ORACLE
FUSION MIDDLEWARE

An Oracle White Paper
July 2012

Achieving High Availability with Oracle Tuxedo

ORACLE

Introduction

On-Line Transaction Processing (OLTP) applications support the core processes of much of the world's businesses. These systems are high volume and mission critical, where failure can have significant consequences to the survival of the business. The following requirements are typically associated with OLTP applications:

- Capacity to support thousands of concurrent users
- Capacity to handle large volumes of data
- High transactional throughput
- Predictable, short response times
- High data integrity and security
- Concurrent database access
- High (often 7 x 24) application availability

A critical requirement for most OLTP systems is that they operate 7 x 24 despite any component outages or required application upgrades. Downtime is a critical problem for businesses, who use computing systems as backbone of their operations, since downtime translates into lost money, unhappy customers and lost business.

Oracle Tuxedo provides numerous built-in capabilities that customers can use to deploy and run highly available application services. Tuxedo can also be combined with Oracle RAC and 3rd party clustering solutions such as with Veritas Cluster Server, HP’s MC ServiceGuard, and others to further increase the availability and robustness of the deployed applications.

This white paper summarizes the high availability features of Oracle Tuxedo. It focuses on system availability in cases of hardware and software failures within the data center. It does not cover environmental, operational and remote client machine failures.

Tuxedo High Availability

System Availability

The availability of a system is defined as the percentage of time during which that system is available. Availability is reduced by two factors: the rate at which a system fails and the recovery time. The following two measures quantify these factors. Mean Time Between Failures (MTBF) is the average time the system runs before it fails and is a measure of system reliability. Mean Time to Repair or Recover (MTTR) is how long it takes to fix the system after it fails or for the system to automatically recover. So availability can be defined as

$$\text{Availability} = \text{MTBF}/(\text{MTBF} + \text{MTTR})$$

Thus, availability improves when reliability increases and recovery time decreases.

Causes of System Outages

The various outages in generic computer systems and applications can be divided into planned and unplanned. Planned outages are caused when the system has to be taken offline for hardware or software upgrade, backup, or other sorts of maintenance. Modern systems can be architected to completely eliminate the need for planned outages through combination of redundant and hot-pluggable hardware components, online OS updates, and ability to upgrade applications while in operations. Upgrades that are transparent to Tuxedo will not affect the applications, and for those that are not transparent, Tuxedo can shift running services among multiple nodes to support a rolling upgrade without an outage.

Unplanned outages are caused by some sort of failure in the operating environment, and these can be divided into the following classes:

- Environmental Failures (e.g., power, communications, air-conditioning, etc.)
- Operations Failures (e.g., human errors in configuration, operation, etc.)
- Hardware Failures (all hardware devices like processor, memory, I/O controllers, network equipment, etc.)
- Software Failures (operating system, database, transaction monitor, application)

In recent years, the increased robustness of hardware and software has made operations, administration, and maintenance the primary cause of downtime as can be seen in table 1.

TABLE 1. DOWNTIME ANALYSIS

PERCENTAGE	CATEGORY	CAUSE
20%	Hardware	Disk, Network, Memory, Processor
30%	Software	Application, Middleware, Database
50%	Operation,	• Scheduled maintenance

administration, maintenance	• Unintentional – Wrong configuration, shutdown wrong host, disconnected wrong cable, etc. • Intentional – Hacking, denial of service attacks, etc

Maximizing System Availability

High Availability (HA) systems aim to shield users from an impact of a component failure and reduce the application environment's MTTR by automating recovery from these failures. To understand the methods for this, we must first define the various levels of availability.

Normal Availability systems are general-purpose computers that have no hardware redundancy or software enhancement to provide fault-processing recovery. They require manual, human intervention to identify and repair failures and restart the system before resuming normal operations.

High Availability systems use hardware and software redundancy together with health monitoring, often through the use of loosely coupled computer clusters, which, provide redundancy at the node level. The cluster is managed by software (a cluster manager) that provides automated fault detection and correction procedures. These clustered systems require no manual intervention to identify a failure, execute a procedure to bypass the affected computer and notify system administrators. They respond automatically to most kinds of failures and restore access to services with minimal interruption. There are two distinct high availability models for client-server architectures:

- Replicated Services Model

This model utilizes distributed applications and distributed databases on multiple servers. The application services are available on more than one server, so an individual failure will not prevent the request from being processed by an alternate server offering the same service. The data is replicated to some or all of the servers, or is otherwise visible from more than one system (e.g. via a parallel database system like Oracle RAC). Therefore, when a server failure occurs, the data and applications are accessible from an alternate node.

- Failover Model

This model utilizes duplicate hardware configurations in which one system has the role of a primary server for data and application services, and the other is a backup server that monitors the state of the primary system. When the backup node detects a failure on the primary server, it takes over the role and identity of the primary.

Fault Tolerant systems consist of proprietary, expensive, and tightly coupled duplicated components. Fault handling capabilities are integrated into and become a function of the operating system. These systems require no manual intervention to identify a failed component and execute a procedure to avert a system failure. They have spontaneous and fully automatic response to failures and provide completely uninterrupted services.

Tuxedo platform and infrastructure

A TP Monitor provides an execution environment for server-based OLTP applications which run continuously in production. Oracle Tuxedo is the leading TP Monitor for open systems, providing

distributed Transaction Processing and messaging middleware. It combines the capabilities of an application server for hosting COBOL, C, and C++ applications with service bus capabilities for high-speed reliable messaging, supporting synchronous, asynchronous, conversational, and event-based publish-subscribe messaging models. It features a high-level API for building distributed application components connected via message-based communications. Components execute in a managed server environment (containers) implemented by core Oracle Tuxedo services. These services implement a sophisticated set of transaction and application management functions, and comprehensive distributed systems administration including:

- Transparent two-phase commit for data integrity and atomic updates
- Transaction and error logging
- An administration framework for managing centralized or distributed transactions
- Fast-cache access methods, automatic server spawning, buffer management, and routing capabilities to optimize transactional throughput

Tuxedo provides rich built-in high availability features, which include:

- Built-in redundancy, replication, distribution, clustering
- Monitoring and fault detection for server and client processes with buddy system and heartbeat mechanisms
- Automatic failover migration for server processes in case of unavailable node in a multi-node domain or cluster environment
- Fault recovery features: restart, re-route, and failover
- Failover mode of data dependent routing
- Distributed transactions with ACID properties

In addition, Tuxedo service virtualization capabilities support transparency of implementation, location, replication, failover, and recovery, and enable dynamic routing, load balancing, and failover/failback. Tuxedo applications can be deployed in a single machine domain, multi-node domain or cluster, and multiple domains as required by separation of application concerns, security zones, or other considerations.

Tuxedo clustering – MP Domain

Tuxedo provides built-in clustering capabilities, which enable customers to deploy Tuxedo applications across multiple nodes without any additional clustering software or hardware. In a Tuxedo cluster (also known as MP domain), services can be transparently shared and load balanced across a mixed set of platform architectures. This environment has no single point of failure and can be dynamically administered to add/remove machines, servers (processes), and services for continuous operation. These capabilities also enable rolling upgrades of the Tuxedo platform and the application, without impacting end-user availability of the services.

Tuxedo domains can be used to achieve either of the two HA models described above:

- **Failover model** using Active/Passive configuration

This commonly used model requires a mirrored configuration, which uses twice the hardware and software requirements but supports only half of the total workload capacity. The data replication (files, databases) must be provided by facilities like Oracle Data Guard or similar 3rd party data replication solutions. Unlike an Active/Active configuration where the node taking over is already running services, the Passive system has to go through a startup process to take over a failing node, and this can potentially create reliability issues unless tested periodically.

- **Replicated services model** using Active/Active configuration

In this configuration Tuxedo can support multiple domains of single or multiple machines. Because all resources are in use, the ability of a node or domain to take over the workload of a failing node or domain is assured, though its performance may degrade. Active/Active configuration also provides an additional benefit in that the system can be scaled by replication of services and data dependent routing. This combination of highest availability and scalability makes Active/Active configurations the best choice for large scale OLTP workloads.

Terminology

Node - one or more CPUs with shared memory, running one copy of an OS, having network connections, and disk storage.

Cluster or **Domain** -one or more Nodes possibly sharing disk storage and networks, and connected by private interconnect over which control information and heartbeats can flow. From an application perspective a Tuxedo domain is a set of Tuxedo system, client, and server processes administered as a single unit from a single Tuxedo configuration file. A Tuxedo domain consists of many system processes, one or more application client processes, one or more application server processes, and one or more computer machines connected over a network. A Tuxedo domain may provide ATMI services, CORBA objects, or both.

Logical Host - a Virtual Node which is defined to reside on a Physical Host (Node) and may be migrated to another Physical Host. More than one Logical Host may be defined on a Physical Host.

Logical Machine - the terminology Oracle Tuxedo uses to refer to a Virtual Node. In a Oracle Tuxedo configuration, a Virtual Node is identified by a Logical Machine ID (LMID).

Master/Slave Node -In the Tuxedo System, the computer configured as the master for an application is called "Master Node". It contains the master copy of the configuration and is the computer where DBBL runs. The other nodes are called "Slave Nodes". One of the slave nodes can be configured as "backup node", where the master node can be migrated to, and becomes the new master node.

Backup Master - the Oracle Tuxedo Logical Machine which can inherit the Master role, i.e. taking over in the event of a failure of the Oracle Tuxedo Master Logical Machine.

Primary/Backup Node: For each Tuxedo server group, two nodes - one primary node, one backup node - can be configured. Then this server group will be booted in the primary node at first, and can be manually migrated to the backup node.

Oracle Tuxedo in High Availability Configurations

Oracle Tuxedo is the industry's #1 platform for distributed transaction processing. It provides mainframe-class scale and performance on open, distributed systems for software written in C, C++, and COBOL, and is the premier platform for building new eXtreme Transaction Processing (XTP) applications and re-hosting mainframe applications on open systems platforms and grid infrastructures. Oracle Tuxedo provides cost-effective reliability, extreme scalability up to hundreds of thousands of transactions per second, and investment protection by extending the life of existing IT assets as part of modern architectures such as SOA. Oracle Tuxedo is Oracle Fusion Middleware's strategic transaction processing product.

As the #1 transaction processing application platform for open systems, Oracle Tuxedo has long maintained a history of setting the standards for Open OLTP technology, application performance, portability, scalability, and availability. It offers rich features that enable the stringent demands of distributed mission-critical applications. In particular, Oracle Tuxedo provides powerful high availability capabilities relative to traditional TP Monitors. The key to this strength is centralized system configuration providing key parameters of system components for use by the Oracle Tuxedo run-time monitor, diagnostic, and repair facilities. The configuration captures the relevant characteristics of all components: server processes, service instantiations, client processes, processor nodes, network connections, resources, and, especially, reconfiguration options.

The following subsections examine the Oracle Tuxedo features which intrinsically contribute to high availability. For additional background information on Tuxedo or specific features not covered here see http://www.oracle.com/tuxedo.

Administrative Infrastructure

Oracle Tuxedo supports an architected solution to the critical problem of distributed application administration. The management infrastructure within Oracle Tuxedo implements a Manager/Agent model, built around a Management Information Base (MIB).

Oracle Tuxedo's MIB enables application administrators to define in one centralized application configuration the hardware, software, and networking resources that make up an application. Application designers can state where servers and services should run, as well as where they should be migrated in the event of a processor failure. They may assign various characteristics to the application's servers, including scheduling information, process recovery criteria, and time-out periods.

Administrative interfaces implemented atop the MIB include a comprehensive command-line tool, a programmatic interface (including scripting), and an SNMP agent for integrating Oracle Tuxedo as a managed application within a larger administrative environment. A GUI-based administration application also takes advantage of the MIB by providing a single point of high-level control over the Oracle Tuxedo environment.

In addition to allowing all system parameters to be polled (and modified where permitted), the MIB supports a full range of System Events. These events are posted whenever significant state changes occur within the Oracle Tuxedo environment.

Oracle Tuxedo also has a rich set of internal mechanisms which provide the runtime support for application availability:

- BBL – The Bulletin Board Liaison is the node monitor, responsible for overseeing all processes (application and administrative) on a node.
- DBBL - The Distinguished BBL is the master monitor, responsible for overseeing the BBL on each node. Also for networked applications, a backup node may be designated for the DBBL.
- BRIDGE – these servers provide inter-node communications in a networked application.
- TMS – The Transaction Management Server is a transaction manager server dedicated to a particular DBMS (or other resource, e.g., MQ queue) when distributed transaction processing is employed.

Other, optional, system servers such as various Client Handlers, Event Brokers and Queue Managers may be in use as well. All of these servers are started at system boot time in accordance with the central configuration.

Failure Detection & Automatic Recovery

Run Time Facilities

Oracle Tuxedo run-time administration provides automatic detection and correction of software faults. The primary facilities are illustrated in Figure 1. Nodes, network connections, application servers, clients as well as the Oracle Tuxedo administrative servers themselves (including DBBL, BBL, and BRIDGE) are all monitored. Additionally, an attempt to correct failures, without operator intervention, is always performed. Errors are reported by posting System Events which may be subscribed to by applications, thereby extending failure detection coverage and enabling application-driven recovery. All errors are also recorded in the Oracle Tuxedo error log (ULOG), itself accessible by applications. The log files are maintained on a per machine basis, but may be consolidated for monitoring using Oracle Manager Log Central.

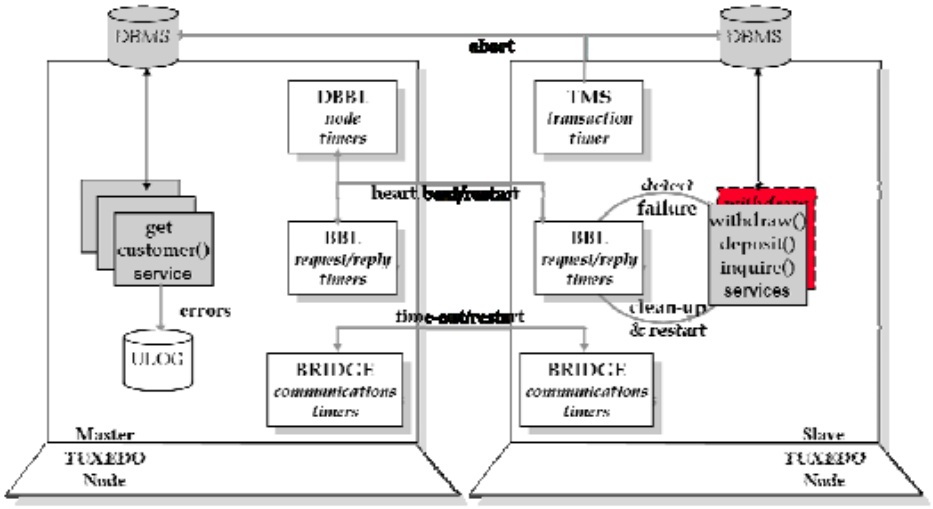

Figure 1. 2-node Tuxedo Cluster with Key Servers

Node Status Checks

The DBBL expects a periodic heartbeat from each BBL. If the heartbeat does not occur, the master will attempt to restart the BBL on the afflicted node. If the BBL cannot be restarted, the node is marked as "partitioned" (i.e., unavailable). The BRIDGE server will also cause a node to be marked as partitioned if a communications time-out occurs and communications cannot be reestablished. (See the Network Connections section for more details.) Because the permanence of a failure is not known, partitioned nodes are not automatically removed from service. The occurrence of the partition is reported via a System Event and recorded in the error log. If the problem is a transient communications outage, connectivity will be automatically restored once communications are re-established. If the problem is severe, the administrator will remove the node from service via administrative command. The nodes that remain will continue to operate as one application.

Oracle Tuxedo also provides automatic migration of node fault. In a cluster environment that configured migration and failover functionality, in case of the master node down or network issue, the BBL at backup site will start new DBBL and form a new cluster when a specific check threshold reached. After the network is recovered, the DBBL on original node checks DBBL failover is already done, and it will exit. The original node rejoins this cluster and reform a cluster.

Server Status Check

The BBL periodically checks the availability of each application server (process) executing on its node. If a failure is detected, Oracle Tuxedo will abort any outstanding transactions and may be configured to automatically restart the application server. Only the service request being processed is lost; in this case, the requester is properly notified and may take appropriate action (e.g., re-try). Any waiting service requests will be processed as usual. Further, Oracle Tuxedo may be configured to automatically invoke an application defined process in conjunction with re-starting the application server. Note that

if the application requires automatic resubmission of failed service requests, then persistent queues may be employed.

Oracle Tuxedo also provides automatic server group migration of node fault. In a cluster environment, in case of a node down or network issue, if a server group that is configured as automatic migration on this node, this server group will be automatically migrated to its backup node. The prerequisite is that the corresponding binaries must have already deployed in backup node.

Administrative Self Checks

The DBBL and BBL periodically check the status of each other and, if necessary, automatically restart each other. Loss of the master node requires migration of the DBBL as specified in the central configuration. This migration can be performed manually or automated with Oracle Enterprise Manager Grid Control or 3rd party clustering software.

Client Status

The BBL checks the status of client processes and effects cleanup upon detection of abnormal termination. In the case of native clients (running directly on a Oracle Tuxedo server node), this is determined by process status. Native client process failures are also detected via a timeout maintained by the BBL on reply queues.

In the case of remote clients, detection of abnormal termination is based on an inactivity timeout. (Spurious timeouts are prevented by using a "keep-alive" protocol to the client.) Upon detection of a client failure, any communications in progress are terminated and the client state is cleaned up. The user has to re-logon and resume work. However, for Tuxedo workstation client (WS), multiple Workstation Listener (WSL) server addresses can be specified, allowing WS client to failover among them in a round-robin manner.

Network Connections

BRIDGE processes maintain inter-node communication via a hierarchy of multiple network addresses as specified in the central configuration. The BRIDGE uses the highest priority connection available, automatically failing over to the next lower priority connection in the event of a network outage, and failing back to the higher priority connection when it becomes available. (Additionally, the BRIDGE process will use a second network address at the same priority if the first one becomes blocked, facilitating increased throughput for heavily loaded networks.) Since these network addresses may be supported by multiple network interfaces, the failure of a host adapter card, connector, or network cable need not result in a loss of communication.

Communication between nodes is monitored by timeouts maintained by the BRIDGE servers. Detection of a failure on any given connection automatically initiates an attempt to reestablish communications. If communications via every network address are lost and cannot be reestablished, then the nodes that remain in communication will continue to operate as an application. Note in particular that the application may be partitioned, with each partition continuing to operate.

Transactions

With Distributed Transaction Processing enabled, each global transaction is monitored by a timeout. If the threshold is exceeded, the transaction is aborted. Once a transaction is begun, Oracle Tuxedo tracks all DBMSs (and other XA-compliant resources, such as JMS or MQ queues) that are accessed. If a timeout occurs at any time prior to completion of the transaction pre-commit, or, if the transaction is explicitly aborted by the application, all DBMSs are instructed to roll back the updates made within the transaction scope. Rollback will be carried out even if a participating node fails. The rollback will in this case occur when the node is brought back on-line. (See also the discussion of data integrity.)

Application Error Returns

Errors detected by an application can be reported to the administrator via System Events and recorded in the error log. These System Events provide a consistent error reporting facility which can be monitored manually by the administrator or, more typically, automatically by system management software.

Maintenance of Data Integrity on Failure

Oracle Tuxedo implements the X/Open Distributed Transaction Processing (DTP) standard. The Transaction Management Servers (TMSs) are responsible for assuring, by way of a two-phase commit protocol, that global transactions are atomic. Global transactions may encompass multiple, heterogeneous DBMS running on multiple, heterogeneous nodes. TMSs, which are replicated and redundant, are also responsible for managing transaction aborts and recovery of transactions during failure restoration.

All communications with the DBMS are via the X/Open standard XA interface. XA provides functions for beginning and ending work on behalf of a global transaction, for pre-committing, committing or aborting global transactions, and for recovery. It is the responsibility of the DBMS vendor to provide the XA library.

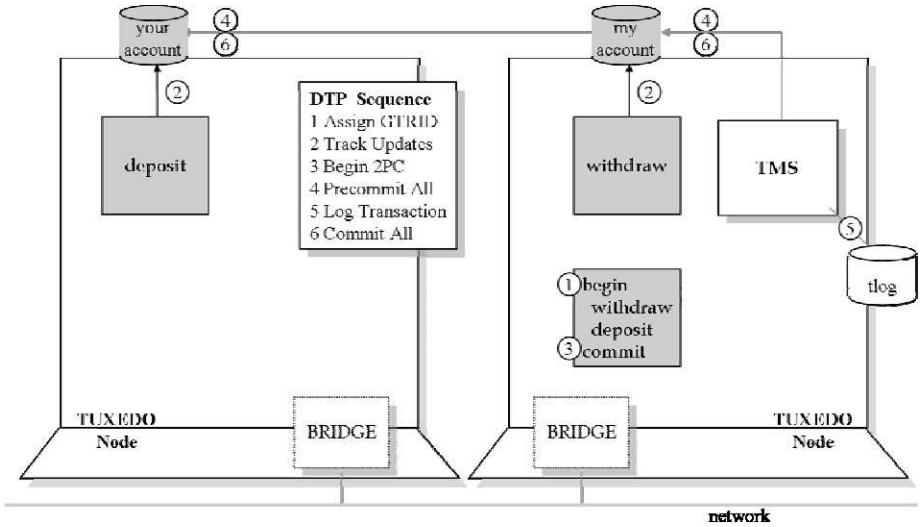

Figure 2. Data Integrity in Distributed Transaction Processing

Transaction Coordination Services

As the communications platform, Oracle Tuxedo tracks DBMSs and other resources that are updated during the course of a transaction. At transaction commit, coordination is provided via a two-phase commit protocol with the resources utilized. The X/Open DTP "TX" API for transaction management is supported: *tx_begin*, *tx_commit* and *tx_rollback*. Optionally, services can be configured to automatically begin and commit transactions, thereby eliminating the need for explicit programming.

When a transaction is initiated a global transaction identifier (GTRID) is generated. As service requests are processed, Oracle Tuxedo instructs the associated DBMS to begin and end work on behalf of the GTRID. It is the responsibility of the DBMS to map the GTRID onto a local transaction. When the transaction is committed, the list of DBMSs involved is handed to a TMS. The TMSs then use the XA interface to execute the two-phase commit. Upon completion of the pre-commit phase the GTRID state is logged to disk so that any interruption of the commit phase can be recovered.

Transaction Recovery Services

Recovery from failures is also provided by the TMSs. For example, if a node failure occurs, all outstanding transactions will be properly committed or aborted when the node is put back in service. The combination of the transaction log and an XA primitive to query the state of a DBMS provide the necessary underlying support.

Transparency

Transaction coordination and recovery is *completely transparent to the application*. The Transaction Manager utilizes the X/Open DTP XA interface provided by the DBMS vendor to track, coordinate and recover transactions.

Maintaining Operations during Failure

Oracle Tuxedo provides a powerful set of tools for maintaining operations in the event of a failure. Additionally, an Oracle Tuxedo application can utilize replication features with a minimum of application code to provide a work around in case of a component failure.

Replicated Services with a Single Database

Oracle Tuxedo provides for arbitrary replication of services. A simple but effective defense against a node failure is to replicate critical services on two or more nodes. This configuration requires concurrent access to the database. This is typically achieved using a parallel DBMS product such as Oracle OPS. This technique is especially useful when Data Dependent Routing (DDR) is employed. During normal operation, Oracle Tuxedo will distribute service requests over the nodes. When a node fails, it is taken out of service using an administrative command. Once the partitioned node has been "cleaned" from the runtime MIB, the remaining nodes will assume the load.

An equivalent alternative is to offer distinct sets of services on each node during normal operation, and utilize the Oracle Tuxedo migration feature upon failure. That is, when a node fails, the set of services, which must be encapsulated in one or more administrative groups, are migrated to an operational node. One problem that must be dealt with, however, is the case when an application server node fails and DTP is employed. When the group is migrated, Oracle Tuxedo will use the open string recorded in

the central configuration to open the data base. Depending on the DBMS product, attempting to open a failed database may fall back to opening the alternate database. If so, all is well. If not, the configuration must be changed by administrative command. Such change, of course, can be automated.

Clients running on the failed node must be reconnected. If the clients are running as native processes on the failed node, the user must logon to an operational node. For remote clients hosted by a handler on the failed node, "reconnect on failure" logic must be implemented. Note, however, that remote clients connected to other nodes are unaffected.

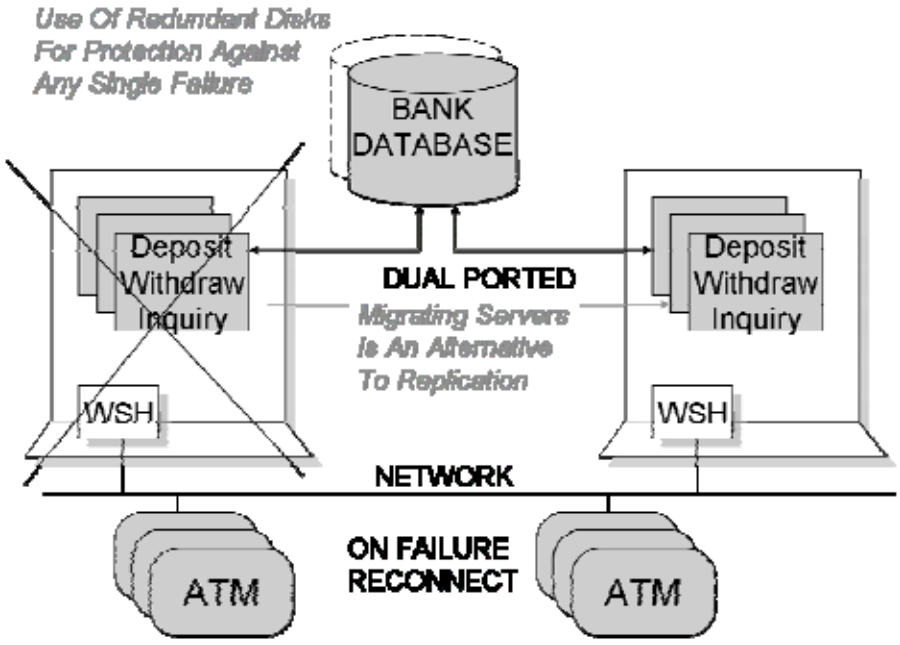

Figure 3. Replicated Servers with a Single Database

Replicated Services with Replicated Data Servers/Oracle RAC

Modern DBMSs offer the capability to replicate data bases. Techniques vary, but the end result is that updates to one copy of a data base will be replicated in the other. Application programmers can take advantage of the replication features of both Oracle Tuxedo and the DBMS to protect against any single failure. The technique is to simply replicate services on two or more nodes as discussed above.

The application should detect the loss of one data server and arrange (indeed, if even necessary) to open the alternate data server (e.g. through an environment variable) and simply abort. When Oracle Tuxedo restarts the application server, it will reconnect to the operational data server and continue processing service requests. This technique also works when DDR is employed.

The Oracle Real Application Clusters (RAC) support clustering of machines that utilize replicated Oracle database services accessing the same Oracle database. Oracle RAC provides the ability to concurrently access the same Oracle database from instances physically located on multiple Oracle server machines, and offers the ability to failover unsuccessful database instances to alternate locations.

RAC is a key piece of Oracle's grid computing products and offers unmatched database scalability and availability using clusters of database servers. In addition, Oracle Database features Automatic Storage Management (ASM), which enables a shared pool of storage for clustered and non-clustered database environments. ASM provides dynamic provisioning of storage, as well as simplified and automated storage administration.

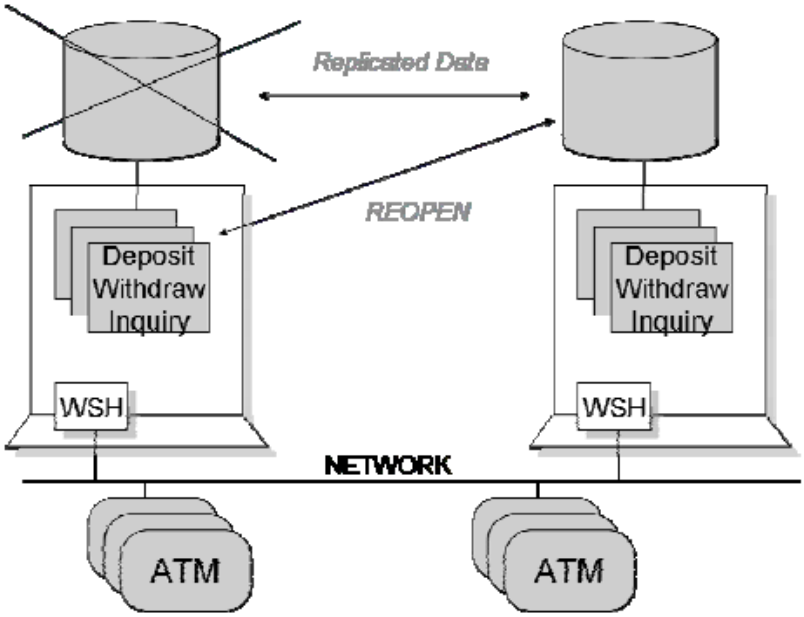

Figure 4. Replicated Servers with Replicated Database

A particular example of such deployment architecture is Oracle's Maximum Availability Architecture (MAA), which provides superior availability by minimizing or eliminating planned and unplanned downtime at all technology stack layers including hardware or software components. Data protection and high availability are achieved regardless of the scope of a failure event - whether from hardware failures that cause data corruptions or from catastrophic acts of nature that impact a broad geographic area.

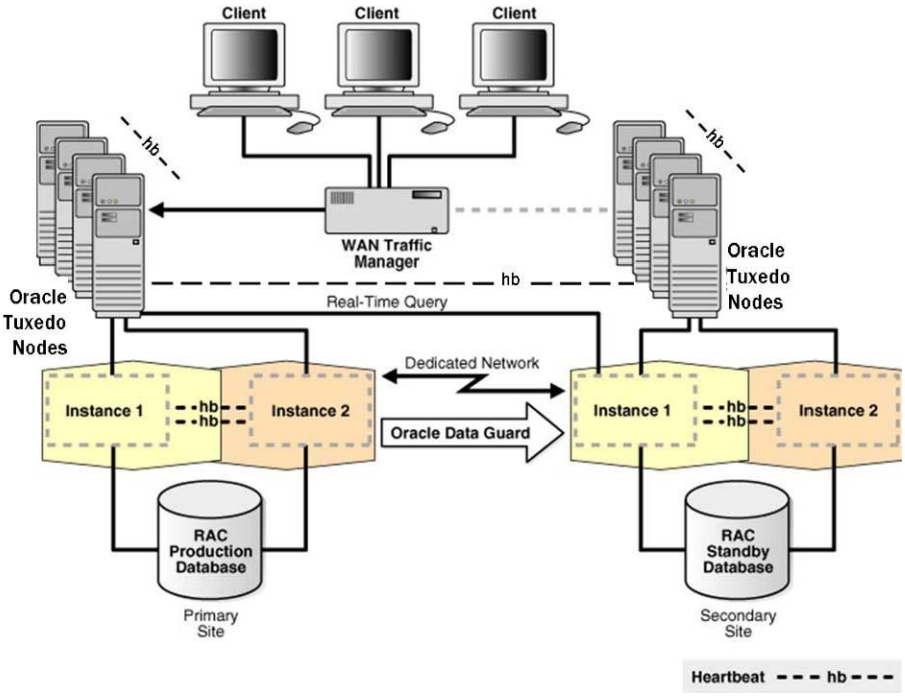

Figure 5. Oracle's Maximum Availability Architecture

MAA also eliminates guesswork and uncertainty when implementing a high availability architecture utilizing the full complement of Oracle HA technologies. MAA Best Practices are described in a series of technical white papers and documentation to assist in designing, implementing, and managing optimum high availability architecture.

Replicated Application

Oracle Tuxedo provides mechanisms by which application and data servers can be replicated, thereby defending against any single node failure. The technique is to maintain identical primary and standby application services on distinct nodes accessing distinct DBMSs. If the primary fails, the standby continues operations (or vice versa).

Synchronization of the DBMS must be assured. Unlike the Replicated Data Server scenario, database replication mechanisms are not used. Instead, the application creates two streams of identical requests which will cause the same processing to execute on both nodes (and ultimately, the same changes to be applied to both DBMSs). This can effected using the Distributed Transaction Processing and Reliable Queuing features.

Using DTP, the application asynchronously invokes both a primary and backup service under the umbrella of a single transaction. The asynchronous invocation assures that processing will occur in parallel, thereby providing high throughput. The administrator can employ a simple naming convention to generate backup service names. Finally, a thin veneer can be written to hide the parallel invocations from the application programmer.

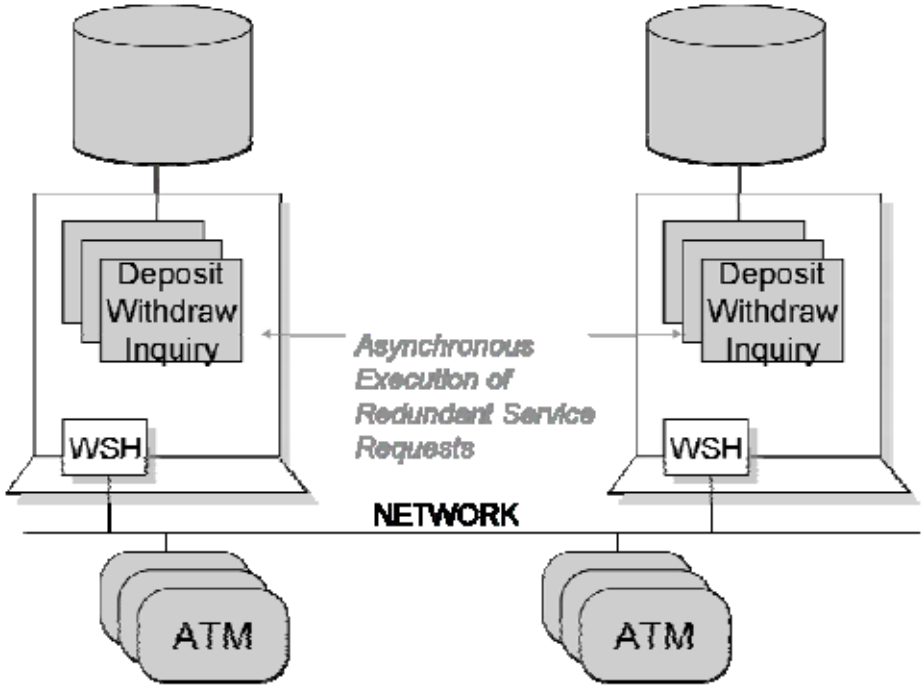

Figure 6. Replicated Application

The actions taken on failure are application dependent. A simple strategy is to post the status of the primary and backup systems so that in the event of a failure operations will continue on the remaining operational system. The real challenge is re-synchronizing the DBMS state after repairs are completed. Generally, this will require a quiescent period. One re-synchronization technique is to use the Reliable Queue feature of Oracle Tuxedo's System Q. During a failure, instead of two parallel on-line service requests, the application enqueues one of the service requests (the one destined for the failed node) on a reliable queue (this can also be done in parallel with the on-line request). Once the failed node can be repaired and brought on-line, any enqueued service requests will be applied.

Restoring Operations

Restoration of operations occurs automatically in certain cases. More generally, an administrative command that reverses the maintenance action is required.

Automatic Restoration

As discussed under Automatic Recovery, recovery from a failure of an application server, an administrative process, a client process, a network disconnects or a transaction time-out is automatic.

Restarting Nodes

Restoration of a node taken out of service is done by applying a boot (or activate) command to a specific node. When a node comes on-line, any incomplete transactions are automatically committed or rolled-back.

Restoring Migrated Groups

Restoration of migrated services is done by migrating the services back to the original node after that node itself has been restored.

Client Processes

Client processes may need to log on to restored nodes.

Conclusion

Oracle Tuxedo is Oracle's flagship product for transaction processing and distributed COBOL, C, and C++ applications. It provides reliability, availability, scalability, dynamic system reconfiguration to application and hides the heterogeneity of computers, databases and networks. In addition, Oracle Tuxedo provides excellent fail-over capabilities and can provide a very powerful highly available solution with automated fail-over of transaction processing application components to the surviving nodes and with minimal impact to the user.

Oracle has pioneered many techniques of server failover that provide IT departments with automatic failover capabilities for several server types. For example, Oracle Database RAC, Oracle WebLogic Server, Oracle Tuxedo and Oracle Coherence clusters can withstand failures of several servers within a cluster and still remain in operation. IT departments can simply remove failed servers from service and repair or replace them and add them back to the server grid. Load balancing, work load management and overload protection, and automatic migration and failover of services or whole servers ensure applications stay up and running.

Choosing and implementing the architecture that best fits the availability requirements of a business can be a daunting task. This architecture must encompass appropriate redundancy, provide adequate protection from all types of outages, ensure consistent high performance and robust security, while being easy to deploy, manage, and scale. Needless to mention, this architecture should be driven by well-understood business requirements.

To build, implement and maintain such an architecture, a business needs high availability best practices that involve both technical and operational aspects of its IT systems and business processes. Such a set of best practices removes the complexity of designing high availability architecture, maximizes availability while using minimum system resources, reduces the implementation and maintenance costs of the high availability systems in place, and makes it easy to duplicate the high availability architecture in other areas of the business.

A well-articulated set of high availability best practices that encompass high availability analysis frameworks, business drivers and system capabilities, improves operational resilience and enhances business agility.

Achieving High Availability with Oracle Tuxedo
July 2012

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