ORACLE
FUSION MIDDLEWARE

An Oracle White Paper
November 2015

Oracle Tuxedo Advanced Performance Pack

Getting the most out of your Tuxedo applications

ORACLE

Executive Overview

The Tuxedo Advanced Performance Pack offers Tuxedo applications significant improvements in performance and availability as well as simplified and automated operations. Customers using the features provided have reported a 2x to 3x improvement in application performance. For applications that use Oracle Real Application Clusters or RAC, the Tuxedo Advanced Performance Pack provides features similar to the WebLogic Active Gridlink driver. This allows applications to transparently leverage many of the RAC features including scalability, dynamic configuration changes, and planned maintenance, without any code changes.

Audience

This white paper is primarily aimed at enterprise architects, solution architects, and Tuxedo administrators that require highly available and scalable C, C++, COBOL, Java, Python, Ruby, and PHP applications.

Scope

The features described in this white paper are available in the Tuxedo Advanced Performance Pack, which is a licensed option for Oracle Tuxedo.

Features

The Tuxedo Advanced Performance Pack includes features that substantially improve the performance and availability of Tuxedo applications. These enhancements can improve application performance by a factor of 2x to 3x or more. They also reduce administration efforts and improve availability for Tuxedo applications utilizing Oracle Database. The table below categorizes the benefit of each of the features into improving performance, improving available, or improving operational efficiency.

Feature	Improves Performance	Improves Availability	Improves Operational Efficiency
Self-tuning Lock Mechanism	●		●
Shared Memory Interprocess Communication	●		
Tightly Coupled Transaction Spanning Domains	●		
Concurrent Global Transaction Table Lock	●		
Common XID	●		●
Partial One Phase Read-Only Optimization for RAC	●		
Instance Awareness	●	●	
XA Transaction Affinity	●		
Failover/Failback across Database Instances		●	●
Load Balancing across RAC Instance	●		●
Single Group Multiple Branches		●	●
Oracle Database Session Tagging			●

Core Tuxedo Enhancements

These enhancements affect all applications whether they are using Oracle Database or not.

Self-tuning Lock Mechanism

Tuxedo provides a facility where data in shared memory is locked using a combination of a user mode semaphore and a kernel mode semaphore. The user mode semaphore is implemented using hardware specific features such as a test and set instruction. A process wanting the semaphore loops trying to get the lock. The idea is that if the lock is only held for a short period of time, there is less overhead looping than there is yielding to a kernel semaphore that will cause a process context switch. However determining the optimal number of times to loop can be difficult as it depends upon the amount of time the semaphore is held on average by the specific workload provided by the application. This feature makes the number of loops (SPINCOUNT) dynamic and optimized based upon the current load.

With the Advanced Performance Pack enabled, Tuxedo monitors a combination of idle CPU time and the number of times spinning for the user mode lock fails. If the rate of failing to acquire the user mode lock exceeds a threshold, and there is sufficient CPU idle time, Tuxedo will increase the dynamic value of SPINCOUNT each scan unit. On the other hand, if the idle CPU time is below a threshold and the user mode CPU time is greater than 10%, then decrease the SPINCOUNT by 75%. This has the net effect of increasing the SPINCOUNT to decrease the number of times a kernel semaphore and process context switch is required, unless the CPU has become too busy at which point SPINCOUNT will be decreased, reducing the amount of time spent spinning on the user mode semaphore, but potentially increasing the percentage of time a kernel mode semaphore will be required.

Shared Memory Interprocess Communication

Tuxedo normally performs all interprocess communication using System V IPC message queues, or emulated versions of those on platforms such as Windows that don't offer them. While System V IPC message queues provide very good performance, they still require a switch into kernel mode and copying the buffer to/from kernel space. This feature provides an alternative messaging model using a shared memory segment managed by Tuxedo.

When this feature is enabled, Tuxedo buffers allocated using tpalloc() are obtained from a shared memory section created by Tuxedo specifically for interprocess communication. This shared memory section is mapped into the process address space of all Tuxedo servers and native clients. Once the buffer has been populated with the payload associated with a request or reply, it can be used in making a request using tpcall(), tpacall(), tpforward(), and tpsend(), or in a reply using tpreturn(). The message is routed using the normal Tuxedo message routing facility, and once the server or client the message is destined for is determined, it is placed on a message queue maintained in the shared memory section for that process. If the receiver of the message is already processing a message, then nothing more is done as the receiver will check its shared memory message queue before trying its System V IPC queue. However if the receiver is currently blocked waiting on its System V IPC message queue, the sender places a small message on the System V IPC message queue to effectively wake up the receiver. The receiver then checks its shared memory message queue and processes any pending messages. When there are no more pending messages in the

shared memory queue, the receiver performs a normal blocking msgget() call to get the next message, causing the process to block if there is no more work to do.

By using Tuxedo buffers allocated from shared memory, Tuxedo can avoid several buffer copies that might occur. In fact it is possible to perform a request/reply interaction with zero buffer copies. The client does a tpalloc() that gets a buffer from shared memory and calls a service with tpcall() using the TPNOCOPY flag to indicate that the client won't access or modify the buffer before receiving a reply. The service implementation accesses the buffer directly in shared memory without copying, and populating that same buffer with its reply message and performs tpreturn(). The client receives the buffer in shared memory and processes it as it would any other reply. The entire exchange occurs without any copying of the buffer. In some cases this can save as many as 7(?) buffer copies. The performance impact of this can be substantial if the buffer sizes are large. The performance is even better if the servers are generally busy as they can avoid making the blocking msgget() call and stay completely in user mode.

Tightly Coupled Transaction Branches Spanning Domains

The Tuxedo domain gateway allows applications to share services between separate Tuxedo domains. It does this by allowing the domain providing one or more services to export those services, and the domain consuming those services to import them. To the local domain, they appear as local services advertised by the domain gateway process GWTDOMAIN. When a request is made to a remote domain, the local domain gateway accepts the request, relays it over a network connection to the remote domain's gateway, which in turn makes a local request to whatever servers are offering the service.

When the request is part of a distributed transaction, the remote domain gateway normally starts a new subordinate transaction before calling the service, and then takes on the role of coordinator of that remote transaction. This new transaction is only related to the original transaction by the remote domain gateway and as such is loosely coupled with the original transaction. This means that from a resource manager's point of view, they are completely independent transactions with separate global transaction IDs or GTRIDs. As separate transactions, resource managers will not allow them to share resources including locks. This has the potential to cause deadlocks when transactions span domains.

The tightly coupled transaction branch feature allows the remote domain gateway to utilize the same GTRID for the remote transaction. This makes any subsequent branches of the transaction tightly coupled to the original transaction allowing resource managers to share locks and other resources, thus eliminating the possibility of a deadlock occurring. When combined with the partial one phase read-only optimization for RAC, it can allow transactions spanning domains to participate in the one phase commit optimization.

Concurrent Global Transaction Table Lock

Tuxedo manages global transactions by maintaining a table of active global transactions and their participants in the Tuxedo bulletin board, called the Global Transaction Table or GTT. As this table is accessed by multiple concurrent processes it must be protected with a semaphore. In the normal Tuxedo case, the bulletin board lock is used to serialize access to this table. However, under

heavy transaction load, the contention for this lock can become substantial resulting in an artificial performance bottleneck.

The Advanced Performance Pack moves the serialization of access to the GTT from the bulletin board lock to a number of other locks, one for accessing the GTT, and one for each entry in the GTT. This allows a much greater level of concurrency when accessing the GTT and eliminates this bottleneck.

Oracle Database Integration Features

The Tuxedo Advanced Performance Pack includes a number of enhancements related to tighter integration with Oracle Database. Many of these features improve the availability and performance of Tuxedo applications that use Oracle Database.

Instance Awareness

Instance awareness means that Tuxedo Advanced Performance Pack is able to determine which particular Oracle Database instance any given Tuxedo server is using. Whenever a connection is made to an Oracle Database, the connection signature provided with FAN is available so Tuxedo knows which instance each connection is using. This allows Tuxedo to make intelligent decisions about how requests should be routed and how Tuxedo should load balance connections across multiple instances as described in the following sections.

FAN Support

FAN or Fast Application Notification is a facility provided by Oracle Database to allow database clients to know about changes in the state of the database. These notifications let an application respond proactively to events such as a planned outage of a RAC node or an imbalance in database load. Tuxedo Advanced Performance Pack now provides support for FAN notifications and automatically takes the appropriate action on behalf of a Tuxedo application to optimize performance and minimize database interruptions.

Planned Maintenance

The Tuxedo Advanced Performance Pack provides a new system server named TMFAN. This server makes a connection to the Oracle Database ONS service to subscribe to FAN events. Using this information and knowing what Tuxedo servers are connected to which RAC service and instance, Tuxedo Advanced Performance Pack can react to planned maintenance events. When TMFAN receives notification of a service or instance going down for maintenance, Tuxedo moves connections from the instances targeted for maintenance to other services or instances that remain available, before the database administrator shuts down the RAC instances. This is done without any interruption to the application such that it is unaware of the instances being taken out of service.

When a planned operation is initiated, TMFAN receives a planned DOWN event message indicating which database instances or services are being shut down. TMFAN then scans the Tuxedo servers that are connected to Oracle Database and marks those servers connected to the instances or services being shut down to indicate that they need to move their connection to other instances.

Unplanned Outages

An unplanned outage occurs when a database server node fails or becomes unreachable. Normally this would cause database clients to hang when nodes and networks fail, and also cause all subsequent requests from the Tuxedo server that had a connection to that instance to fail. In the case of an instance failure, the TMFAN server receives a DOWN event notification from ONS that the instance has failed, and marks the appropriate Tuxedo servers as needing to reconnect. Using FAN eliminates the hang condition that occurs following hard outages involving nodes and networks.

When using Tuxedo Advanced Performance Pack and RAC with local connections, static state and no XA, Transparent Application Failover (TAF), an Oracle RAC feature, can be enabled. TAF allows connections to transparently fail over to another instance in the case a database request can't be completed due to a failure of the database connection. TAF not only reestablishes the database connection and session, it can also reestablish the state of a failed SELECT statement. If an application sets up session state before it starts database access, the application requires a callback to be registered in order to allow the application to re-establish the session state when a session fails over.

For XA servers not using TAF, Tuxedo will notice the session failure at the next XA call to the database, typically xa_start() or xa_end(), at which point Tuxedo will call xa_close() followed by xa_open() before dispatching the next request, to re-establish the database connection. This will cause the Tuxedo server to switch its database connection to one of the remaining instances for the database service in the RAC cluster. Non-XA servers without TAF enabled will have to handle the connection failure and reestablish the database connection in their own code.

Tuxedo XA servers will reconnect to another instance before their next request is dispatched, although the service currently executing may receive a failure. For non-XA servers that are adapted to use TAF, TAF will reconnect to another instance and can attempt to replay the SELECT statements that would have otherwise returned a failure to the application.

RAC Load Balancing

FAN notifies the Tuxedo TMFAN server of load balancing advisories or RLB events. These advisories tell Tuxedo how much load each RAC instance should receive. If the change in advised load exceeds the threshold specified in the TMFAN command line switches, then Tuxedo will update the load information in the Tuxedo bulletin board and attempt to balance the number of connections for that service according to the advisory. Tuxedo uses the load information in the bulletin board as part of its request routing algorithm. When selecting a specific Tuxedo server to route a request to, Tuxedo will see if the potential servers are connected to the same database service, and if so route the request to the instance that has the lower database load.

End-to-End Application Tracing

Oracle Database provides a feature to assist in tracing application activity associated with database calls. A database client can tag a session with the name of the client application, the name of a module within the application, the name of an action being performed by the application, and user information. This information is carried to the database server and shown in database reports and logs, helping diagnose potential problem applications. With the Tuxedo Advanced Performance Pack, Tuxedo can automatically tag Oracle Database requests with the name of the Tuxedo server as

the module name, the name of the Tuxedo service as the action name, the name of the Tuxedo application client name as the client identifier, and finally the Tuxedo user as the client info.

When sessions have been tagged in the way described above, DBAs can readily see what applications, Tuxedo servers and services, and users are either having difficulties or perhaps causing performance problems. DBAs can examine performance and usage graphs based upon these tags to help prevent or troubleshoot performance problems, or poor user experience.

RAC Related Transaction Enhancements

These enhancements improve the performance, availability, and operation efficiency of Tuxedo applications that use Oracle Real Application Clusters and XA distributed transactions. RAC combined with Tuxedo and Tuxedo Advanced Performance Pack provide some of the best performance and availability by an application server on the market.

Common XID

In XA transactions the transaction manager creates a transaction branch identifier called the XID for each branch of the transaction. This identifier is made up of three components. The first is the format ID which identifies the type of transaction manager managing the transaction. The second part is the global transaction identifier or GTRID which uniquely identifies the global transaction. The third component is the branch qualifier that identifies the specific branch of the global transaction. Each global transaction within a Tuxedo domain shares a common format identifier and a common GTRID. Without the Tuxedo Advanced Performance Pack, each Tuxedo server group participating in an XA transaction has its own transaction branch, meaning a unique branch qualifier. If a transaction spans two or more separate Tuxedo server groups, two or more branches would be involved in the transaction, even if the servers in those groups were connected to the same resource manager. Once more than a single branch is involved in the transaction, Tuxedo needs to perform a two phase commit and write to the transaction log.

The common XID feature of the Tuxedo Advanced Performance Pack tries to minimize the number of transaction branches to reduce the size of the commit tree, ideally reducing the number of branches to one, thus allowing a one phase commit. To do this the Tuxedo Advanced Performance Pack tracks which servers have connections to an Oracle Database. As well because of instance awareness (see below) Tuxedo Advanced Performance Pack knows which RAC instance any given server is connected to. When routing a request to a server, Tuxedo Advanced Performance Pack uses the instance awareness to know if the instance the candidate server is connected to is already enrolled in the transaction. If it is and its group is coordinating the transaction, Tuxedo uses the same XID that was used to initially enroll the instance in the transaction. This occurs even if the server is another Tuxedo server group, meaning that even though there is a new group participating in the transaction, a new transaction branch isn't necessarily added to the transaction.

Partial One Phase Read-Only Optimization for RAC

One of the performance features of Oracle RAC for XA transactions is that if distributed transaction branches span multiple RAC instances, RAC will respond with XA_RDONLY to the

xa_prepare() calls on all branches but the last branch prepared. The partial readonly one phase commit optimization in Tuxedo Advanced Performance Pack leverages this RAC feature by preparing all but one database branch before trying to prepare the last branch. If all branches of the transaction were against the same RAC database, these previous xa_prepare() calls would all receive XA_RDONLY, and Tuxedo would then skip the prepare call on the last branch and simply call xa_commit(TMONEPHASE), performing a one phase commit on the last branch. While this skips the prepare call on that last branch, it more importantly skips the transaction log write as the transaction is being completed with a one phase commit. Skipping the transaction log write can have a significant performance benefit to applications that make heavy use of XA.

XA Transaction Affinity

Because Tuxedo Advanced Performance Pack tracks which Tuxedo servers are connected to which RAC instance, Tuxedo can use this information as part of its request routing algorithm. Without the Tuxedo Advanced Performance Pack, Tuxedo only considered client/server affinity, server load, and data dependent routing in making the selection for which Tuxedo server should handle a specific request. Because splitting XA transactions across multiple RAC instances incurs a significant overhead in RAC to coordinate the branches of the transaction, Tuxedo now uses instance awareness to track which RAC instances are participating in an XA transaction. This allows Tuxedo to try and route service requests to minimize the number of instances that are enlisted in the transaction.

Tuxedo attempts to route a request to the least loaded server that has a connection to a RAC instance that is already participating in the transaction. In an ideal situation, this results in all database interactions in a distributed transaction using the same RAC instance. If no other resource managers are involved in the transaction, Tuxedo can perform a one phase commit eliminating the need to perform the prepare phase of the two phase commit protocol and eliminating the need for a transaction log write. Even if Tuxedo can't perform the one phase commit operation, the performance of the RAC database is improved by minimizing the cross instance communication that would otherwise need to occur.

In some situations a Tuxedo service may be offered by multiple Tuxedo servers and those servers may be connected to difference database services. Even if Tuxedo can't select a server connected to a database instance already enlisted in a transaction, Tuxedo will attempt to select a Tuxedo server connected to the same database service.

Single Group Multiple Branches

The servers in a Tuxedo server group are normally expected to be connected to a single resource manager. Although in some respects Oracle RAC is a single resource manager in that it represents a single database, the instances in some regards act more like separate resource managers in that a single transaction branch isn't allowed to span multiple RAC instances. Normally Tuxedo associates a single transaction branch with a server group. However this can cause problems if the database service the group is associated with can span multiple instances as a single transaction branch could potentially be used across multiple instances, which RAC doesn't allow. This means that the database services used by Tuxedo server groups need to be single instance or singleton services to prevent a transaction branch from being split across instances.

With the Tuxedo Advanced Performance Pack, Tuxedo servers in a server group can now be associated with services that span multiple instances as the pack provides the ability for Tuxedo server groups to use multiple branches. This feature, called Single Group Multiple Branches, relies on instance awareness so Tuxedo knows whether it can use the existing branch associated with the group, or whether a new branch is needed because the server handling the request is attached to a different instance. The result is that Tuxedo can now leverage the benefits of database services that are offered on multiple instances to improve performance and availability.

Summary

The enhancements provided by the Tuxedo Advanced Performance Pack provide significant performance, availability, and operational benefits to Tuxedo applications without requiring any changes to Tuxedo application code. The RAC related features are on par with those provided Oracle WebLogic Server Active GridLink, yet for non-Java applications in an environment where database connection pooling isn't used or required. By allowing the dynamic reconfiguration of RAC databases and support for Application Continuity, Tuxedo application can now provide continuous uptime without any loss of service due to normal database maintenance procedures, and minimal loss of requests only in certain failure scenarios. Performance improvements as reported by some customers suggest that application performance may be increased by a factor of 2 to 3x by these enhancements. Tuxedo combined with the Tuxedo Advanced Performance Pack provides the best possible application performance and availability for all of your Tuxedo applications.

White Paper Oracle Tuxedo Advanced Performance Pack
November 2015
Author: Todd Little

Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

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

Hardware and Software, Engineered to Work Together