ORACLE
FUSION MIDDLEWARE

An Oracle White Paper
May 2014

Mission Critical Applications with Oracle Engineered Systems and Tuxedo

ORACLE

Disclaimer

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

Table Of Figures

Introduction

Oracle engineered systems have introduced a new way of creating hardware and software platforms to support enterprise applications. These systems have the hardware and software designed to work together to provide extreme performance, simplified management, and an integrated flexible platform for both physical and virtual environments. Whereas in the past, each layer of an application platform was designed and built independently, Oracle engineered systems are being designed from the ground up to optimize enterprise application performance. Oracle Exalogic, Oracle SuperCluster, and Oracle Exadata are three of these systems and they include the following features:

InfiniBand Networking

InfiniBand has long been associated with high performance computing. It provides a fully interconnected mesh network that offloads much of the network processing to highly intelligent Host Channel Adapters or HCAs. An InfiniBand HCA takes care of much of the networking protocol stack by providing guaranteed sequential delivery of messages even in the presence of failures. This allows the operating system and application to skip the TCP/IP layers as the hardware provides for the capabilities those layers normally provide. Using the Socket Direct Protocol, applications can bypass the kernel TCP/IP layers thereby reducing CPU consumption and minimizing latency.

One of the main features that InfiniBand HCAs provide is the ability to directly access the memory of a remote machine. This capability called Remote Direct Memory Access or RDMA allows an application on one machine to directly read from or write to the memory on another machine. All of this can be done in user mode without having to involve the kernel. This can reduce network latency to less than 1 microsecond, and dramatically reduce host CPU consumption as the transfers occur without having to switch to kernel mode.

Integrated Storage

Each Oracle Exalogic and SuperCluster system includes an integrated highly available storage appliance. These storage appliances are built using the Oracle ZFS file system and hardware optimized to deliver enterprise storage. These NAS systems provide built-in transactional file system updates, snap shot capabilities, and improved data integrity features to ensure that stored data is always accessible and without errors. Each appliance has dual heads to ensure availability of the file systems and are connected to the other hosts in the system via InfiniBand. This ensures that there are two paths to any storage as the InfiniBand controllers provide redundant network paths, the drives are dual ported between the two heads, and the two heads monitor each other allowing one to take over from the other in the case of a head failure.

Oracle Exadata systems come with integrated Oracle Exadata Storage Servers, or storage cells, that provide high performance access to storage for the database servers. The software in these servers perform operations such as Smart Scan, Hybrid Columnar Compression and other advanced storage capabilities to offload these operations from the database servers and minimize the amount of data that has to be transferred between the storage servers and the storage cell. The cells are connected via InfiniBand and data is automatically mirrored across multiple cells.

Large Physical Memory

Oracle Exalogic and SuperCluster systems are configured with large amounts of physical memory. Depending upon the generation and type of the system, this ranges from 96GB to 1TB of DRAM in each compute node. This large amount of memory can be utilized to provide large buffer caches, support a larger number of virtual machines, or leveraged to provide extremely high performance inter-process messaging.

Tuxedo Enhancements for Exalogic and SuperCluster Systems

As of the Tuxedo 11.1.1.3.0 release and later, Tuxedo has built in features to maximize performance of Tuxedo applications on Exalogic and SuperCluster systems hardware. These features are part of the Exalogic Elastic Cloud Software enhancements made to Tuxedo to optimize Tuxedo application performance on engineered systems. This includes using InfiniBand RDMA to deliver requests or replies to services in a clustered environment, using the large amount of physical memory to provide shared memory messaging, and optimizing the performance of locking mechanisms with the higher core counts associated with engineered systems. These features are described in more detail below.

BRIDGE Bypass

Tuxedo natively supports clustered environments. A Tuxedo cluster consists of two or more machines that work together to provide a highly scalable and available platform for running native language applications. As more resources are needed, additional machines can be added to the cluster to increase the hardware resources available to the Tuxedo application. Machines within the cluster communicate with each other via TCP/IP using a system server called the BRIDGE. The BRIDGE relays messages from one machine to another machine in the cluster.

When operating in a clustered environment, Tuxedo is managed as a single entity and all services available on any machine in the cluster can be utilized by any client or other server running on any machine. Thus if the DEPOSIT service is advertised on machine A, that service is visible and accessible to all clients and servers in the cluster. The Tuxedo request routing mechanism can then load balance requests across the servers offering a service regardless of the machine on which the servers are running.

Availability is improved in a clustered environment because servers can be run on multiple machines and as long as at least one copy of the server is running on a machine, the services it offers will be available to all machines. For example, in a cluster with three machines named A, B, and C, the server offering the DEPOSIT service is running on machines A and B, but not on C. If machine A is down because of a failure or for maintenance, the DEPOSIT service will still be available to all clients and servers on machines B and C. When machine A is brought back up and the server offering the DEPOSIT service has booted on machine A, Tuxedo will again start routing requests to that server. This allows 24x365 operation without any interruption in service. Adding additional machines will further increase the availability of the application.

In non-engineered systems, the BRIDGE server is used to relay requests and reply messages to and from servers running on another machine in the cluster. The BRIDGE effectively acts as a proxy server to allow processes on different machines to transparently communicate with each other. The normal flow for local requests is:

1. The request message is placed on the servers System V IPC queue

2. The server takes the message from its System V IPC queue, processes the request, and creates a response message
3. The response message from the server is then placed on the clients reply queue

For requests to servers located on another machine, the flow is as follows:

1. The request message is placed on the System V IPC queue for the BRIDGE
2. The BRIDGE then sends the request message via TCP/IP to the BRIDGE on the remote machine
3. The BRIDGE on the remote machine takes the request message from the network and places the System V IPC queue of the server on the remote machine
4. The remote server takes the message from its System V IPC queue, processes the request, creates a reply message, and places the reply message on System V IPC queue of the remote BRIDGE
5. The remote BRIDGE then sends the response message via TCP/IP to the BRIDGE on the local machine
6. The BRIDGE on the local machine takes the message from the network and places the response message on the local clients reply queue

This diagram shows the message flow on non-engineered systems:

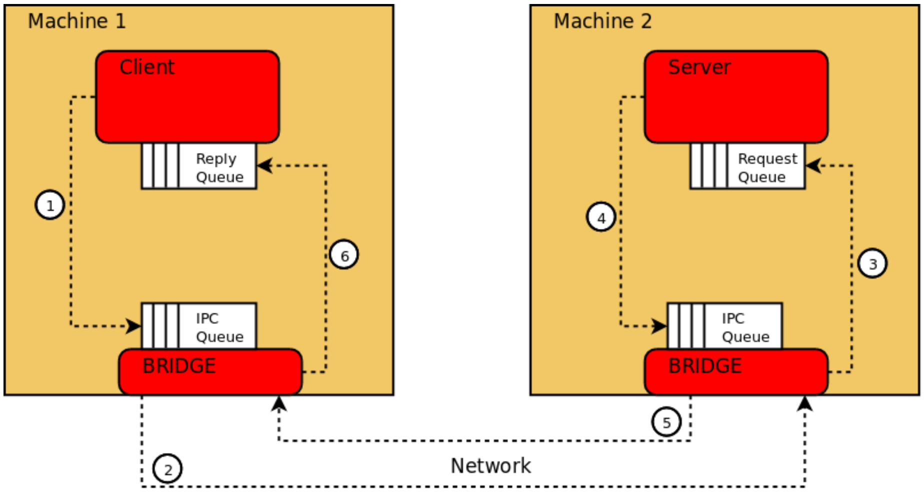


Figure 1: BRIDGE Communication

Clearly executing requests locally provides better performance as the whole process can be done in less than 35 microseconds. When executing requests on a remote machine, the process can take more than 1 millisecond. For fine grained services this might be fairly high, but for normal coarse grained services the cost of making a remote request is offset by the additional scalability and availability clustering offers.

On engineered systems the path is much more direct. By utilizing the RDMA capabilities of InfiniBand, Tuxedo can minimize the impact of using a service on another machine. Tuxedo 11.1.1.3 introduced the ability for clients to directly access remote services. Leveraging InfiniBand, Tuxedo is able to allow processes accessing remote services to directly communicate with those processes, bypassing the BRIDGE process. Thus on an engineered system, Tuxedo uses RDMA to place requests and responses directly on the associated queue. This makes the path for accessing remote services look like:

1. The request message is sent to the remote servers queue using InfiniBand RDMA
2. The remote server takes the message from its queue, processes the request, and creates a reply message which is then sent directly via RDMA to the clients reply queue

This diagram illustrates the message path on engineered systems:

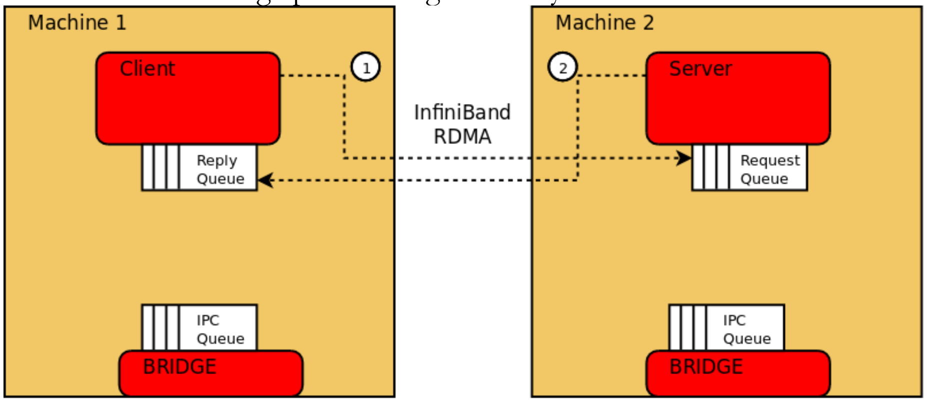
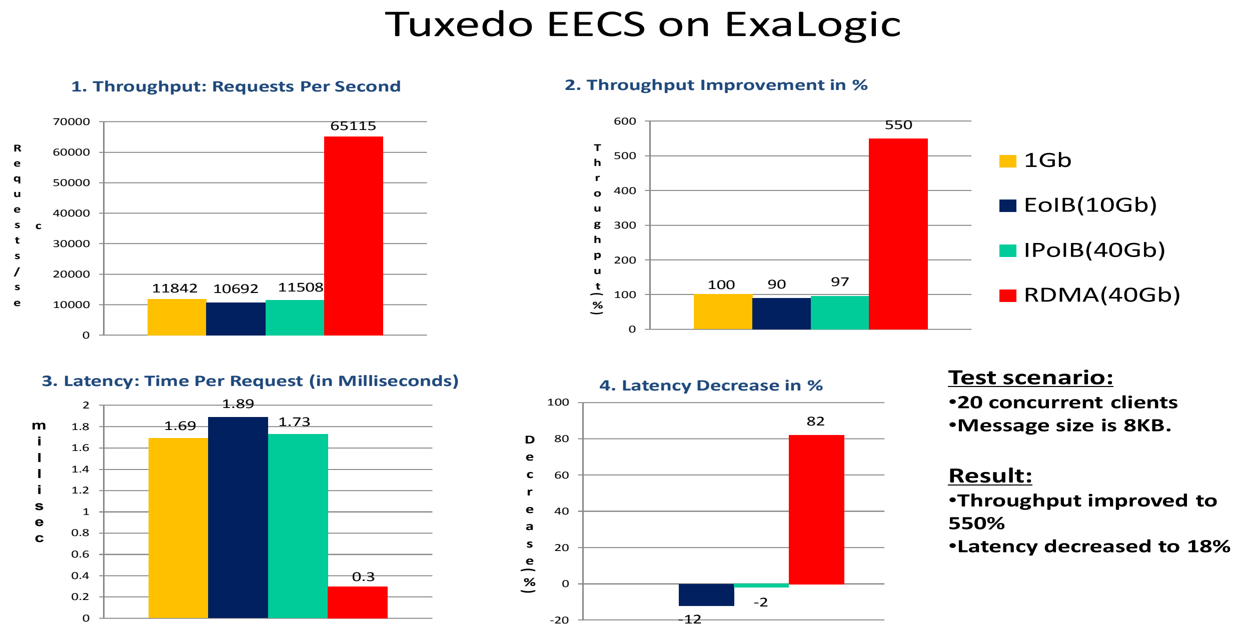


Figure 2: BRIDGE Bypass on engineered systems

This path looks much like the path taken by requests made to local servers and performs much better than the path taken through the BRIDGE. The following charts show the amount of messaging performance improvement that can be obtained by leveraging the RDMA capabilities of InfiniBand on engineered systems.



Test scenario:

- •20 concurrent clients
- •Message size is 8KB.

Result:

- •Throughput improved to 550%
- •Latency decreased to 18%

Figure 3: BRIDGE Bypass Performance

As the charts show, throughput can increase up to 550% and latency decrease by as much as 82%. For lightweight services this can have a dramatic effect on performance and significantly improve Tuxedo's ability to load balance requests across a cluster of machines as the cost to invoke a remote service decreases by so much.

Domain Gateway Bypass

Large Tuxedo applications are often made up of multiple Tuxedo domains. These domains operate largely independently, although often sharing services. They may be configured based upon application boundaries, administrative boundaries, or geographical boundaries. A standard feature of Tuxedo is the domain gateway. It allows Tuxedo domains to share services with other Tuxedo domains. On non-engineered systems, the domain gateway operates much like the BRIDGE in that it acts as a proxy for services in another domain. The major difference is that the BRIDGE is also used to convey machine and cluster state information to the other nodes in a cluster, whereas essentially no state information is shared by the domain gateways.

For domains running within InfiniBand connected engineered systems, the Tuxedo 12.1.3 release offers the option to utilize RDMA for communication between clients and servers across domain boundaries, bypassing the domain gateways to improve performance. On non-engineered systems, a request to a service in a remote domain is first given to the local domain gateway. The local domain gateway then sends the message via TCP/IP to the remote domain gateway, which then in turn gives the message to a server in the remote domain that offers the service. The reply to the service takes the same route in reverse. Although domain gateways can handle thousands of requests per second, the additional steps and buffer copies limit the throughput over what can be achieved. By bypassing the domain gateways, the performance of domain to domain communication within engineered systems can be greatly improved.

Like the BRIDGE bypass feature, communications between clients and servers in remote domains is coordinated and set up by the domain gateways. They exchange the necessary RDMA keys to allow a client to directly send messages to the remote server, bypassing the domain gateway once the communication end points have been set up.

Enhanced Support for RAC

Tuxedo on engineered systems offers a number of enhancements to support Oracle Database Real Application Clusters or RAC. These enhancements improve the performance and availability of Tuxedo applications that utilize RAC.

Oracle Database FAN Support

Oracle RAC provides a facility called FAN or Fast Application Notification. On engineered systems, Tuxedo makes use of this facility to track the topology of a RAC database to know what database services are available on each instance and the status of each instance. A Tuxedo system server registers with RAC to receive FAN notifications that tell Tuxedo when instances are down or being brought down, what instances are up, and how much load each instance should handle. Tuxedo uses this information to spread the database connections for a database service to all available instances in proportion to the Realtime Load Balancing or RLB events that FAN generates.

When an instance is being brought down, Tuxedo will automatically redirect connections to that instance to another instance that provides the database service. This ensures that RAC instances can be brought down without an interruption to the Tuxedo applications using that instance. When a new instance is brought up, Tuxedo will use the information FAN has provided to migrate the connection of some servers to the new instance based upon the RLB information it has received. Together this allows the seamless support for instances being brought up and down.

Transaction Affinity

Tuxedo now tracks the Oracle database instances that are participating in a global transaction. This information is carried along with other transaction context when transactions span domains. Tuxedo uses this information to provide transaction affinity, meaning that Tuxedo attempts to route requests to Tuxedo servers that already have a connection to the instance or instances that are already part of the transaction. This helps reduce the number of participants in a global transaction speeding up the commit processing. As well it improves database performance as the requests end up on instances where the transaction and its associated locks and data are already in memory.

Tightly Coupled Transactions

On Oracle engineered systems, Tuxedo will attempt to preserve the global transaction identifier or GTRID for transactions that span multiple domains. Previously a subordinate or interposed transaction was created on the remote domain with a new GTRID. This meant that if both domains interacted with a common resource manager, the resource manager would see the two transactions as completely independent. By sharing the same GTRID, the transactions in each domain will be seen as part of a single global distributed transaction.

Common XID

Going beyond tightly coupled transactions, on engineered systems, Tuxedo will try to not only use a common GTRID, but if possible use a common branch qualifier. This works across multiple groups within a domain whereas on non-engineered systems each group participating in the transaction would have used a unique branch qualifier. With this feature all groups in the transaction connecting to the same Oracle database instance will use the same branch qualifier. Likewise if a transaction spans domains, not only will the same GTRID be used, but if the request can be routed to an instance already participating in the transaction, Tuxedo will use the same branch qualifier as was previously used.

One Phase Commit

Because of the transaction affinity and common XID features, in many scenarios where the only resource manager actually participating in the transaction is Oracle database, a one phase commit can be performed, even though the transaction spans multiple groups or domains. Using a one phase commit eliminates the need for a transaction log write and dramatically improves the performance of applications using distributed transactions.

Shared Memory Message Queues

Another feature on engineered systems is the ability to utilize shared memory to exchange request and response messages. Using a shared memory segment, request messages are allocated from a buffer pool in shared memory and used to pass the message to the server offering the requested service, and to pass responses back to the client. Without any code changes, this minimizes the number of buffer copies that are required to pass messages between clients and servers. With minimal code changes, a client can indicate that once it makes a request, it will no longer use the request buffer, and Tuxedo can pass the message to the server with zero buffer copies. A servers response message is always sent this way, i.e., with zero copies. This typically bypasses the normal System V IPC queue mechanism used by Tuxedo and offers much greater performance, although under light load a System V IPC message is used to wake the receiving process and notify it that it has a message in shared memory to process. This diagram shows the normal System V IPC based messaging:

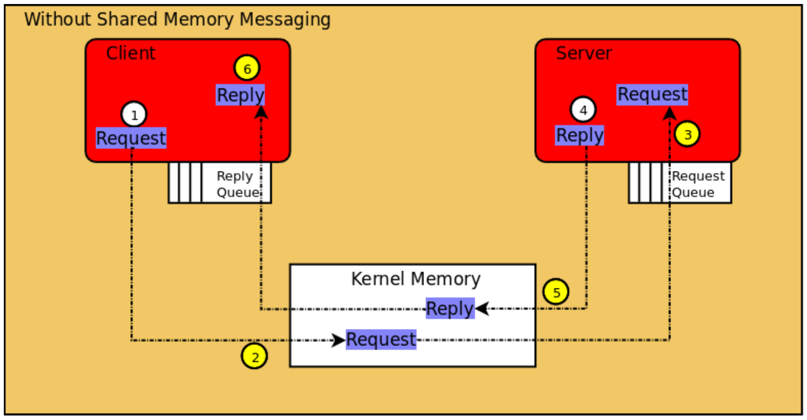


Figure 4: IPC Message Flow

The steps shown in yellow indicate where a message needs to be copied in memory. For small messages this overhead may not be significant, but for larger messages copying the request and reply messages each twice can incur substantial overhead. This diagram shows the messaging path using shared memory:

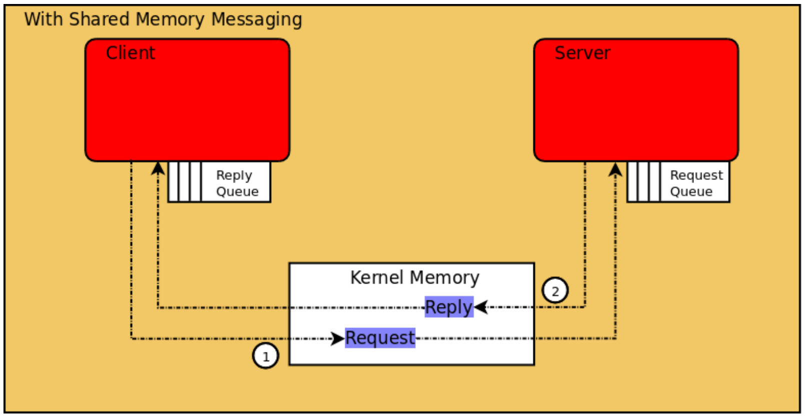
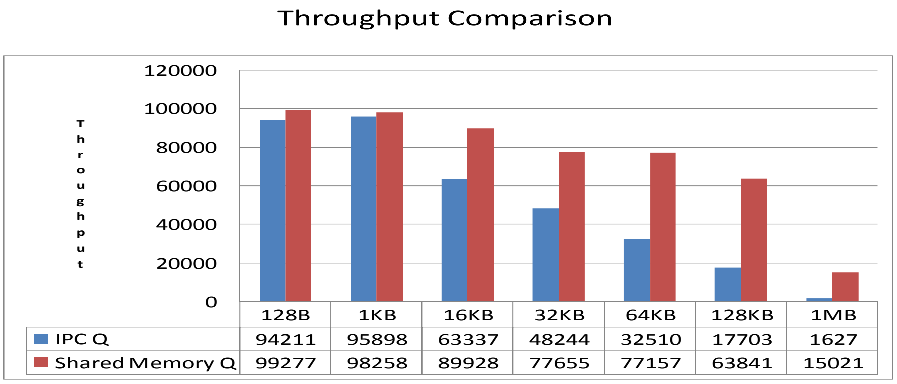


Figure 5: Shared Memory Messaging Flow

Note there are no buffer copies made in this messaging path. For larger messages, the improvement in messaging performance can exceed 800% as the following chart shows:



	128B	1KB	16KB	32KB	64KB	128KB	1MB
🟦 IPC Q	94211	95898	63337	48244	32510	17703	1627
🟥 Shared Memory Q	99277	98258	89928	77655	77157	63841	15021

Figure 6: Shared Memory Messaging Throughput

Self-tuning Locking Mechanism

Tuxedo makes heavy use of System V shared memory segments. This includes the bulletin board that holds the Tuxedo configuration and state of the application. In order to coordinate the access to shared memory segments; Tuxedo uses a combination of a user mode semaphores and kernel mode System V semaphores. The user mode semaphore is implemented using a test and set instruction which has extremely low overhead. If the semaphore can't be obtained immediately, Tuxedo loops trying to obtain the semaphore for a period of time referred to as SPINCOUNT. On non-engineered systems, this is a value that must be set by the administrator of the Tuxedo application. If the process trying to obtain the user mode semaphore is unable to obtain the semaphore in the number of loops specified by SPINCOUNT, Tuxedo reverts to using a kernel mode semaphore to block the process until the semaphore can be obtained. The optimum value of SPINCOUNT is normally determined by trial and error based upon the application's access to shared memory, the amount of contention for a semaphore the application generates, and the number of cores or processors on a machine.

On engineered systems, Tuxedo varies the amount of time spent looping trying to obtain the user mode semaphore based upon the current CPU utilization and the number of times Tuxedo has had to resort to a kernel mode semaphore because the number of loops was exceeded. This is controlled by two parameters. The first is SPINTUNING_FACTOR, this sets the target for how often Tuxedo resorts to using a kernel mode semaphore to coordinate access to shared memory. So setting a value of 100 indicates that Tuxedo should loop enough times such that only 1 in 100 times does it need to resort to a kernel mode semaphore. The other parameter is SPINTUNING_MINIDLECPU which sets the goal for the minimum amount of idle CPU time. If the idle CPU time falls below SPINTUNING_MINIDLECPU, then Tuxedo will decrease the value of SPINCOUNT to try and get back to SPINTUNING_MINIDLECPU idle time. Thus if SPINTUNING_MINIDLECPU is set to 10, Tuxedo will continue to increase the value of SPINCOUNT until either the SPINTUNING_FACTOR is reached or the idle CPU drops below 10 percent.

SDP Support

One of the benefits of using InfiniBand based network hardware is the ability to utilize the socket direct protocol, or SDP. This protocol allows applications to communicate with each other via the normal socket interface but bypass the network processing associated with TCP/IP which includes things like ordering, fragmentation, timeouts, retries, and the like because the InfiniBand hardware takes care of those concerns. As well SDP can support zero copy transfers as the InfiniBand hardware is capable of directly transferring buffers from the caller's address space. By utilizing SDP, Tuxedo applications can reduce the amount of CPU consumed for networking operations as well as increase the overall throughput of network operations. SDP can be used on all Tuxedo network connections including BRIDGE to BRIDGE communication, the domain gateway GWTDOMAIN for communication with other Tuxedo domains, for workstation and Jolt clients, and as well for communication with WebLogic Server via the WebLogic Tuxedo Connector.

Deployment Topologies

Cluster or Multiple Machine Domains

The most common deployment topology for Tuxedo on engineeered systems is to utilize a multiple machine domain, also called a cluster. In this model an application is deployed across multiple host nodes and managed as a single domain. The obvious advantages include high availability and simplified management. Application servers run on each node in the cluster and provide services to all clients or servers in the cluster transparently. The load within the cluster is distributed in such a way to optimize response time for service requests. Thus if the servers on one machine are responding slowly while server on another machine are responding more quickly, Tuxedo will shift load to the machines that are responding more quickly. When combined with TSAM Plus, this is done in essentially real time with all aspects of response time being taken into consideration, including network time to route the request to a remote machine, current work queued to the servers, and the network time to return the response to the requester.

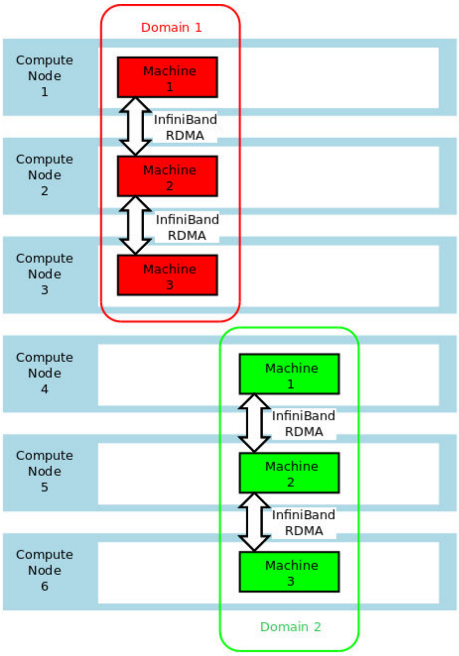


Figure 7: Clustered Deployment

Multiple Single Machine Domains

Another popular deployment strategy for Tuxedo is to use multiple single machine domains. In this scenario each domain resides on a single machine and is managed independently of the other domains. For applications that require more resources than a single machine can provide, the Tuxedo domain gateway can be used to connect the separate domains. While this is similar to a cluster deployment, it differs in several significant ways. First each domain is separate from every other domain, and only services that are exported from the domain are available to be used by other domains. Second the availability of services is only known to the local domain, so if a domain is using the services of another domain and those services for some reason or another are not currently available, requests to those services will fail. The major advantage of this deployment model is that it is possible to control what machines or domains have access to the services offered by a domain, and the domains operate in essentially a shared nothing architecture, including but not limited to configuration.

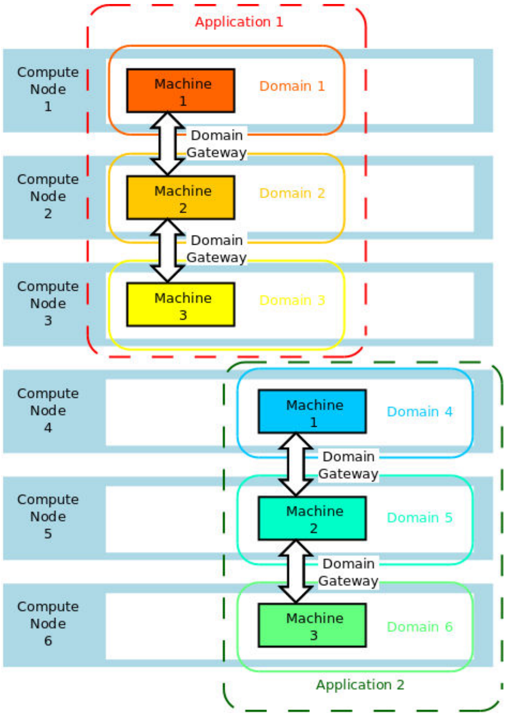


Figure 8: Multiple Single Machine Domains

Multiple Clusters Sharing Machines

Often a Tuxedo application won't require all of the resources available in a clustered configuration, but still desire the high availability of a cluster. In this model multiple clusters are used and share a common set of machines. Tuxedo allows multiple domains to be deployed on a single machine, thus several independent applications can share the resources of multiple machines, each benefitting from the high availability achieve by running on multiple machines.

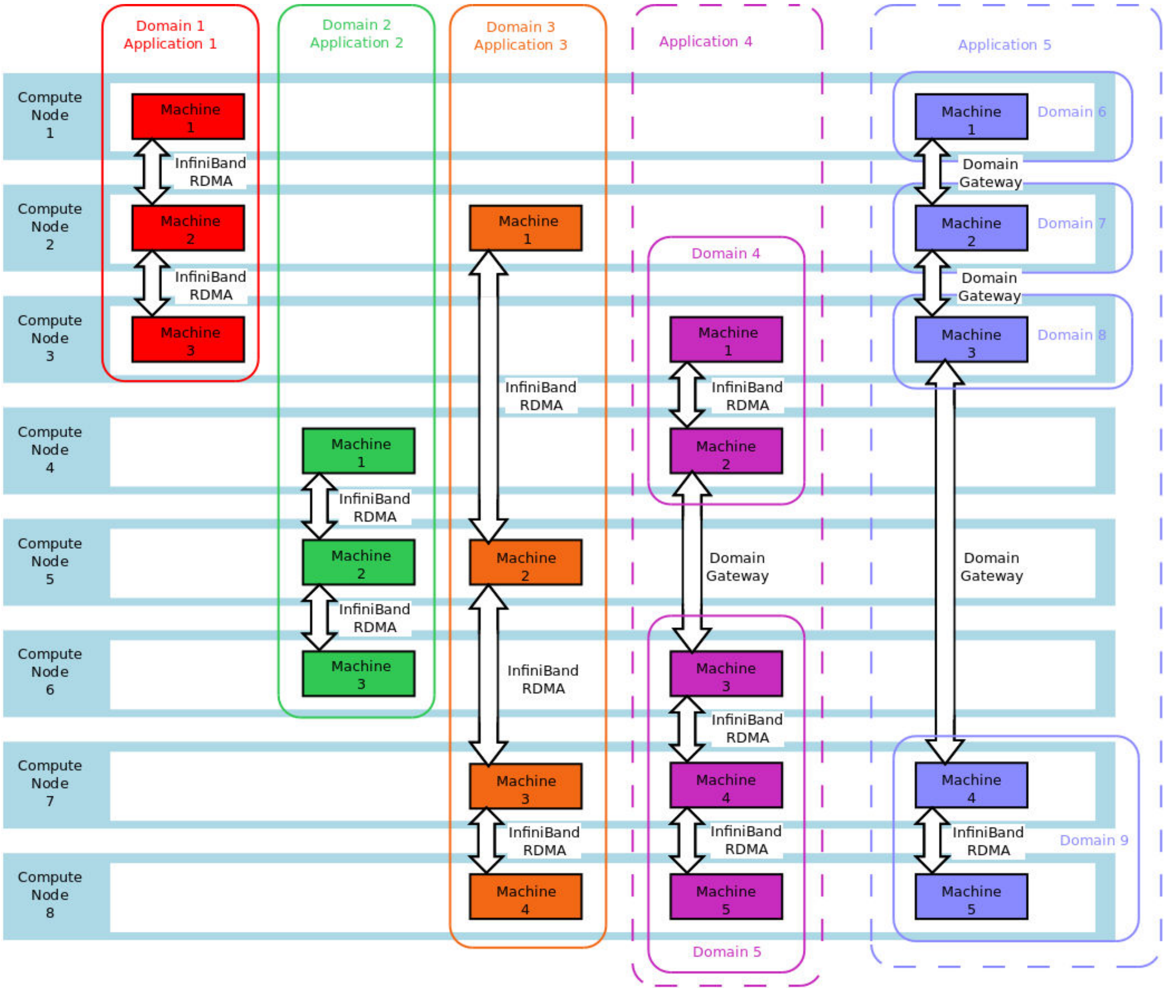


Figure 9: Mixed Deployment Sharing Nodes

In the above diagram we see 5 different applications deployed as clustered domains, single machine domains, and a mix of clustered and single machine domains all sharing a quarter rack of Exalogic. Assuming that application servers are deployed on multiple machines within the domains associated with the application, then high availability should be achieved.

Best Practices to Optimize Performance and Availability

Software Redundancy

One of the key requirements to achieve high availability in a system is redundancy. The engineered systems platforms have built-in hardware redundancy with no single points of failure. To ensure overall application availability, the software must also be configured with no single points of failure.

Multiple Copies of Servers

Tuxedo allows multiple copies of application servers to be configured within a domain. These multiple copies can be configured for both scalability of performance as well as availability of the application. Within a Tuxedo group, this can be easily accomplished by setting the MIN and MAX values for a server to something other than their default of 1. By setting MIN to 5, Tuxedo will automatically boot 5 copies of the server when the application starts up. Thus if an application server fails, the remaining servers can continue operating and handling requests. Setting MAX to 10 allows additional servers to be booted as needed without having to change the configuration.

Clustered Configuration

One of the most powerful features in Tuxedo is its clustering capabilities, sometimes referred to MP mode. This allows a single Tuxedo application to span multiple machines increasing performance and availability of the application, without any change to the application. A Tuxedo cluster is managed as a single entity and all services within the cluster are available to all clients and servers regardless of their location. Application services continue to be available even in the event of a machine failure. By configuring 3 or more machines in a cluster, even multiple simultaneous machine failures can be survived.

The recommended minimum cluster configuration is 3 machines or nodes. This allows a machine to be taken down for maintenance and still have redundancy of application services. If additional machines are needed to handle the load, than a N+M configuration can be created where N machines are needed to handle the load, and M machines are configured for redundancy to improve application availability. Using this approach it is fairly easy to achieve 99.999% availability or more.

Symmetric

Machines in a cluster can be configured symmetrically where all application servers are configured to run on all machines and all machines are running the same versions of software. As long as any given server has multiple copies running on two or more machines, the application should be highly available. Symmetric configurations tend to be easier to manage as machines are then for the most part interchangeable and adding or removing machines from the cluster is simplified.

Asymmetric

Clusters can also be configured asymmetrically where different application components are running on different machines, or different software versions are running on the machines in the cluster, including different operating system versions, different Tuxedo versions, and different application versions. For very highly available applications that need 7x24x365 availability, machines in a cluster can have their software upgraded in a rolling update fashion. A machine is shutdown, its software updated, and then booted back into the cluster. Tuxedo supports running multiple versions of application components in parallel through a feature called application service version. This allows applications to be upgraded incrementally without having to shutdown the entire application.

Automated Operations

One of the keys to improving the availability of applications is to reduce the time to recover from a failure, either planned such as maintenance, or unexpected such as a server dying.

Enabling Tuxedo Automated Operations

Tuxedo provides a variety of automatic operations such as restarting servers or failing over network connections. To the extent these can be controlled by the administrator, they are part of the Tuxedo configuration as initially defined in the UBBCONFIG file. Below are some of the common parameters to set to achieve automated operations.

Server Restart

A simple way to improve application availability is to enable the automatic restarting of a server that fails unexpectedly. This is accomplished by setting the parameters RESTART=Y and MAXGEN>1 in the server section of the UBBCONFIG file. Note, the default is RESTART=N, so no automatic restarting will take place. Besides setting RESTART=Y, it is recommended to set the GRACE and MAXGEN parameters appropriately to prevent restart loops where a server is restarted and almost immediately fails. MAXGEN controls how many times Tuxedo will start an application, so a value of 3 means to restart the application at most 2 times within GRACE seconds.

Another advantage of setting servers to restart automatically is that any requests already in its request queue will be maintained and processed as soon as the server restarts. With RESTART=N, Tuxedo will delete the server's IPC queue, thus causing any requests in its queue to eventually timeout.

Automatic Migration

Sometimes it is necessary to move a group of servers from one machine to another machine or move the Distinguished Bulletin Board Liaison (DBBL) to another machine. This process is called migration in Tuxedo terms. Tuxedo 12c added features to make this migration process automatic if enabled. To enable automatic migration of the DBBL modify the *RESOURCES section of the UBBCONFIG file to define a backup machine for the MASTER entry, and set the DBBLFAILOVER option to the time before DBBL migration occurs. To enable automatic migration of server groups define the machine to failover to in LMID parameter of the *GROUPS section. Set the MIGRATE and LAN options, and define the time before server group migration occurs using the SGRPFAILOVER parameter in the *RESOURCES section of the UBBCONFIG file.

Monitoring Application Performance and Availability

A major challenge in managing enterprise applications is monitoring all the components and ensuring that Service Level Agreements are being met. Typical enterprise applications consist of a front-end web tier that handles static content such as Oracle Traffic Director or Oracle HTTP Server, a user interface tier often behind it that provides dynamic page creation such as Oracle WebLogic Server, a services tier behind that to provide access to enterprise services required to create the dynamic pages such as Tuxedo, and finally a database tier behind the services tier to provide access to persistent data such as Oracle Database. Underneath all of this are the various hardware and operating system platforms that support all these tiers like Oracle Linux and Oracle engineered systems such as Exalogic, SuperCluster, and Exadata.

TSAM Plus

The Tuxedo System and Application Monitor Plus (TSAM Plus) product focuses on trying to address many of the above issues, especially as they relate to Tuxedo application performance. TSAM Plus provides detailed performance metrics such as CPU consumption and response time including end-to-end response time of applications within Tuxedo. Data is collected in real time on the monitored Tuxedo systems and reported to TSAM Plus Manager, a component of TSAM Plus that stores the data and provides a user interface to access and report on the collected data.
Since the data is collected as the application executes, a high volume application could produce enormous amounts of data. In order to limit the amount of data, TSAM Plus provides flexible means of controller what data is collected from which components and how often that data is collected. For example, if service data is being collected, it can be collected from a single server, all

servers in some list of groups, or all servers in the application. As well data can be collected at fixed intervals, so once a second, or on a ratio basis where one out of every five data points is actually collected and stored. In general it is advised to have somewhere between 1000 to 5000 data points collected per second for each TSAM Plus Manager instance.

Besides collecting and reporting on performance metrics, TSAM Plus allows setting service level agreements (SLAs) that can be used to generate alerts. These agreements can be checked even if monitoring data isn't being collected. Thus even though only one out of 100 service invocations may have their performance data collected, each one could be checked to ensure that SLAs are being met. If an SLA is not being met, an alert can be generated to notify an operations staff that SLAs aren't being met, and a Tuxedo event or Enterprise Manager incident can be raised which could then trigger some automated remedial action to try and prevent additional missed SLAs.

Oracle Enterprise Manager

TSAM Plus provides an Enterprise Manager Cloud Control option that allows Enterprise Manager to configure, operate, administer and manage Tuxedo applications. An agent is installed on the Tuxedo systems to be monitored that allows Enterprise Manager to periodically collect Tuxedo application performance metrics, application state, and manage application configuration. The metrics collected can then be used to analyze overall application performance, monitor the availability of the application and its various components, and trigger incidents that can then initiate automated management operations such as starting up additional servers or machines to provide dynamic scale up or scale down capabilities.

One of the surest ways to improve application availability is to automate operations as much as possible. In highly available systems, human error can account for more than 70% of down time. Enterprise Manager along with the TSAM Plus Cloud Control allow many standard operational procedures to be scripted and automated, eliminating many errors of this type. These scripts can be scheduled for periodic execution such as planned maintenance operations, initiated manually for non-scheduled regular maintenance, or triggered automatically as a result of some observed problem such as missed performance or availability SLAs.

By providing a single pane of glass for all monitoring, management, and administration across the entire stack from hardware, operating system, network, database, application server, to web tier, Enterprise Manager allows the operations staff to have a complete view and make the most informed decisions in responding to operational situations. This further helps increase the availability of enterprise applications.

Storage Availability

Storage is a critical resource in most enterprise applications. Storage can range from locally attached hard drives or SSDs to SAN to database. One of the key considerations in the design of storage systems is the availability of the storage when it is holding critical data. To achieve high availability, the data must be stored on at least two devices and there can be no single point of failure that affects both copies. This means that there must be redundant paths to storage at the controller level, the network level, and the software level.

For file system storage, Oracle engineered systems use a ZFS Storage Appliance or ZFSSA. Although each compute node in an engineered system has local SSDs, these drives are intended primarily to hold operating system files and other non-application specific or critical data as the drives are single ported. The ZFSSA consists of two head nodes that share a dual ported JBOD. The ZFS file system used by the appliance provides high levels of data integrity through checksumming all data and metadata, and high levels of availability by maintaining multiple copies of the data through techniques like RAID-Z. The two head nodes monitor each other and in the event of failure, the passive head can take over operations from the failed active head. Each head is connected to the rest of the engineered system via InfiniBand that provides dual network paths to prevent loss of access in the event of a path failure. All of this leads to a highly available file store.

Another primary store for data in engineered systems is the Oracle Database, which is often deployed on Exadata systems to provide extreme performance and high availability. Exadata systems also employ InfiniBand networking which can be used to connect the database to Exalogic and SuperCluster systems. Tuxedo supports using the Socket Direct Protocol or SDP to access Oracle Databases on engineered systems to significantly improve the performance of database access.

Transaction Logs

One of the most critical files in a Tuxedo application is the transaction log. This file is used to maintain a record of partially committed transactions. Once a transaction has been decided to commit, a record is placed in the transaction log recording this decision such that if a failure occurs before all resource managers involved in a distributed transaction have committed their branch, Tuxedo can reliably finish committing the transaction during recovery. If the transaction log is placed on storage that is not highly available, it is possible that the ACID properties of a transaction could be violated, resulting in inconsistent state. In the event of a system failure of a Tuxedo node, when Tuxedo restarts, it reads the contents of the transaction log to determine which transactions were in the midst of being committed when the failure occurred. If the transaction log is unavailable during a restart, then transactions may end up being only partially committed and the application left in an inconsistent state and with resources locked in resource managers.

One option for transaction logs is to place them in the Oracle Database. This ensures that the logs are highly available. To enable storing transaction logs in the Oracle Database, set the TLOGDEV parameter in the UBBCONFIG to point to the database and set the TLOGNAME parameter to a unique table name for the transaction log.

Queue Spaces

Queue spaces in Tuxedo hold the contents of persistent queues. Like transaction logs, they should be placed on highly available storage as the state of persistent queues is often critical to the consistency of an application. On engineered systems, these files should be placed on the ZFSSA to ensure the queue space is accessible to other nodes. This allows the machine or group holding the Tuxedo queue servers to be migrated to another machine in the case of failure, and still have access to the files holding the queue spaces.

Disaster Recovery Topologies

Disaster recovery is becoming more and more important for information systems as businesses have become totally dependent on their enterprise applications. Although planning and implementing a disaster recovery environment is a complex task, but this section will try and point out important considerations and solutions that can help make disaster recovery plans successful.

Active/Passive

The most common disaster recover topology is an active/passive one where one site is handling the entire current load and another remote site is idle and ready to accept load should the active site fail. All application components include any front-end web tier, JEE application server, Tuxedo, and database are all co-located at each site. This topology tends to be easier to manage as only a single site may need to be monitored at any given time and often this topology provides the best performance. The performance benefit primarily comes from having local network connections between all of the components. For Tuxedo applications typically the most important network connection requiring the low latency of a local area network is for database access. How important this is largely depends upon how chatty the services are in accessing the database. Each database request generally requires a round trip to the database, so any additional latency introduced in the network is multiplied by the number of database requests made by a service. In an active/passive environment, the database is co-located and often connected via InfiniBand if the database is being provided by an Exadata system.

To minimize the time required to switch over from the active site to the passive site, it is generally suggested that some form of replication be used to replicate any persistent state such as files and databases. This eliminates the time required to restore the passive system from backups. To replicate files on Exalogic or SuperCluster, the ZFSSA has built in file replication that can be used to replicate files to the ZFSSA at the passive site. For the database, either Oracle Active Data Guard or Oracle GoldenGate can be used to propagate database changes from the active site to the passive site. To minimize the impact of replication, the data is usually replicated asynchronously, meaning that requests are acknowledged immediately before the data has been successfully replicated. In some application scenarios, this can lead to inconsistencies, so in those cases, synchronous replication can be performed.

One thing to consider when replicating data from the active to the passive site is transactional consistency. If the application is using Tuxedo's distributed transaction management facilities, then it's critical that the transaction log and the database replication be synchronized. This can be achieved using synchronous replication, although this may have a dramatic effect on performance as updates won't be acknowledged until the data has been safely replicated to the remote site. If the Tuxedo transaction logs are maintained in the file system, then synchronous replication is the only option that ensures consistency. A better performing solution is to place the Tuxedo transaction logs in the database. This allows asynchronous replication to be used as the state information is all contained in the database and the above mentioned replication techniques guarantee consistency.

A major drawback of an active/passive disaster recovery model is that the passive site isn't handling actual load. At most it's just accepting replication updates, but not actual user load. If there is a problem at the passive site such as a configuration, software, or hardware problem, it may not show up until trying to failover. Another drawback to active/passive deployments is that a significant amount of hardware, potentially half of the total hardware, is sitting idle. One potential solution to this is to run read-only workloads on the passive sites such as reporting tasks.

Active/Active

In an active/active disaster recovery deployment model, active load is handled at both sites. An external load balancer is used to direct traffic to each site. This model has the major advantages of utilizing all hardware all the time and ensuring that both sites are operating normally. The biggest difficulty in achieving an active/active deployment is handling persistent data such as files and databases. In an ideal situation, these would be logically partitioned in such a way that the partitions are only accessed locally, with some sort of replication updating the other site such that in the case of a failure, the other site could take over and handle the load for the failed partition. This has the advantage of utilizing local network access to the data, but still achieving high availability for the partitions in the presence of a failure. While partitioning of data is ideal, it's likely that some data can't reasonably be partitioned. As long as the access to this non-partitioned data is less frequent, an active/active model can still be deployed.

Common practice for active/active Tuxedo deployments is to have a Tuxedo cluster at each site configured to support the same set of services. This provides high availability at each site minimizing the number of times failover to the remote site needs to occur. If partitioning is being used and the load balancer is unable to properly route some or all requests to the correct site for the required partition, the sites can be linked using the Tuxedo domain gateway and traffic routed to the correct site using Tuxedo data dependent routing feature. This feature allows requests to be routed to specific domains or groups of servers depending upon the contents of the request. While this might incur an extra network hop for the request and the reply, it allows the service to execute on the servers where the partition resides. Since Tuxedo services often make multiple database requests, this allows the servers with local database connections to process the requests.

An alternative option to deploying clusters at each site is to create a split cluster that spans both sites. While this is supported and can also use data dependent routing to have requests execute on the servers with local connections to a partition if needed, it isn't generally recommended. The likelihood of having a partitioned cluster increases significantly when the machines in the cluster are distributed geographically. Although request processing still continues in a partitioned cluster, the partitioned portion of the cluster can't be managed until the cluster reforms.

Another consideration for active/active deployments is failover procedures. These are necessary even if just for transaction logs. Assuming transaction logs at each site are being replicated to the remote site, in the event of a failure, the transaction logs need to be recovered to complete any in-flight transactions.

Summary

Tuxedo provides the most scalable, available, and reliable platform for deploying mission critical enterprise applications. With support for C, C++, COBOL, Python, Ruby, PHP, and Java, Tuxedo allows application developers to choose the language that best suits their particular requirements. Combine this with the state of the art hardware found on Oracle engineered systems such as InfiniBand to connect applications to the most scalable database appliance found in Exadata and applications can now scale from less than a single core to 720 cores in a single rack to 5,760 cores in multiple connected racks. Add in the performance optimizations built into Tuxedo for Oracle engineered systems, and you have an unbeatable platform for mission critical applications.

ORACLE®

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

White Paper Mission Critical Applications with Oracle Engineered Systems and Tuxedo
May 2014
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

Hardware and Software, Engineered to Work Together