at the source. Enforcing commit order dependencies is sometimes necessary in order to ensure correctness but it also limits the performance benefits of parallel replication and makes it highly dependent on the workload mix.

Although enforcing data dependencies between transactions is an absolute must to ensure correctness, for many (indeed most) workloads enforcing strict commit ordering at the target is not necessary (since the operation dependency enforcement is sufficient to ensure correctness). Examples of workloads that typically do not require commit order enforcement are online mobile charging (concurrent phone calls for different subscribers) and session management (each session is independent of all others).

TimesTen provides a configuration option to disable commit order dependency enforcement. When this option is activated most commit ordering enforcement is disabled (some operations, such as DDL, still require commit ordering to be enforced and these are handled automatically). Lab benchmarks and real-world customer experience has shown that this can lead to dramatic throughput improvements for parallel replication. See the section on **A/S Pair Replication Performance** later in this document for some examples.

## Disaster Recovery

TimesTen A/S pair replication supports the deployment of up to 127 read-only subscriber databases. Although the main Active-Standby database pair must be connected via a LAN grade network, the read-only subscribers can be connected to the core A/S pair over higher latency, lower bandwidth connections such as a WAN (of course the network must still have sufficient capacity to sustain the throughput necessary for replication to not lag behind).

A 'remote' read-only subscriber can form the basis for database disaster recovery. If the remote site is equipped as a disaster recovery site (i.e. it has the necessary infrastructure to support full operation as a primary data center) then should some disaster befall the primary site, the read-only subscriber at the DR site can be promoted, by the execution of a few simple commands, to a fully updateable 'active' database. Applications can then start to work against this active database and in the meantime a new standby database can be instantiated locally from the active without disrupting application processing.

From a purely database perspective, the DR copy can be promoted to active within a few seconds and, depending on the database size, full A/S pair HA protection may be available at the DR site within minutes.

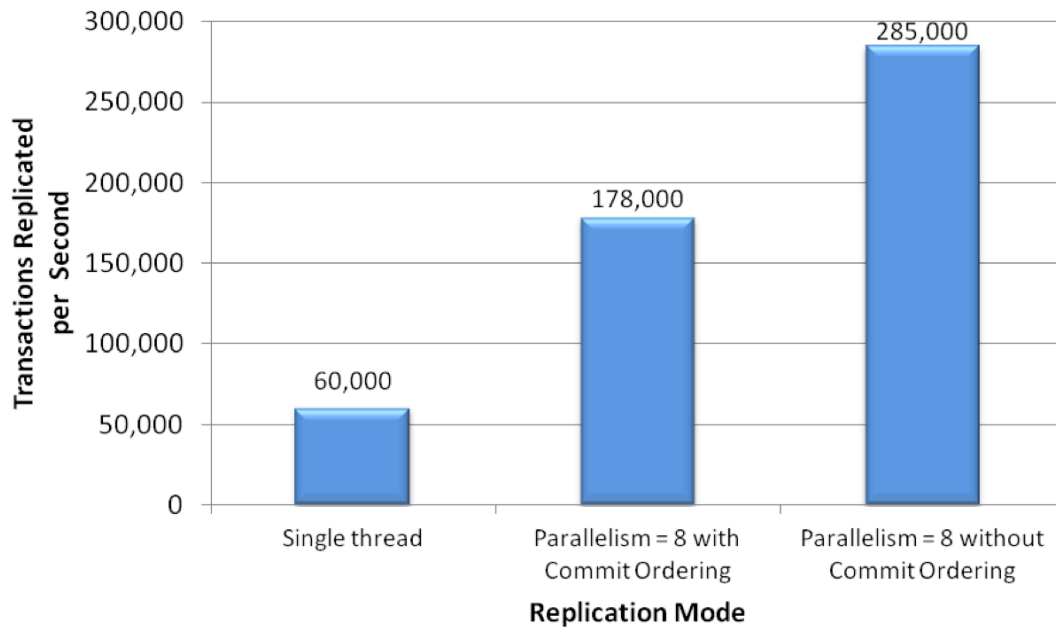
## A/S Pair Replication Performance

TimesTen is all about performance and replication is no exception. In this section you will find performance measurements for various workloads and replication configurations. For full details on the hardware, database parameters, schema and workloads see Appendix A.

The first three benchmarks used the [TptBm](#) benchmark program that is included as part of the [TimesTen Quick Start Guide](#). This is a database intensive benchmark program that can be configured to execute a variety of workloads.

The first example (Figure 5) consists of singleton committed updates (i.e. one single row update operation per transaction) with asynchronous replication.

Here we can see that for this workload, single stream replication can sustain a maximum throughput of around 60,000 transactions per second. Enabling parallel replication (with parallelism = 8) delivers a peak throughput of almost 180,000 TPS (almost 3x that of single stream replication) using the default setting whereby commit ordering is enforced. Enabling the 'no commit ordering' optimization shows even better results with the peak throughput increasing to 285,000 TPS (nearly 5x that of single stream replication).

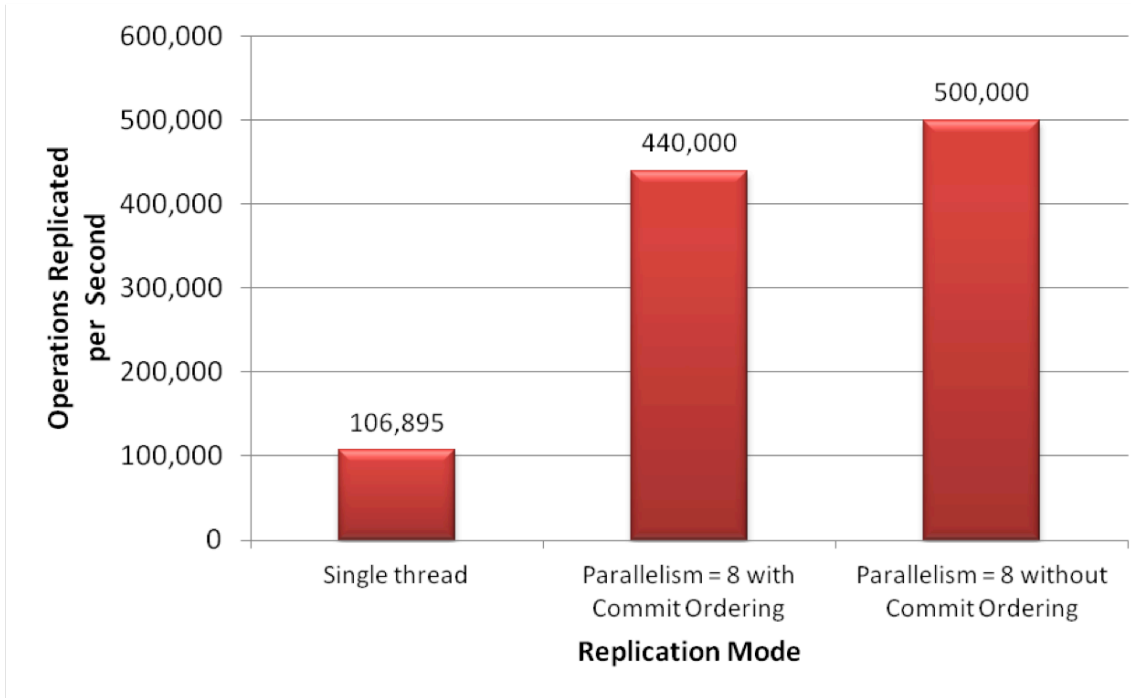


**Figure 5: Asynchronous replication throughput for committed singleton updates**

Of course, most applications do not execute just a single row DML statement per transaction. The next example (Figure 6) shows how the number of operations in a transaction affects replication throughput (note that the scale here is operations/second).

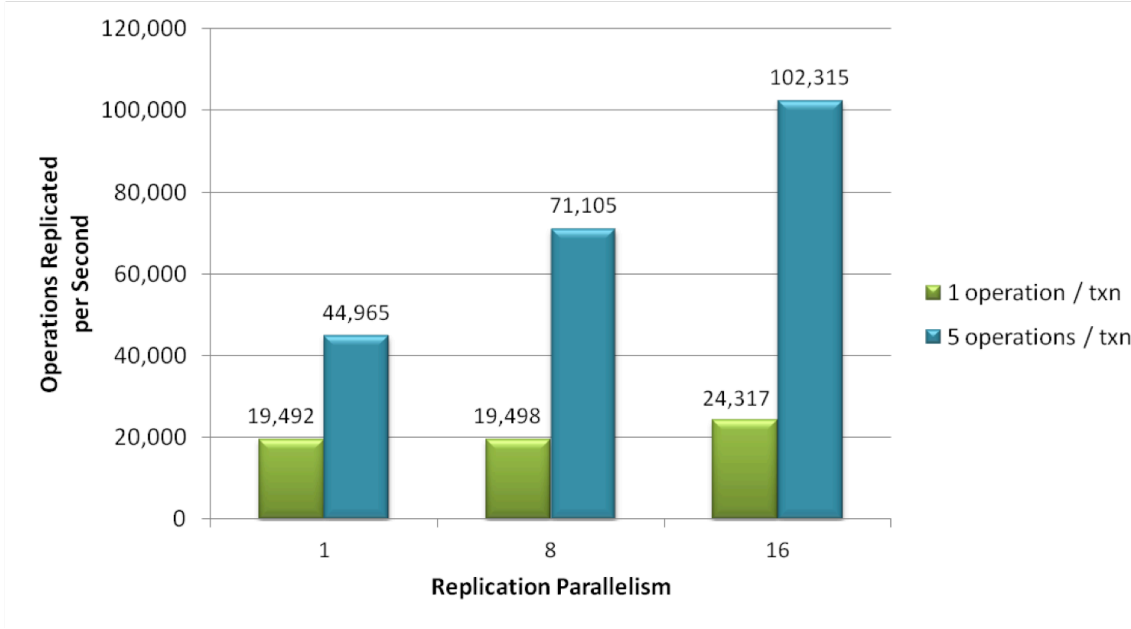
We can see that increasing the number of row updates per transaction from 1 (in the previous example) up to 5 has a major benefit for replication throughput; we can achieve 500,000 replicated operations per second.





**Figure 6: Asynchronous replication throughput with multiple operations per transaction**

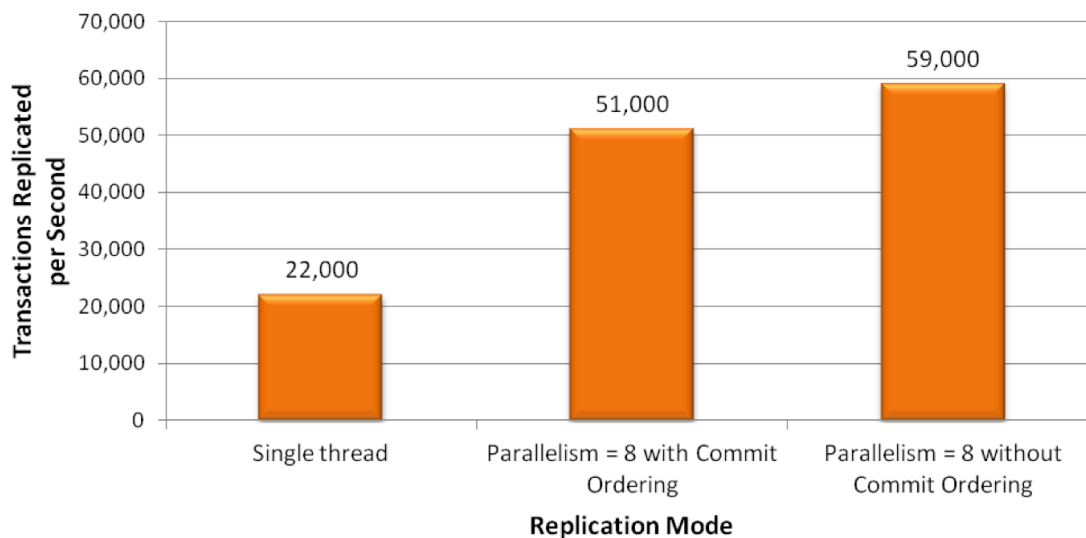
The last TptBm example (Figure 7) is the same as the previous one except that (a) replication is configured for fully synchronous mode (RETURN TWOSAFE) and (b) the no commit ordering optimization is not enabled (it is not allowed with RETURN TWOSAFE mode).



**Figure 7: Synchronous replication throughput with multiple operations per transaction**

Not surprisingly, the throughput of synchronous replication is lower than that of asynchronous replication. The key takeaway here is that transactions that include multiple row changes deliver much higher throughput with synchronous replication, and more scalability as the parallelism increases, compared to singleton transactions. A peak throughput of over 102,000 operations per second for fully synchronous replication is quite impressive.

The final benchmark (Figure 8) uses a different workload; it simulates a session management application. The workload inserts a row in one transaction then deletes a row in another transaction. The table is very 'wide' – it has 31 columns - resulting in a lower overall insert/update/delete transaction rate than was seen for the TptBm update workload.



**Figure 8: Asynchronous replication throughput for session management workload**

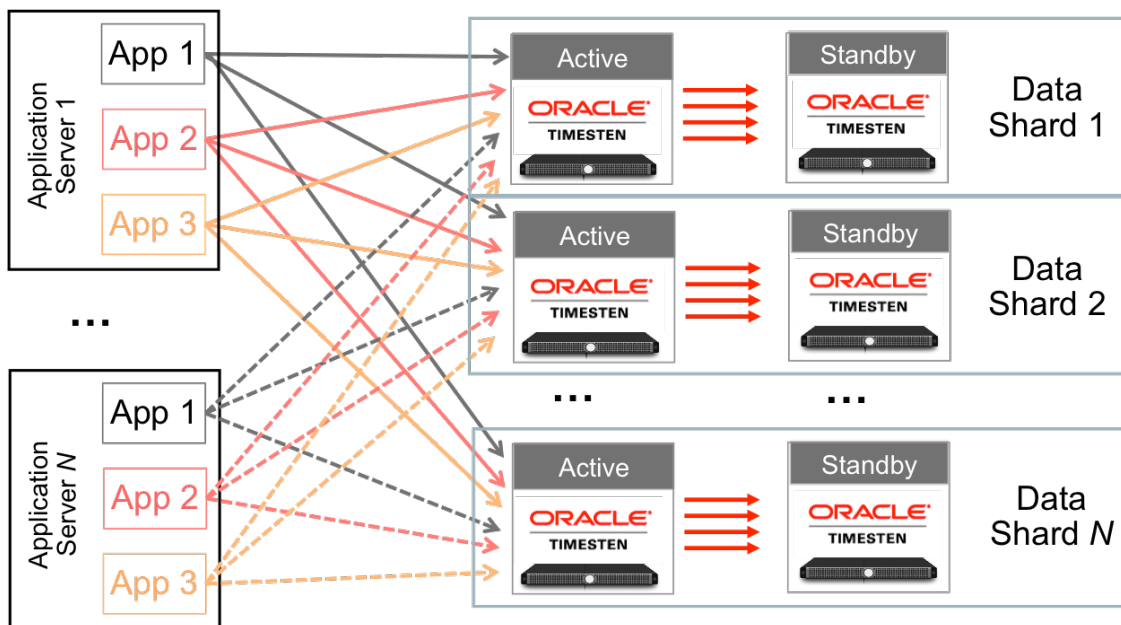
For this workload parallel replication shows almost 3x better throughput compared to single stream replication.

## Example Use Cases

### Location Services for Mobile Networks

This example comes from one of the world's largest providers of mobile telecoms network equipment and associated solutions. This product is their 'Mobile Positioning System'; a centralized database and service that maintains location information (gathered from cell tower triangulation, GPS sensor information etc.) for all the location aware devices connected to the mobile network. The system maintains a central database of up to date location information and provides various APIs and services which can be used by device and network applications to deliver location aware services to the network's users. Examples include location based promotions and discounts, targeted advertising and so on.

The application uses a 'sharding' approach whereby the data is divided into multiple independent shards or partitions each hosted by a TimesTen A/S Pair. This provides flexibility and scalability and allows the system to scale to handle huge volumes of traffic. See Figure 9.



**Figure 9: Architecture of Location Services Use Case**

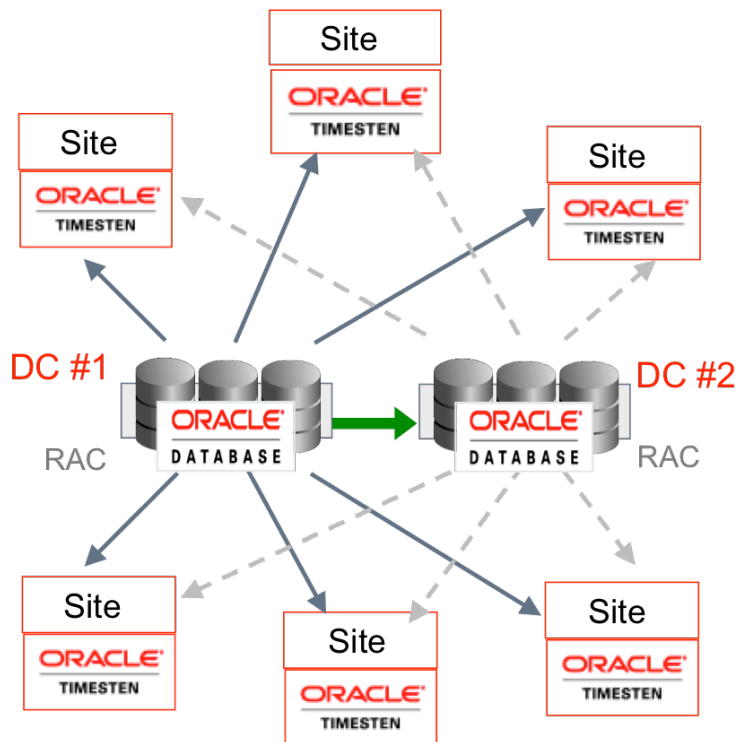
As you might imagine the performance and availability requirements for a system of this nature are challenging. TimesTen was chosen as the database to be used within this system primarily for its performance, high availability and reliability. The latest version of this application to be deployed utilizes TimesTen A/S pair parallel replication with the 'no commit order enforcement' optimization enabled and in real-world traffic simulations is able to sustain continuous operation at over 70,000 business transactions per second, which equates to almost 300,000 database change operations replicated per second. At the same time the responsiveness of TimesTen enables the various APIs and services which run on the platform to deliver low, consistent response times for the various applications which use those services.

#### VOIP Application Server

This example is from a large US residential phone service provider. Their challenge was to build a VOIP application server to support multiple value added applications. This application server would then be deployed in multiple switching centers across the United States. The data needed by the applications hosted in these application servers resides in a centralized Oracle RAC Database located in the Eastern US with a second DR copy, maintained using Oracle Data Guard, located in the Western US.

The VOIP application server deployment locations are remote from the RAC databases and the network connections between the sites introduce significant latency to database access as well as being a potential point of failure. The applications that run in the VOIP application servers require extremely low and consistent data access latency to ensure fast and reliable call setup as this is a major factor in customer perception of service quality. The data required by the application servers is read-only. No updates originate from the application servers – all updates occur at the central Oracle database by way of provisioning systems, customer care agents etc.

The solution adopted by this service provider was to integrate the TimesTen Application-Tier Database Cache into the VOIP application server and use it to cache the critical data from the central Oracle database. See Figure 10 and Figure 11.



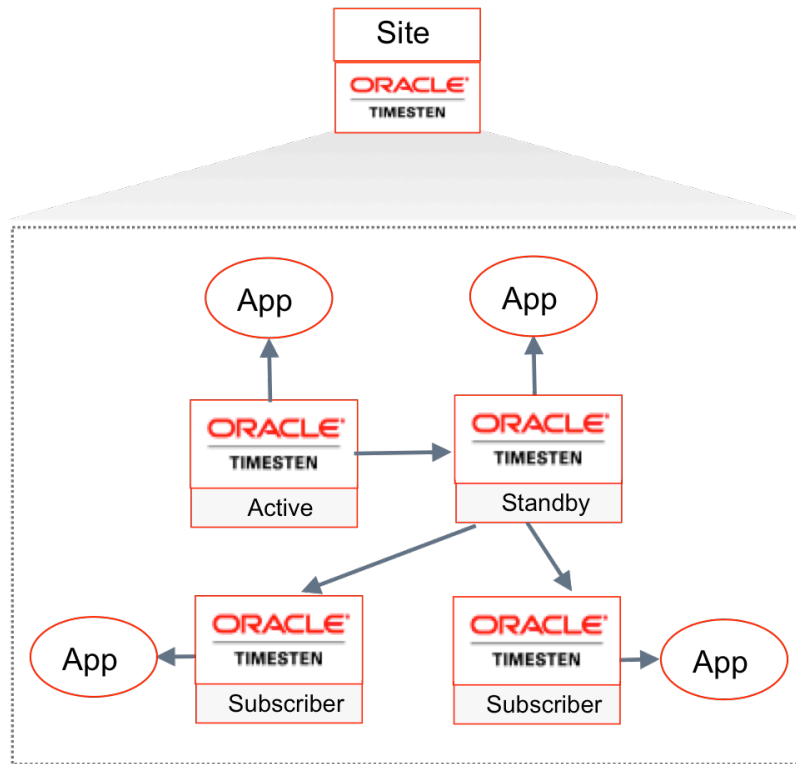
**Figure 10: Overall Architecture of VOIP Application Server Use Case**

In order to:

- » Increase the data query capacity at each switching center in order to support multiple VOIP appservers
- » Provide high availability for each VOIP appserver in the event of a local (cache) database failure
- » Allow each switching center to continue to provide service during scheduled maintenance
- » Provide for autonomous operation of the switching center in the event that the central Oracle database is unavailable (e.g. network connectivity failure)
- » Decrease the cache refresh workload on the central Oracle database

they deployed TimesTen Active/Standby Pair replication with multiple read-only subscribers within each switching center with each TimesTen database copy serving one application server. This architecture provides several key benefits:

- Low, predictable response times for access to the cached data from the VOIP appservers.
- Maximum availability at each site; sites can operate independently for hours or even days in the event that the central RAC database is unavailable.
- The data sub-setting capability of the TimesTen caching technology allowed them to limit the data cached at each site to just that which applies to the region served by the site.
- Central control of what data is cached at each site



**Figure 11: Site Architecture of VOIP Application Server Use Case**

This architecture provides multiple levels of redundancy and very high levels of availability together with the required low latency data access. Measurements show that accessing data in the local TimesTen database has a typical latency of around 2 ms versus 150 ms for data accessed in the central Oracle Database; an improvement of 75x and 50x better than the operational target of 100 ms.



## Conclusion

TimesTen is an In-Memory Relational Database offering very low latency and extremely high throughput for SQL operations while providing standard relational database functionality through standard APIs. It can be deployed as a standalone database or as a high performance relational cache for the Oracle Database.

TimesTen includes comprehensive and sophisticated data replication functionality that enables TimesTen databases to be made highly available without compromising performance.

TimesTen technology enables customers to develop and deploy applications that not only provide real-time responsiveness and extreme throughput but which are also highly available and provide the highest levels of data protection.

## Appendix A – Example Benchmark Details

### Test Systems

- » 2 x Oracle X4-4 Servers.
- » Intel(R) Xeon(R) CPU E7-4890 v2 @ 2.80GHz.
- » 4 sockets, 15 cores/socket, 2 threads / core or 120 virtual CPUs.
- » 512GB RAM
- » 10GbE network.
- » Oracle Linux Server 6.5
- » Transaction logs are on 2 x 1.6GB SSD drives.
- » Database files are on the internal HDD RAID drives.

### Database Parameters (SYS.ODBC.INI settings)


```
DatabaseCharacterSet=UTF8
ConnectionCharacterSet=UTF8
Datastore=/hdd/datastores/Active
LogDir=/ssd/Active
LogFileSize=4096
LogBufMB=4096
PermSize=20000
TempSize=2000
CkptFrequency=0
CkptLogVolume=4096
CkptRate=0
PrivateCommands=1
MemoryLock=4
Preallocate=1
ReplicationApplyOrdering=[0]2]
ReplicationParallelism=X X=[1-32]
LogBufParallelism=X X=[2-64]
```

### TPTBM Table and Workload

100% update workload

```
create table VPN_USERS (
  VPN_ID          TT_INTEGER NOT NULL,
  VPN_NB          TT_INTEGER NOT NULL,
  DIRECTORY_NB    CHAR(10 BYTE) NOT NULL,
  LAST_CALLING_PARTY CHAR(10 BYTE) NOT NULL,
  DESCR           CHAR(100 BYTE) NOT NULL,
  primary key (VPN_ID, VPN_NB))
unique hash on (VPN_ID, VPN_NB) pages = 390546;
```

```
create active standby pair ... [return twosafe];
```



```
update vpn_users set last_calling_party = ? where vpn_id = ? and vpn_nb = ?;
```

### Session Management Table and Workload

```
create table session (sessionId binary(21) not null primary key,  
    i1 integer, i2 integer, i3 integer, i4 integer, i5 integer,  
    b2 binary(16),  
    c1 char(64), c2 char(64),  
    b3 binary(16), b4 binary(1),  
    c4 char (48),  
    b5 binary(16), b6 binary(16), b7 binary(1),  
    c5 char (64), c6 char(64), c7 char (32),  
    b8 binary(16),  
    i6 integer,  
    c8 char(64),  
    c9 char(64), c10 char(64), c11 char(64), c12 char(16),  
    c13 char(16), c14 char(16), c15 char(16), c16 char(64),  
    i7 integer, i8 integer);
```

```
create index ii1 on session(b2);create index ii2 on session (b6);  
create index ii3 on session(c1); create index ii4 on session(c6);  
create index ii5 on session(i5);create index ii6 on session(i6);
```

```
insert into session values (?, ?, ... <31 columns>);  
commit;
```

```
select c1 from session where sessionId = ?;  
delete from session where sessionId = ?;  
commit;
```









**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](http://blogs.oracle.com/oracle)
-  [facebook.com/oracle](http://facebook.com/oracle)
-  [twitter.com/oracle](http://twitter.com/oracle)
-  [oracle.com](http://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

Oracle TimesTen In-Memory Database High Availability  
October 2015  
Author: Chris Jenkins  
Contributing Authors: Susan Cheung, Simon Law, Jenny Bloom, Jason Yang