UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

¿SOLUCIÓN DE ALTA DISPONIBILIDAD DE BASE DE DATOS POR *HARDWARE* O POR *SOFTWARE*?

EDGAR FELIPE ALEJANDRO LÓPEZ LEMUS

Asesorado por: Ing. Moisés Fabriccio Díaz Arévalo

Guatemala, julio de 2005

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Backup Es una copia de la base de datos en un medio removible.

Cluster Grupo de sistemas independientes que trabaja como un único sistema.

Commit Comando de base de datos que significa realizado, hace que los cambios en los datos sean permanentes.

Control files Contiene las entradas que especifican la estructura física de la base de datos Oracle.

Datafiles Contienen todos los datos de la base de datos Oracle.

DDL *Data definition language* o lenguaje de definición de datos.

DML *Data manipulation language* o lenguaje de manipulación de datos.

Downtime Tiempo fuera de servicio planeado o no planeado que pueden experimentar los sistemas al no estar disponibles en el caso de fallas en sus componentes.

Failback Proceso de rehabilitar operaciones en el nodo primario cuando ya ha sido reparado.

Failover Recuperación de una falla, regularmente se ejecuta hacia un nodo activo.

Fault resilient Resistencia a fallas.

Fault Tolerance	Tolerancia a fallos. Habilidad de un proceso para manejar interrupciones sin tener que afectar los datos o la integridad del proceso.
Hardware	Toda pieza física y palpable que conforma una computadora.
Host	Nombre genérico que se le da a una computadora a la cual se conectan varios usuarios
Instancia	Es la combinación de los procesos en segundo plano y los *buffers* de memoria. Cada instancia tiene su propio *SGA*.
Interconexión	Enlaces de comunicación entre los nodos del cluster.
LAN	*Local Area Network* o red de área local.
Latencia	La diferencia entre tiempo cuando el dato origen es cambiado y cuando el dato destino refleja ese cambio.
Logminer	Es un utilitario y básicamente lo que hace es leer o recorrer los *redo logs*, viendo su contenido para efectos de búsqueda de transacciones, información, etc.
MAN	*Metropolitan Area Network* o red de área metropolitana
Mirroring	"Espejeado". Copia exacta de un disco a otro.
Nodo	Un nodo de un *cluster* es cualquier sistema que sea miembro de éste.

Nologging Operación que no deja registro o no inserta información en los *redo logs*.

OLTP Procesamiento de transacciones en línea, generalmente en un entorno de alta intensidad en procesamiento de datos.

RAID *Redundant array of Independent Disks* o arreglo redundante de discos independientes o de bajo costo.

Reconcile Función de reconciliación entre instancias.

Redo Log files Cada base de datos tiene 2 o más *redo log files*. Su principal función es almacenar los cambios hechos a la base de datos Oracle.

Row Conjunto de atributos o valores pertenecientes a una entidad o registro en una tabla. Un *row* es una colección de información correspondiente a la columna para un solo registro.

Rowid Es un identificador global único para un *row* en la base de datos. Es creado al mismo tiempo que el *row* es insertado en la tabla y destruido cuando es removido de la tabla.

SCSI *Small Computer System Interface*.

SGA Es un grupo de estructuras de memoria compartida que contienen los datos e información de control de una instancia de base de datos Oracle.

Sistema operativo	Conjunto de programas de *software* que controlan el funcionamiento general del computador y que administran los recursos del mismo.
Software	Todos los programas que hacen funcionar a una computadora.
TCP/IP	Protocolo de comunicación utilizado para transmitir datos en una red de computadoras.
WAN	*Wide Area Network* o red de área amplia.

RESUMEN

Los sistemas de información de los negocios y empresas globales poseen clientes en todo el mundo que requieren acceso a los datos las 24 horas del día, 7 días de la semana. Lo que quiere decir que no hay ninguna interrupción para los sistemas, eso significa alta disponibilidad; sin embargo, los componentes de un computador pueden fallar y con ello se pierde el acceso a la base de datos, lo que se traduce en pérdida para las empresas. Es por eso que las aplicaciones web, sistemas en línea que generan gran procesamiento de transacciones, necesitan altos niveles de disponibilidad para los sistemas manejadores de bases de datos, los cuales no pueden bajar de 99% del tiempo en servicio. Experimentar un porcentaje de tiempo fuera de servicio del 1% al año, significa que tendrá un tiempo de 3.65 días para reparar al componente fallido o darle mantenimiento. Claro está que si se quiere aumentar ese porcentaje a casi el 100%, equivale a encontrar una solución de alta disponibilidad de base de datos ya sea por medio de *hardware* o por medio de *software*, que aun que signifique una gran inversión, si el negocio es de misión crítica, vale la pena.

En el mercado se encuentran varios manejadores de base de datos, en el presente trabajo se encontrará los manejadores más comunes como Oracle 9i y SQL *Server* 2000; los cuales tienen soluciones de alta disponibilidad de base de datos y éstas combinadas con soluciones de alta disponibilidad de base de datos por *hardware*, hacen posible disminuir ese tiempo fuera de servicio.

Los *clusters* son grupos de sistemas independientes que funcionan como un único sistema, estos proporcionan un mejor rendimiento al distribuir la carga entre los nodos que lo conforman. Combinados con un almacenamiento compartido que proporcione redundancia de los datos, utilizando un arreglo redundante de discos independientes mas comúnmente conocido como *RAID* se solucionaría el problema del tiempo fuera de servicio.

La replicación es una solución de alta disponibilidad de base de datos por *software* que implica tener una o más copias de la base de datos en producción en más de un lugar y que la misma se pueda utilizar en caso de una destrucción total del sitio de producción sin que los usuarios noten qué es lo que ha sucedido.

Por lo tanto, es necesario comparar las características más relevantes resumidas en un cuadro que contengan las soluciones de alta disponibilidad de base de datos por *hardware* o por *software* como si las mismas proporcionan recuperación a desastres, disponibilidad, confiabilidad, escalabilidad, costo, etc., para tomar una decisión que ayudara a tener un considerable tiempo en servicio de sus sistemas de información.

OBJETIVOS

General

Comparar las ventajas y desventajas de la alta disponibilidad de base de datos por *hardware* o por *software*.

Específicos

1. Describir el porqué de la alta disponibilidad de las base de datos 24 x 7.

2. Definir qué es un *cluster*.

3. Describir el porqué de la replicación.

4. Describir las ventajas y desventajas de alta disponibilidad de base de datos por medio de *hardware*.

5. Describir las ventajas y desventajas de la alta disponibilidad de base de datos por medio de *software*.

INTRODUCCIÓN

En el presente trabajo se expone la alta disponibilidad de las bases de datos puesto que ya no se trata de una simple preferencia de las organizaciones, ahora es un factor crítico para su éxito, sobre todo, cuando la información vital de las empresas se comparte electrónicamente entre los empleados, socios y proveedores. La más mínima falla de la red puede resultar en la pérdida de ganancias, pérdida de productividad o algo peor.

Un *cluster* es una colección de computadores independientes que ejecutan una serie de aplicaciones de forma conjunta y aparecen ante clientes como un solo sistema. Los *clusters* son una solución de *hardware* al que se le puede agregar un almacenamiento redundante y éste ser una buena solución de alta disponibilidad de base de datos ya que el *cluster* puede ocultar ante los usuarios del sistema los fallos de componentes y proporcionar servicios de alta disponibilidad.

La replicación es una solución basada en *software* y es el proceso de mantener copias de los datos del sistema principal que pueden ser usados como datos alternativos sobre otros sistemas. Esas copias son usadas si el sistema principal necesita estar fuera de línea para efectuar copias de seguridad o rutinas de mantenimiento (alta disponibilidad), en caso de emergencia, cuando el sistema principal haya fallado (recuperación de fallo) o en una situación de destrucción total como lo es un terremoto (recuperación de desastre). Una simple copia de seguridad restaurada en un sistema alterno, puede ser considerada como una forma de replicación.

Cuando se habla de alta disponibilidad en *hardware*, se hace referencia a tener dos máquinas con igual *hardware* y sistema operativo con un canal de alta velocidad y un ancho de banda grande (fibra óptica), la(s) máquina(s) comparte(n) el acceso a una base de datos en un arreglo de discos.

Cuando se habla de alta disponibilidad por software, se hace referencia a tener una máquina 1 con un tipo de *hardware*, sistema operativo y la base de datos del sistema situada en un punto geográficamente distante de otro local ya sea en unos cientos de kilómetros o bien en una parte del mundo, ésta utiliza un canal de comunicación con un ancho de banda normal o incluso por medio de Internet, con el cual se estará refrescando los cambios a la base de datos del lugar origen al remoto en donde se encuentra la máquina 2 con igual tipo de *hardware*, sistema operativo y una réplica de la base de datos de la máquina 1.

Es importante mencionar que para que las soluciones de alta disponibilidad de base de datos por *hardware* o *software* más comunes le ayuden a tener ideas, características y parámetros para la evaluación de las alternativas que mejor satisfagan las necesidades de la empresa que las consulta, es necesario que posea información sobre las soluciones más comunes sobre esta tecnología.

1. ALTA DISPONIBILIDAD DE BASE DE DATOS 24 x 7

1.1. Alta disponibilidad de base de datos 24 x 7

Los sistemas de información de los negocios y empresas globales poseen clientes en todo el mundo que requieren acceso a los datos todos los días las 24 horas. Varias aplicaciones necesitan altos niveles de disponibilidad para los sistemas manejadores de bases de datos, incluyendo los sistemas de tiempo real, aplicaciones web y otros tipos de sistemas en línea.

La criticidad se determina en función de varios factores. Uno de ellos es la naturaleza del servicio; si para la empresa es fundamental que esté disponible (independientemente del número de usuarios) este servicio es crítico. Si el servicio ha de estar activo en un periodo de tiempo para un número elevado de usuarios, también es crítico. En ambos casos la no disponibilidad puede suponer elevadas pérdidas para la empresa. Un caso típico es un proveedor de servicios Internet.

La alta disponibilidad significa acceder a los datos y aplicaciones donde quiera que lo necesite y con un nivel de desempeño aceptable. La alta disponibilidad congenia con los aspectos de servicio del sistema como un no rompimiento total y como es percibido por los usuarios finales. En este contexto, confiabilidad de componentes de *hardware* y *software* y rendimiento, tiempo de respuesta, etc. son partes de un sistema disponible. La alta disponibilidad es la proporción de tiempo en el que un sistema es productivo y es usualmente expresado como un porcentaje.

Los sistemas de alta disponibilidad bajan entre 100% y 99.9% de disponibilidad. En la tabla que se muestra a continuación la primer columna significa el porcentaje de tiempo en que un sistema corre sin ningún tipo de falla, en la siguiente columna se muestra el resto del porcentaje en que un sistema debe estar fuera de servicio y en la siguiente columna su equivalente en tiempo al año.

Tabla I. Tiempo sin servicio anual

Porcentaje de tiempo en servicio	Porcentaje de tiempo fuera de servicio	Anual	Normalizado (segundos)
98%	2 %	7.30 días	630,720
99%	1 %	3.65 días	315,360
99.8%	0.2%	17 horas, 30 minutos	63,000
99.9%	0.1%	8 horas, 45 minutos	31,500
99.99%	0.01%	52 minutos, 30 segundos	3,150
99.999%	0.001%	5 minutos, 15 segundos	315
99.9999%	0.0001%	31.5 segundos	31.5

Fuente: www.enlace.cl/empresa/anexos/alta_disponibilidad.pdf

La disponibilidad es expresada de la siguiente manera:

$$\frac{\mathbf{MTTF}}{\mathbf{(MTTF + MTTR)}}$$

donde:

MTTF (*Mean-Time-To-Failure*) tiempo promedio que un sistema corre (sin fallas) después que ha sido inicializado o reparado.

MTTR (*Mean-Time-To-Repair*) es el tiempo promedio necesario para reparar o restaurar una falla en el sistema.

En el entorno de negocios, hoy en día, pocas empresas pueden quedarse sin acceso a las misiones críticas por más de 8 horas, no pueden tolerar más que una falla por año, alrededor de 8760 horas. Además, pocos usuarios finales deberían considerar un sistema "disponible" si el desempeño está por debajo de algún nivel o si el sistema es únicamente disponible para algún porcentaje de la comunidad de usuarios. Considerando esos hechos por un mínimo, "nivel de entrada" sistema de alta disponibilidad, el MTTF es 8760 horas y el MTTR es 8 horas, lo cual da una disponibilidad de 99.9%.

1.2. Niveles de disponibilidad

- **Convencional**: las funciones de negocios pueden verse interrumpidas y la integridad de los datos no es esencial.
  Disponibilidad: 90%
  Mecanismos: servidor de base de datos regular con respaldo tradicional (técnicas de recuperación y copia de seguridad).

- **Media** (*High Reliable*): las funciones de negocios pueden verse interrumpidas pero se debe mantener la integridad de datos.
  Disponibilidad de servicio: 95%
  Mecanismos: bitácoras de operaciones.

- **Alta Disponibilidad**: las funciones de negocios aceptan pequeñas interrupciones y al retomar se reprocesan transacciones.
  Disponibilidad: 99%
  Mecanismos: bitácoras de operaciones, recuperación automática.

- **Resistencia a fallas** (*Fault Resilient*): requiere de operación ininterrumpida en horario laboral, se retoma en caso de falla automáticamente.
  Disponibilidad: 99.9%
  Mecanismos: *clustering*, *mirroring*.

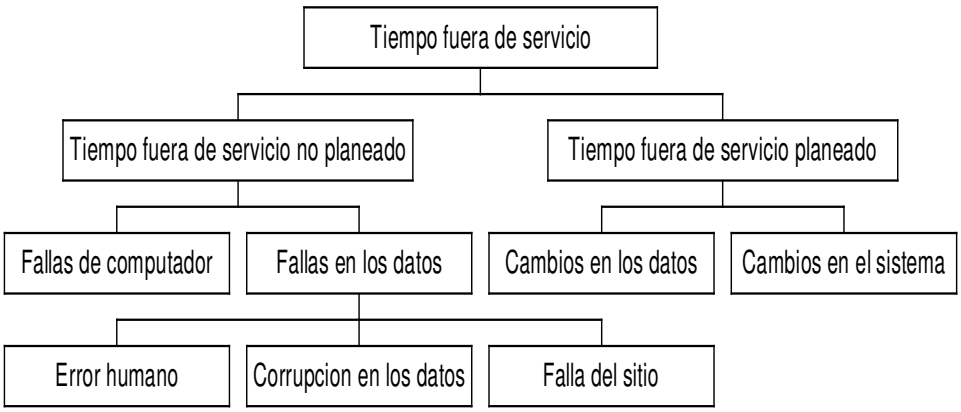
- **Tolerancia a fallas** (*Fault Tolerance*): capacidad de procesamiento continuo y cualquier falla debe ser transparente para el usuario.
  Disponibilidad: 99.99%
  Mecanismos: duplicidad del sitio y redundancia.

- **Tolerancia a Desastres**: disponibilidad en todo momento, capacidad para soportar desastres naturales y humanos.
  Disponibilidad: 99.999%
  Mecanismos: Los anteriores más dos sitios y mecanismos de recuperación.

1.3. Tiempo fuera de Servicio

La meta de la alta disponibilidad es cuidar cualquier rompimiento en el servicio. Los sistemas de alta disponibilidad tienen representaciones, capacidades y empleo de estrategias especialmente diseñadas para minimizar el tiempo fuera de servicio. El Tiempo fuera de servicio puede ser planeado o no planeado.

Figura 1. Causas del tiempo fuera de servicio



1.3.1. Tiempo fuera de servicio planeado

El tiempo fuera de servicio **planeado** es el cuando el sistema no está disponible debido al programa de mantenimiento con actualizaciones de *software* / *hardware* y cuando se hace copias de seguridad del sistema. El método usado para minimizar el tiempo fuera de servicio planeado debe ser:

- Proveer copias de seguridad (*backups*), mantenimiento, actualizaciones mientras el sistema está arriba y corriendo.

- Reducir el tiempo para desempeñar esas tareas que pueden ser únicamente cuando el sistema está bajo.

1.3.2. Tiempo fuera de servicio no planeado

El tiempo fuera de servicio **no planeado** es el tiempo que el sistema no está disponible debido a fallas de componentes defectuosos del computador o defectos del medio ambiente. Errores humanos y desastres naturales son ejemplos de defectos del medio ambiente. El método usado para minimizar el tiempo fuera de servicio no planeado es para:

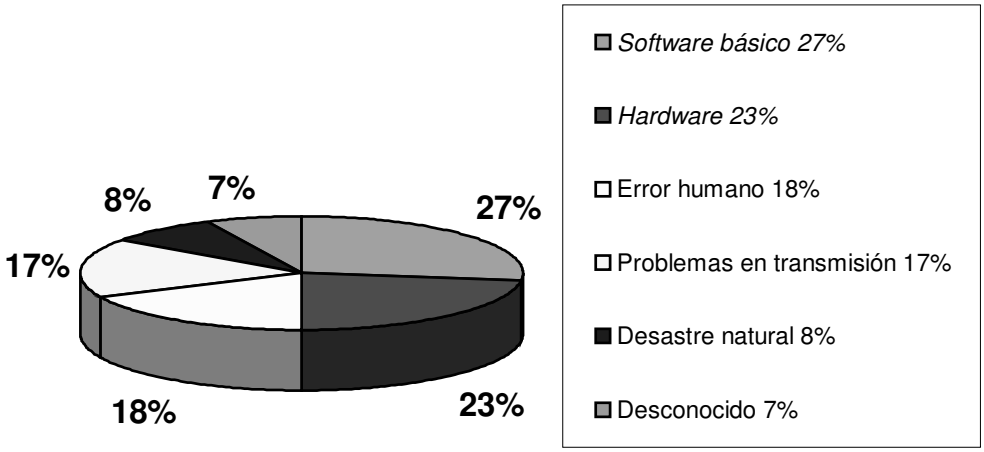
- Minimizar el número de defectos y el efecto / recuperación en el tiempo de los defectos en un sistema.

- Evitar un punto singular de falla por la utilización redundante de partes (*failover*).

- Reducir el impacto de los defectos del medio ambiente usando UPS y copia idéntica de los datos fuera del sitio y/o replicación en caliente de componentes omitidos.

Para mencionar el porcentaje de las causas no planeadas que provocan un paro se debe revisar la gráfica que se presenta a continuación.

Figura 2. Gráfica de las causas desconocidas no planeadas



Fuente: Gartner/Dataquest www.enlace.cl/alta_disponibilidad.pdf

La solución lógica para incrementar la disponibilidad es mantener los datos en más de un lugar (recuperación de desastre), evitar que el usuario perciba las fallas en el servicio (recuperación de fallas) y mejorar el tiempo de respuesta (rendimiento).

El criterio para una la alta disponibilidad y alto desempeño incluye una solución:

- Mínimo impacto sobre la disponibilidad y desempeño de un sistema primario.

- Una copia completa de la base de datos primaria para reportar y extraer, que siempre es accesible donde no hay emergencia.

- La copia de la base de datos deberá ser una imagen de la actualización de la base de datos primaria.

- Capacidad para llegar a convertirse en la base de datos primaria (*failover*) en caso de desastre.

- El *failover* de la base de datos secundaria deberá ser sin pérdida de datos.

- Después del desastre, la solución deberá habilitar un cambio de vuelta al sistema primario.

- La copia no requiere administración de la propia base de datos en adición a la administración del sistema primario.

- Redundancia en CPU, componentes de red, fuentes de poder, almacenamiento de datos, etc.

- Ubicación remota del sistema secundario.

2. ALTA DISPONIBILIDAD DE BASE DE DATOS POR *HARDWARE*

2.1. ¿Qué es un Cluster?

Un *cluster* es un grupo de sistemas independientes que ejecutan una serie de aplicaciones de forma conjunta y aparecen ante clientes y aplicaciones como un solo sistema. Las configuraciones del *cluster* son usadas para disponibilidad y escalabilidad.

2.1.1. Nodo de *cluster*

Un nodo de *cluster* es cualquier sistema que sea miembro de un cluster. Los tres tipos de nodos de *cluster* que pueden encontrarse en un domino de recuperación son primarios, de reserva y de duplicación.

2.1.1.1. Nodo primario

Es el nodo de *cluster* que sirve de punto de acceso y de copia principal de un recurso. Si se produce una anomalía en este nodo, todos los objetos del grupo de recursos de *cluster* que tengan a este nodo como punto de acceso primario conmutarán por anomalía a un nodo de reserva.

2.1.1.2. Nodo de reserva

Es el nodo de *cluster* que asumirá el cometido de servir de acceso primario si el nodo primario actual sufre una anomalía. Este nodo de clúster contiene una copia de un recurso de *cluster*. En el caso de un grupo de recursos de *cluster* de tipo datos, las copias de los datos se mantienen actualizadas mediante duplicación.

2.1.1.3. Nodo de duplicación

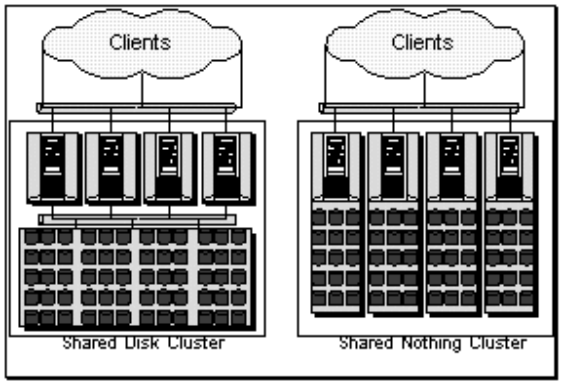
Es un nodo de *cluster* que contiene copias de los recursos de éste, pero que no puede asumir el cometido de nodo primario o de nodo de reserva.

2.2. Modelos básicos de *cluster*

2.2.1. *Cluster shared disk*

En un *cluster shared disk*, todos los procesadores tienen acceso directo a todos los discos y datos, pero ellos no comparten la memoria principal. Se necesita un *software* extra, llamado cache distribuido, el cual es requerido para manejar la concurrencia global de cache entre los procesadores.

Figura 3. *Cluster shared disk* y *shared nothing*



Fuente: http://www.microsoft.com/technet/prodtechnol/sql/2000/plan/default.mspx

2.2.2. *Cluster shared nothing*

Cada nodo en un *cluster shared nothing* es una computadora totalmente independiente con sus propios recursos y su propio sistema operativo. Cada nodo tiene su propia memoria y almacenamiento en disco; la comunicación entre los nodos es a través de mensajes entre una interconexión compartida.

2.3. Servidor *cluster*

Un servidor *cluster* (*Cluster server)* es un grupo de servidores independientes manejados como un solo recurso unificado, esto permite compartir cargas entre los computadores para obtener un excelente rendimiento y un muy seguro sistema de alta disponibilidad, pues cuando algún servidor salga de funcionamiento los otros toman el control y mantienen arriba todos los servicios para los usuarios.

2.4. Cluster por replicación de datos

Un cluster que utiliza la replicación de datos es la mejor solución para alcanzar los requisitos de disponibilidad continua, proporcionando una recuperación rápida para la más amplia gama de paradas posibles y máxima flexibilidad de las aplicaciones. La inversión depende del tamaño de los servidores (número de CPU, velocidad de CPU, cantidad de memoria, espacio en disco, tipo de sistema operativo), va desde un nivel intermedio hasta un nivel alto de inversión.

2.5. Conmutación por anomalía

Conmutación por anomalía significa que el sistema conmuta automáticamente a uno o más de los sistemas de reserva en caso se produzca una anomalía en el sistema.

2.6. Conmutación por administración

Una conmutación por administración se produce si se conmuta manualmente el acceso de un sistema a otro sistema. En general, es posible que se realice esta acción si se lleva a cabo el mantenimiento del sistema, por ejemplo, en el caso de aplicar arreglos temporales de programa, instalar una nueva versión o versión de mantenimiento (*release*) o actualizar el sistema.

2.7. Gestión del proceso de grupos de recursos de *cluster* de datos y de aplicación

Si se produce una conmutación por anomalía o una conmutación por administración en un *cluster*, los grupos de recursos de *cluster* gestionan todos los cambios en el dominio de recuperación. Una conmutación por anomalía o una conmutación por administración puede afectar a grupos de recursos de *cluster* de tipo datos y de tipo aplicación. Se procesan todos los grupos de recursos de *cluster* de datos resistentes (tipo 1) antes de que se procese algún grupo de recursos de *cluster* de aplicación (tipo 2). Quizá se desea utilizar un bloqueo para retener la aplicación hasta que los datos estén disponibles para su proceso.

2.8. Ventajas de un cluster

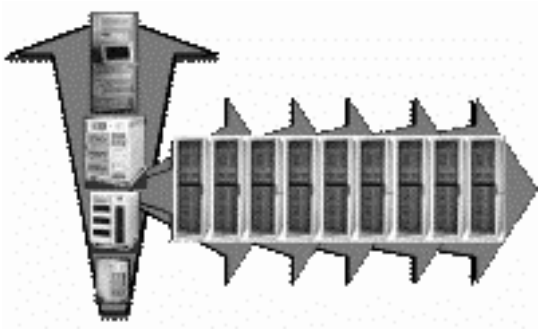
2.8.1. Escalabilidad

La escalabilidad es la capacidad de un equipo para hacer frente a volúmenes de trabajo cada vez mayores sin, por ello, dejar de prestar un nivel de rendimiento aceptable.

Existen dos tipos de escalabilidad de *hardware* los cuales se muestran en la siguiente figura:

- Escalabilidad vertical
- Escalabilidad horizontal

Figura 4. Escalabilidad vertical y escalabilidad horizontal



Fuente: http://www.microsoft.com/technet/prodtechnol/sql/2000/plan/default.mspx

2.8.1.1. Escalabilidad vertical

La escalabilidad vertical se basa en la utilización de un gran equipo cuya capacidad se aumenta a medida que lo exige la carga de trabajo existente, ésta se alcanza en cuanto a la adición de *hardware* a un solo nodo, procesadores, memoria RAM .

2.8.1.2. Escalabilidad horizontal

La escalabilidad horizontal se basa en cambio, en la utilización de un *cluster* compuesto de varios equipos de mediana potencia que funcionan en tandem de forma muy parecida a como lo hacen las unidades de un *RAID* (*Redundant array of independent disks* o Arreglo redundante de discos independientes). Del mismo modo que se añaden discos a un *RAID* para aumentar su capacidad, se pueden añadir nodos a un *cluster* para aumentar su rendimiento.

2.9. *RAID* (local)

2.9.1. Introducción a los sistemas *RAID*

RAID combina múltiples discos duros en un arreglo, junto con una controladora, que gestiona la repartición de datos entre el mencionado arreglo de discos y almacena la información procurando evitar que se pierdan datos si uno o más discos llegan a fallar. Esto proporciona una mayor protección de los datos, velocidades más altas de transferencia y una mayor capacidad que la que proporcionaría un único disco duro. El servidor ve al sistema *RAID* como si se tratase de cualquier otro disco duro externo. Existen distintos niveles de redundancia en los arreglos RAID (generalmente se reconocen desde RAID-0 hasta RAID-5 aunque existen proveedores que han especificado unilateralmente otros niveles, los que definen distintas especificaciones de almacenamiento.

La mayoría de los sistemas operativos de red modernos (Windows Server, Unix, etc.), tienen capacidad de manejar algunos o todos los niveles antes mencionados de *RAID* (sistemas *RAID* basados en *software*), pero cuando se buscan altos niveles de seguridad en la redundancia de la información almacenada, se recurre a sistemas *RAID* basados en *hardware*. Además de ser más seguros, las soluciones *RAID* basadas en *hardware* son también más rápidas que las soluciones basadas en *software*.

2.9.2. Funcionamiento del sistema *RAID*

El funcionamiento del sistema RAID se sustenta en dos elementos: *DATA STRIPPING* y PARIDAD.

2.9.2.1. *Data stripping*

En el *data stripping* (creación de bandas), los datos que llegan al *RAID* procedentes del servidor, son divididos por la controladoras RAID en segmentos cuyo tamaño depende del bloque que se defina. Posteriormente, esos segmentos son enviados a los diferentes discos que componen el sistema *RAID*.

2.9.2.2. Paridad

Por otro lado, en el concepto de paridad, la controladora *RAID* genera bits de paridad y los almacena en los discos del *RAID*, obteniendo así la redundancia de datos. De este modo, si un disco falla los datos pueden ser regenerados.

Los discos optimizados para RAID poseen circuitos integrados que detecta si el disco está fallando, de ser así este circuito se encargará por encima del tiempo real de sacar la información y almacenarla en los otros discos.

2.9.2.3. *Hot spare*

Un *hot spare* es un disco que permanece siempre en el sistema esperando a que otro se estropee y él entre directamente en funcionamiento.

2.9.2.4. *Hot swap*

Una de las ventajas del sistema RAID es la posibilidad de conectar y desconectar los discos en “caliente”, es decir, que si un disco falla no hará falta apagar el sistema para reemplazarlo.

2.9.3. Ventajas

El rendimiento general del sistema aumenta ya que pueden funcionar de forma paralela con los diferentes discos del conjunto. Dependiendo del nivel de *RAID* que se elija, si uno de los discos del conjunto falla, la unidad continua funcionando, sin pérdida del tiempo ni de datos. La reconstrucción de los datos del disco que ha fallado se hace de forma automática sin intervención humana. En el caso de algunos sistemas operativos la regeneración de datos se hace desde el *software*, aunque en estos sistemas se pueden usar controladoras *RAID* que sí regenerarían los datos automáticamente. La capacidad global del disco aumentará, ya que se suman las capacidades de los diferentes discos que componen el conjunto.

Para conseguir mejoras en las prestaciones del RAID, se utiliza memoria cache de lectura, que contiene datos de forma temporal, minimizando así el número de accesos necesarios. En la memoria caché de escritura se almacena un determinado número de bloques de datos, adyacentes a ser escritos, disminuyendo de ese modo los accesos a disco. Un sistema de discos RAID es plenamente multiusuario, ya que todas las solicitudes de los usuarios pueden ser atendidas simultánemante.

2.9.4. Tipos de implementaciones

Existen dos tipos de implementaciones, según la controladora, para los sistemas *RAID*.

2.9.4.1. *Hardware RAID*

Las controladoras de *hardware RAID* ofrecen mayores prestaciones y son independientes del sistema operativo, lo cual evita que se genere cualquier tipo de conflicto.

2.9.4.2. *Software RAID*

Por el contrario, si se emplea un *software RAID* para que realice las funciones propias del *RAID*, se consumirán recursos del servidor. Esto disminuirá su rendimiento. Además, al trabajar con un *software* específico se depende del sistema operativo, lo cual puede generar conflictos que deriven incluso en la corrupción de los datos.

2.9.5. ¿Controladora *RAID* interna o externa?

Sin duda es mejor utilizar un sistema *RAID* externo al servidor. Si el *RAID* es interno y se produce un fallo, en el propio *RAID* o en otro componente, se pierden los datos. En cambio, si el *RAID* es externo, aunque falle el servidor los datos seguirán intactos. Luego se conecta el *RAID* a otro servidor y el trabajo puede continuar sin perder tiempo. Además, solo el *RAID* externo permite configuraciones *cluster*, y al no ser un sistema propietario, se puede conectar a cualquier servidor, con independencia de su marca, sin que dé ningún problema ni limite a las posibilidades de configuración.

2.9.6. Tipo de cache

2.9.6.1. *Write back*

Envían al servidor un mensaje de terminación tan pronto como han escrito los datos en la caché. La controladora escribirá los datos en disco a su conveniencia. De esta forma se mejoran, en definitiva, las prestaciones de escritura.

2.9.6.2. *Write through*

En este segundo caso, la controladora escribe los datos a disco y posteriormente manda el mensaje de finalización al servidor. Este proceso de caché resulta más lento que el anterior.

2.9.7. Tipo de arreglos

2.9.7.1. Arreglos paralelos

Son aquellos en que cada disco participa en todas las operaciones de entrada o salida. Este tipo de arreglo ofrece tasas altísimas de transferencia debido a que las operaciones son distribuidas a través de todos los discos del arreglo y ocurren en forma prácticamente simultánea. La tasa de transferencia será muy cercana, 95%, a la suma de las tasa de los discos miembros, mientras que los índices de operaciones de entrada o salida serán similares a las alcanzadas por un disco individual. Un arreglo paralelo accesará sólo un archivo a la vez pero lo hará a muy alta velocidad. Algunas implementaciones requieren de actividades adicionales como la sincronización de discos. Los *RAID* de niveles 2 y 3 se implementan con arreglos paralelos.

2.9.7.2. Arreglos independientes

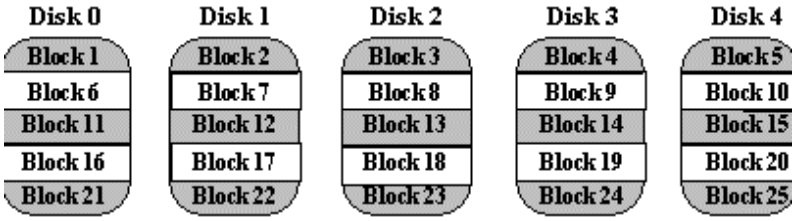
Son denominados así aquellos arreglos en los cuales cada disco integrante opera en forma independiente, aún en el caso de que le sea solicitado atender varios requerimientos en forma concurrente. Este modelo ofrece operaciones de entrada o salida sumamente rápidas debido a que cada disco está en posición de atender un requerimiento por separado. De esta forma las operaciones de entrada o salida serán atendidas a una velocidad cercana, 95%, a la suma de las capacidades de los discos presentes, mientras que la tasa de transferencia será similar a la de un disco individual debido a que cada archivo está almacenado en sólo un disco. Los niveles 4 y 5 de RAID se implementan con arreglos independientes, mientras que los niveles 0 y 1 pueden ser implementados por cualquiera de las categorías de arreglos independientes. *Stripping* y *mirroring RAID* a niveles 0, 1 y 0 & 1 puede ser implementado, tanto en forma de arreglos independientes o paralelos.

2.9.8. RAID 0

2.9.8.1. Definición

El RAID de nivel 0 distribuye la información en bloques a través de varios discos duros, esta técnica es llamada *striping*. Maneja varios discos duros como si fueran uno solo, lo que proporciona una mayor velocidad de lectura y escritura. Éste es el único nivel que no duplica la información, es decir, no proporciona redundancia, por lo tanto no desperdicia capacidad de almacenamiento.

Figura 5. *RAID* de nivel 0



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.8.2. Requerimientos mínimos

El *RAID* 0 requiere al menos 2 discos para su implementación y sus respectivas tarjetas controladoras.

2.9.8.3. Confiabilidad

Es una buena alternativa en sistemas donde sea más importante el rendimiento que la seguridad de los datos, ya que no proporciona tolerancia a fallos, si un disco falla el sistema se cae, por lo cual es bueno implementarlo en ambientes que puedan soportar una pérdida de tiempo de operación para reemplazar el disco que falle y reponer toda la información.

2.9.8.4. Rendimiento

El *RAID* de nivel 0 ofrece muy buen rendimiento permitiendo tener acceso a más de un disco a la vez, logrando una tasa de transferencia más elevada y un rápido tiempo de acceso.

2.9.8.5. Disponibilidad

Este *RAID* de nivel 0 no proporciona redundancia ya que los datos se distribuyen a través de los discos y si un disco falla el sistema se cae, por lo tanto, este nivel no proporciona alta disponibilidad.

2.9.8.6. Ventajas

- Rápida velocidad de lectura y escritura.
- Se logra un rendimiento mejorado cuando se distribuyen los datos a través de varias controladoras y cada una de ellas con un solo disco.
- Se emplea toda la capacidad del disco.
- Permite acceder a más de un disco a la vez, logrando una tasa de transferencia más elevada y un rápido tiempo de acceso.

2.9.8.7. Desventajas

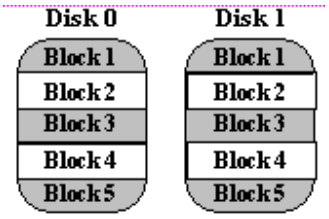
- Si un disco falla, el sistema cae.
- No existe protección de datos.
- No existe información de paridad.
- No es verdaderamente un disco *RAID* ya que no tiene integridad de datos.

2.9.9. *RAID* 1

2.9.9.1. Definición

El *RAID* de nivel 1 usa un tipo de configuración conocido como "espejeado", ya que la información de un disco es completamente duplicada en otro disco (redundancia); proporciona tolerancia a fallas ya que los discos guardan exactamente la misma información por parejas, de tal manera que si uno falla, el segundo toma su lugar.

Figura 6. *RAID* de nivel 1



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.9.2. Requerimientos mínimos

El *RAID* de nivel 1 requiere al menos dos discos para su implementación, cada disco con su tarjeta controladora.

2.9.9.3. Confiabilidad

Se protege la información en caso de falla tanto del disco como del controlador.

En caso de *duplexing* que es el agregar un segundo controlador al arreglo de discos; la ventaja es que con *duplexing* se elimina que el adaptador sea el punto de falla; la única desventaja de duplexing es el costo del segundo adaptador; ya que si un disco suspende su operación el otro continua disponible. De este modo se evita la pérdida de información y las interrupciones del sistema debido a fallas de discos.

2.9.9.4. Rendimiento

El rendimiento de la lectura se mejora ya que cualquiera de los dos discos puede leerse al mismo tiempo. El rendimiento de escritura es el mismo que el del almacenamiento en un solo disco. El *RAID* de nivel 1 proporciona el mejor rendimiento y la mejor tolerancia a fallos en un sistema multiusuario. El rendimiento no se ve afectado cuando un disco falla; el almacenamiento funciona normalmente hacia los usuarios externos.

2.9.9.5. Disponibilidad

El RAID de nivel 1 proporciona alto nivel de protección ya que el espejeado proporciona el 100% de duplicación de los datos, está diseñado para sistemas donde la disponibilidad de la información es esencial (datos de misión crítica) y su reemplazo resultaría difícil y costoso, más costoso que reponer el disco en sí.

2.9.9.6. Ventajas

- Puede perder un disco y recuperarse (tolerante a fallas).
- Es el más confiable con respecto a guardar la información (redundancia).

- Alto nivel de protección ya que el espejeado proporciona el 100% de duplicación de los datos.
- Entrega el mejor rendimiento de cualquier arreglo redundante durante una reconstrucción.
- No es necesario la reconstrucción de los datos. Si un disco falla, todo lo requerido es copiar bloque por bloque a un nuevo disco.

2.9.9.7. Desventajas

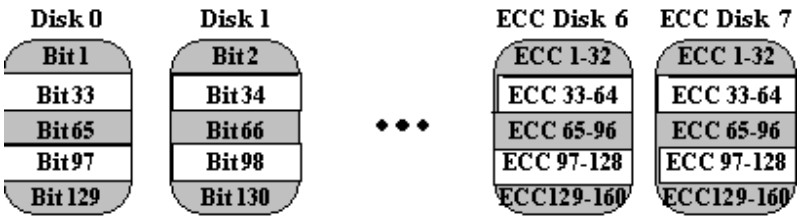
- Debido a que escribe la información dos veces, hay un menor rendimiento comparado con la escritura hacia un solo disco.
- Requiere dos discos para un 100% de redundancia, duplicando el costo.
- Es lento en la escritura de datos, porque debe escribir en dos localizaciones si se hace por medio de *software*.
- Cuando se escriben datos deteriorados en un disco, estos son duplicados con los mismos defectos en el disco espejo.
- Se desperdicia el 50% de la capacidad.

2.9.10. *RAID* 2

2.9.10.1. Definición

El *RAID* de nivel 2 adapta la técnica comúnmente usada para detectar y corregir errores en memorias de estado sólido. En un RAID de nivel 2, el código ECC (*Error correction code*) se intercala a través de varios discos a nivel de bit. El método empleado es el *Hamming*. Está diseñado para ser utilizado con discos que carecen de detección de error interna (discos antiguos). Todos los discos SCSI soportan detección de error interna, por lo que este nivel de RAID tiene muy poca utilidad práctica para esos modelos de discos.

Figura 7. RAID de nivel 2



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.10.2. Confiabilidad

Debido a que el código *Hamming* se usa tanto para detección como para corrección de errores (*Error detection* and *correction*), *RAID* 2 no hace uso completo de las amplias capacidades de detección de errores contenidas en los discos. Las propiedades del código *Hamming* también restringen las configuraciones posibles de matrices para *RAID* 2, particularmente el cálculo de paridad de los discos. Por lo tanto, *RAID* 2 no ha sido implementado en productos comerciales, lo que también es debido a que requiere características especiales en los discos y no usa discos estándares.

2.9.10.3. Rendimiento

Debido a que es esencialmente una tecnología de acceso paralelo, *RAID* 2 está más indicado para aplicaciones que requieran una alta tasa de transferencia y menos conveniente para aquellas otras que requieran una alta tasa de demanda de entrada y salida.

2.9.10.4. Disponibilidad

RAID 2 usa múltiples discos dedicados para almacenar información de paridad y por lo tanto requiere que un arreglo contenga un número relativo de discos individuales. Por ejemplo, un *RAID* 2 con 4 discos de datos requiere tres discos dedicados a paridad. Consecuentemente, *RAID* 2 tiene alta redundancia de cualquier esquema *RAID* orientado a la paridad.

2.9.10.5. Ventajas

- Utiliza códigos de corrección de *Hamming*.
- Permite altas tasas de transferencia.

2.9.10.6. Desventajas

- No puede ser usado con discos SCSI.
- *RAID* 2 no hace uso completo de las amplias capacidades de detección de errores contenidas en los discos.

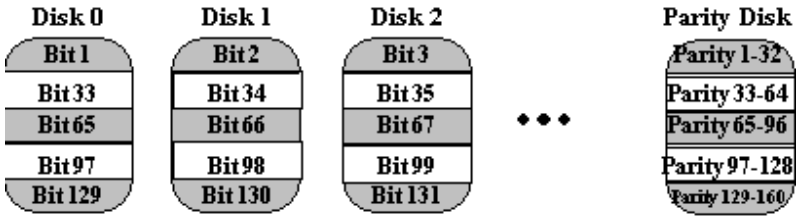
2.9.11. *RAID* 3

2.9.11.1. Definición

El *RAID* de nivel 3 introduce el chequeo de paridad o la corrección de errores. Divide el bloque de datos y los distribuye a través de múltiples discos, añadiendo redundancia mediante la utilización de un disco de paridad dedicado, que detecta errores en los datos almacenados producidos por una falla de cualquier disco y los reconstruye mediante algoritmos especiales.

Si la falla se produce en el disco de paridad se pierde la redundancia pero se mantiene intacta la información original.

Figura 8. *RAID* de nivel 3



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.11.2. Requerimientos mínimos

El *RAID* de nivel 3 requiere mínimo tres discos y se utiliza la capacidad de un disco para la información de control.

2.9.11.3. Confiabilidad

Debido a que *RAID* de nivel 3 escribe los datos en grandes bloques de información es una alternativa apropiada para aplicaciones tales como video, que envían y reciben grandes archivos. Este arreglo de discos *RAID* de nivel 3 son frecuentemente usados en aplicaciones que requieren alto ancho de banda pero no altas tasas de entrada y salida.

2.9.11.4. Rendimiento

El *RAID* de nivel 3 proporciona alto rendimiento debido a la distribución de los datos para transferencias de datos secuenciales; pero en cuanto a transferencias de datos pequeños su rendimiento es pobre. Las lecturas son rápidas. Si embargo, cada escritura requiere que el disco de paridad sea accesado y escrito conforme los datos son escritos en los demás discos.

2.9.11.5. Disponibilidad

El *RAID* de nivel 3 proporciona disponibilidad ya que gracias al disco de paridad se puede reconstruir el disco dañado pero si éste falla se pierde la redundancia.

2.9.11.6. Ventajas

- Disco de paridad para corrección de errores.
- Se puede perder un disco y recuperarse.
- Datos a nivel de *bytes*.

2.9.11.7. Desventajas

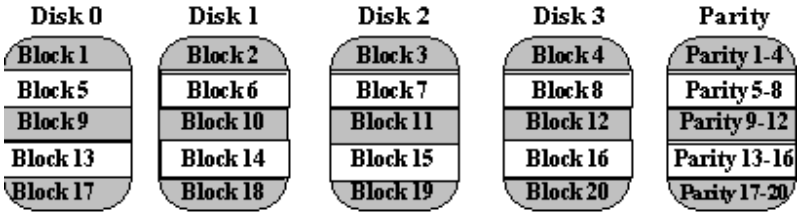
- El disco de paridad puede convertirse en un cuello de botella ya que cada cambio en el grupo *RAID* requiere un cambio en la información de paridad.
- No plantea una solución de fallo simultáneo en dos discos.
- Es especialmente recomendado para aplicaciones como video, imágenes, etc., que envían y reciben grandes archivos.
- Si se pierde el disco de paridad se pierde la redundancia.
- Una falla en un disco reduce el arreglo a *RAID* 0.

2.9.12. *RAID* 4

2.9.12.1. Definición

El *RAID* de nivel 4 distribuye los datos a nivel de bloque (la principal diferencia con el nivel 3), a través de varios discos con la paridad almacenada en un disco. La información de paridad permite la recuperación de cualquier disco en caso de falla.

Figura 9. *RAID* de nivel 4



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.12.2. Requerimientos mínimos

Se requiere un mínimo de tres unidades para implementar una solución *RAID* 4.

2.9.12.3. Confiabilidad

Debido a su organización interna, este *RAID* es especialmente indicado para el almacenamiento de ficheros de gran tamaño, esto lo hace ideal para aplicaciones gráficas donde se requiera, además, fiabilidad de los datos. La ventaja con el *RAID* de nivel 4 está en que se puede acceder a los discos de forma individual.

2.9.12.4. Rendimiento

El rendimiento de *RAID* de nivel 4 es muy bueno para lecturas (similar al nivel 0). Sin embargo, la escritura requiere que los datos de paridad sean actualizados cada vez. Esto retarda particularmente las escrituras aleatorias pequeñas, aunque las escrituras grandes o secuenciales son razonablemente rápidas. Debido a que solamente un disco del arreglo es utilizado para datos redundantes, el costo por *megabyte* de un arreglo *RAID* de nivel 4 es relativamente bajo.

2.9.12.5. Disponibilidad

Este arreglo *RAID* de nivel 4 proporciona tolerancia al fallo basándose en la utilización de un disco dedicado a guardar la información de paridad calculada a partir de los datos guardados en los otros discos. En caso de avería de cualquiera de las unidades de disco, la información se puede reconstruir en tiempo real mediante la realización de una operación lógica de O exclusivo.

2.9.12.6. Ventajas

- Rápido en lectura.
- Datos a nivel de bloques.
- Un disco de paridad.

2.9.12.7. Desventajas

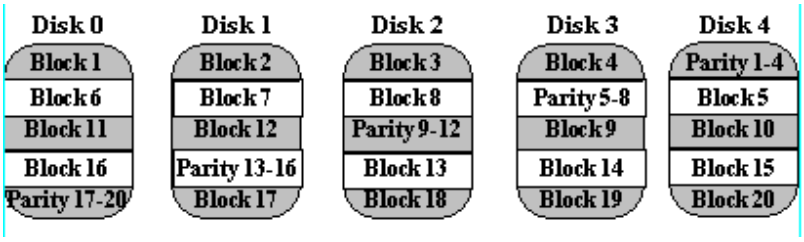
- Escritura lenta en pequeños accesos y regular en grandes escrituras.

2.9.13. *RAID* 5

2.9.13.1. Definición

El *RAID* de nivel 5 crea datos de paridad, distribuyéndolos a través de los discos (excepto en aquel disco en que se almacena la información original), obviando la necesidad de un disco de paridad dedicado. El nivel 5 es el más completo de todos los niveles de redundancia por distribución, porque si un disco falla, la información de paridad en los otros permite la reconstrucción de toda su información. Aún más, el nivel 5 escribe datos en los discos al nivel de bloques (en lugar de trabajar al nivel de *bytes*), volviéndolo mas apropiado para múltiples transacciones pequeñas como correo electrónico, procesadores de palabras, hojas electrónicas y aplicaciones de bases de datos.

Figura 10. *RAID* de nivel 5



Fuente: http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome.html

2.9.13.2. Requerimientos mínimos

Los requerimientos mínimos para implementar el *RAID* de nivel 5 son de al menos 3 discos.

2.9.13.3. Confiabilidad

RAID de nivel 5 reduce (pero no elimina) los cuellos de botella que se formaban en las soluciones *RAID* de niveles 2 al 4, ya que distribuye la paridad de los datos en todos los discos físicos del arreglo, de tal modo que permite escrituras y lecturas paralelas. El *RAID* de nivel 5, es confiable para aplicaciones intensas de entrada y salida y de lectura y escritura, tal como procesamiento de transacciones. Es mejor para los sistemas multiusuario en los cuales el rendimiento no es crítico o que realizan pocas operaciones de escritura.

2.9.13.4. Rendimiento

El *RAID* de nivel 5 asegura un mejor rendimiento de operaciones de entrada o salida para aplicaciones en la que el sistema realiza búsquedas aleatorias de muchos archivos pequeños como sucede en las aplicaciones transaccionales. Por otra parte, el rendimiento es muy bajo cuando se hacen lecturas o escrituras secuenciales de grandes volúmenes de información.

2.9.13.5. Disponibilidad

El nivel 5 proporciona disponibilidad debido a la redundancia por distribución, porque si un disco falla, la información de paridad en los otros permite la reconstrucción de toda su información.

2.9.13.6. Ventajas

- Distribuye la paridad en todos los discos.
- Apropiado para aplicaciones transaccionales.

- Es tolerante a fallos.

2.9.13.7. Desventajas

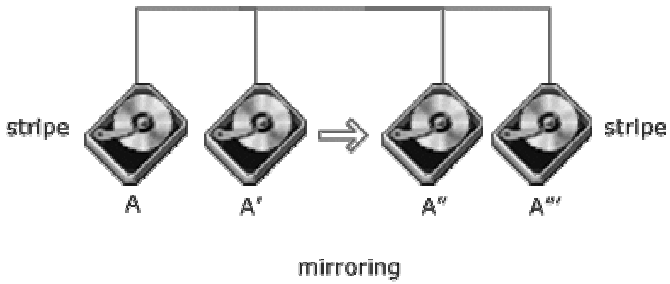
- No plantea una solución al fallo simultáneo en dos discos.
- No aumenta el rendimiento en las aplicaciones, aunque la velocidad de transferencias de datos es alta.

2.9.14. *RAID* 0 + 1

2.9.14.1. Definición

Es un nivel de arreglo de discos, donde la información se distribuye en bloques como en *RAID* de nivel 0, adicionalmente cada disco se duplica como *RAID* de nivel 1, creando un segundo nivel de arreglo. Se conoce como "*striping* de arreglos duplicados". Se requieren, dos canales, dos discos para cada canal y se utiliza el 50% de la capacidad para información de control.

Figura 11. *RAID* de nivel 0 + 1



Fuente: http://linux.cudeso.be/raid.php

2.9.14.2. Requerimientos mínimos

Los requerimientos mínimos para implementar el *RAID* 0 + 1 son de un mínimo de 4 discos para su implementación y sólo dos de ellos se utilizan para el almacenamiento de datos.

2.9.14.3. Confiabilidad

El *RAID* de nivel 0 + 1 aplica para sistemas de misión crítica donde se requiera mayor confiabilidad de la información, ya que pueden fallar dos discos inclusive (uno por cada canal) y los datos todavía se mantienen en línea. Es apropiado también en escrituras aleatorias pequeñas.

2.9.14.4. Rendimiento

El *RAID* de nivel 0 + 1 provee las mismas características de rendimiento del nivel 0 mientras proporciona la redundancia de un arreglo *mirrored*; sin embargo, porque cada miembro es duplicado el costo por *megabyte* de un *RAID* de nivel 0 + 1 es extremadamente alto.

2.9.14.5. Disponibilidad

Este nivel ofrece un 100% de redundancia de la información y un soporte para grandes volúmenes de datos.

2.9.14.6. Ventajas

- 100% de redundancia de la información.
- Soporta grandes volúmenes de datos.

2.9.14.7. Desventajas

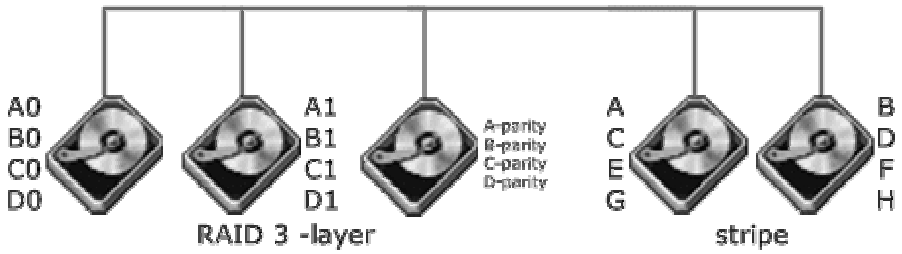
- Costo elevado.
- Menor capacidad de transferencia que otros.

2.9.15. *RAID* 53

2.9.15.1. Definición

Es el nivel de arreglo de discos más reciente, es implementado como un arreglo *striped* de nivel 0, en el cual cada segmento es un arreglo *RAID* de nivel 3. Éste tiene la misma redundancia y tolerancia a fallos que *RAID* 3. Podría ser usado para sistemas que necesitan una configuración *RAID* 3 con alta tasa de transferencia de datos, pero es caro e ineficiente.

Figura 12. *RAID* 53



Fuente: http://linux.cudeso.be/raid.php

2.9.15.2. Requerimientos mínimos

Los requerimientos mínimos para implementar el *RAID* 53 son de 5 discos.

2.9.15.3. Confiabilidad

La confiabilidad es bastante alta debido a que puede sobrevivir a una sola falla en cada arreglo *RAID* 3.

2.9.15.4. Rendimiento

Su rendimiento es bastante alto en cuanto a lecturas y escrituras se refiere.

2.9.15.5. Ventajas

- Posee la misma tolerancia a fallos que el *RAID* 3.
- Alta tasa de transferencia de datos gracias a los segmentos del arreglo de *RAID* 3.

2.9.15.6. Desventajas

- Demasiado caro para implementar.
- No está disponible en el mercado. Su costo es bajo respecto a *RAID* 0 + 1, pero sigue siendo caro respecto a *RAID* 3 ó 5.

2.10. Opciones de almacenamiento para arquitecturas de alta disponibilidad

2.10.1. *Storage Area Network* (*SAN*)

2.10.1.1. Definición

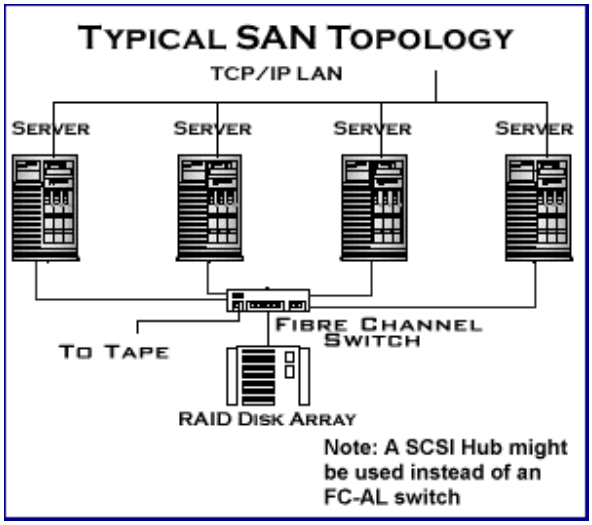
SAN es una infraestructura de red diseñada para proveer un entorno de flexibilidad, alto rendimiento y alta escalabilidad. *SAN* logra esto habilitando varias conexiones directas entre el servidor y los dispositivos de almacenamiento tales como librerías y sistemas *RAID*, de manera independiente de la *LAN* pero coexistiendo con ella. Este arreglo hace posible que el almacenamiento sea accesible a todos los servidores en la red permitiendo que la información se consolide y sea compartida entre diversos y diferentes servidores de red sin ningún impacto en la *LAN* (red de área local). Ya que la información no reside directamente en ninguno de los servidores, los recursos de estos pueden ser utilizados para otros propósitos incrementado la capacidad y el rendimiento de la red. Además la escalabilidad de toda la *SAN* puede ser mantenida dentro de cualquiera de los recursos individuales: a medida que se agregan dispositivos adicionales a la *SAN*, estos son accesibles desde cualquier servidor en la red.

Una *SAN*, además, hace que la información esté disponible para los usuarios, proporcionando un ambiente seguro para toda aquella información sensitiva, simplificando la administración y reduciendo costos generales de operación y mantenimiento.

SAN emplea interconexiones de alto rendimiento como la tecnología del canal de fibra (FC), permitiendo de una manera innovadora proteger y distribuir datos importantes. FC proporciona enlaces de hasta 120 Km de distancia permitiendo que las conexiones parezcan locales a la aplicación. La redundancia de conexión en *SAN* protege contra desastres locales.

Es confiable en las aplicaciones de bases de datos de misión crítica, en las cuales el tiempo de respuesta, la disponibilidad y escalabilidad del almacenamiento son esenciales.

Figura 13. Topología típica *SAN*



Fuente: www.infrastor.com/downloads/papaers/sanvsnas.pdf

2.10.1.2. Rendimiento

SAN mejora el rendimiento ayudando a las redes de área local congestionadas por el alto tráfico de volumen de datos que son generados por copias de seguridad, grandes migraciones de datos, aplicaciones de audio y video.

El tiempo de respuesta del almacenamiento es rápido por los enlaces del canal de fibra ya que estos pueden transferir los datos a 100 *MBps*. En el futuro, el ancho de banda del canal de fibra se duplicará a 200 *MBps*, manteniéndose delante de los avances de SCSI.

2.10.1.3. Disponibilidad

Los recursos de un servidor y del almacenamiento pueden ser agregados en línea sin interrumpir el acceso a los datos. En adición, *SAN* reduce los costos de *hardware* de la alta disponibilidad, haciéndolo aceptable para más aplicaciones. En una arquitectura *SAN*, todos los servidores pueden tener acceso directo al almacenamiento. Esto significa que un servidor puede proveer protección contra fallos para docenas de otros servidores más de lo que es posible en un *cluster* tradicional de discos compartidos. Para recuperación de desastre y *backup*, los sitios con copias idénticas pueden ser localizados a una distancia segura uno de cada otro, hasta 10 kilómetros usando cableado de fibra óptica.

2.10.1.4. Crecimiento

SAN ofrece excelente escalabilidad, agregando almacenamiento sin tener tiempo fuera de servicio (*downtime*) y ruptura asociada con las actualizaciones del servidor.

2.10.1.5. Ventajas

- Alta disponibilidad.
- Confiabilidad en la transferencia de datos.
- Reduce el trafico en la red primaria.

- Flexibilidad de configuración.
- Alto rendimiento.
- Alta escalabilidad.
- Administración centralizada.

2.10.1.6. Desventajas

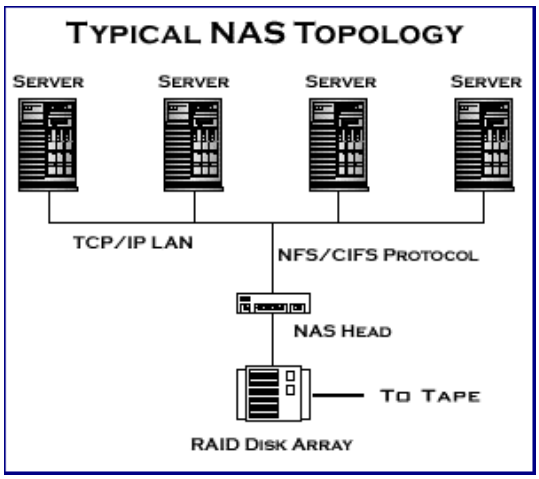
- Costo.

2.10.2. *Network Attached Storage* (*NAS*)

2.10.2.1. Definición

NAS es un sistema de almacenamiento orientado al servicio de archivos a través de una red de área local. Se basa en *file servers* con tecnología y sistemas operativos específicamente desarrollados para este propósito. Son equipos de bajo costo relativo, capaces de operar sobre redes de distinta naturaleza, multiplataformas y altamente eficientes. Los dispositivos *NAS* proporcionan servicios puros y dedicados al almacenamiento y *file sharing* además de proporcionar características avanzadas de seguridad tales como políticas de acceso, *RAID* y alimentación redundante para aumentar su confiabilidad y prestaciones.

Figura 14. Topología típica *NAS*



Fuente: www.infrastor.com/downloads/papaers/sanvsnas.pdf

2.10.2.2. Rendimiento

NAS es más lento que *SAN*, pero más rápido que el almacenamiento adjunto al servidor. *NAS* mejora el rendimiento descargando el servicio de archivo proporcionado por el *host*, liberando de esta manera los ciclos de CPU para otras tareas. Actualizar *NAS* a *Ethernet Gigabit* hace que sea incluso una solución atractiva, aunque enormes cantidades de datos pueden estar sobre la red.

2.10.2.3. Disponibilidad

NAS habilita copias de seguridad rápidas, minimizando la interrupción para acceder a los datos. Algunos aparatos *NAS* presentan funciones de replicación de datos para una rápida recuperación. La simplicidad de los aparatos los hacen más confiables que los tradicionales servidores de archivos *LAN*. Y el acercamiento del aparato elimina muchas fallas inducidas por sistemas operativos y *hardware* complejo. En adición los aparatos *NAS* pueden ser configurados para *failover*.

Carecen del rango de supervivencia y prestaciones que a menudo se requiere en una red corporativa. La capacidad *RAID* incorporada proporciona un cierto grado de protección.

2.10.2.4. Crecimiento

Agregar más capacidad de almacenamiento a una red simplemente se traduce en agregar y conectar más unidades *NAS* a ella. *NAS* es una eficiente forma para las redes corporativas de agregar capacidad de almacenamiento y liberar efectivamente a los servidores de las tareas que implican el compartir archivos.

Con *NAS*, la capacidad de almacenamiento no está amarrada a la capacidad del servidor. Las empresas agregan almacenamiento según sea necesario para la misma. Los productos *NAS* pueden escalar hasta múltiples *terabytes* y al descargar el servicio de archivos hacia esos dispositivos, entonces los servidores pueden soportar más usuarios.

2.10.2.5. Ventajas

- Pocas limitaciones en cuanto a distancia.
- Sistema específico orientado al servicio de archivos por lo tanto más eficiente, más confiable y disminuye los puntos de falla.
- Acceso a archivos sobre ambiente de red, no depende de un servidor.
- Acceso a archivos desde clientes multiplataformas.
- Fácil administración e instalación.
- No requiere licencias (multiusuario).
- Optimización del almacenamiento y el compartir archivos heterogéneos.

2.10.2.6. Desventajas

- Genera mucho tráfico de red en entornos que tienen mucha demanda de archivos.

3. ALTA DISPONIBILIDAD DE BASE DE DATOS POR *SOFTWARE*

3.1. Replicación

3.1.1. Definición

La replicación es el proceso de mantener copias de datos en producción que pueden ser usadas como sitios alternativos de datos en producción sobre otros sistemas. Esas copias son usadas si el sistema de producción necesita estar fuera de línea para *backups* o rutinas de mantenimiento (alta disponibilidad), o en casos de una emergencia cuando el sistema de producción haya fallado (recuperación de desastres). La replicación hace posible disminuir el tráfico de red y permite la tolerancia a fallos.

Esos sitios de datos alternativos pueden ser usados cada día en modo únicamente de lectura sin comprometer la viabilidad de los datos como una representación exacta de la producción de estos. Consecuentemente, ellos pueden servir para un propósito de consultar información y procesar reportes sin cargar el sistema principal de producción, lo cual ayuda en gran parte a mejorar el desempeño del proceso de transacciones en línea (*OLTP*) sobre el sistema de producción. Este concepto se ha venido desarrollando por los proveedores de sistemas administradores de bases de datos relacionales y por otras organizaciones.

Un simple *backup* restaurado sobre un sistema alternativo puede ser considerado una forma de replicación; sin embargo, este tipo de *backup* tiene las mismas limitaciones como las asociadas con las instantáneas (*snapshots*) o exportar / importar, ya que ellos son copias estáticas de los datos de producción en un punto específico en el tiempo, dándole antigüedad conforme envejece. Otras tecnologías de replicación pueden actualizar una copia de los datos replicados, pero únicamente sin ser accesada.

3.1.2. Ventajas

La replicación introduce una serie de ventajas que son:

- Distribuir la carga de trabajo en múltiples servidores.
- Mejorar la disponibilidad de datos.
- Mover subconjuntos de datos de la base de datos central a otras bases de datos.
- Optimizar las transacciones al separar los sistemas transaccionales de los sistemas de decisión.
- Reducir el tráfico de red y dar soporte optimizado para cambios de los modelos organizacionales.

3.1.3. Tipos de Replicación

Hay diversas formas de clasificar los tipos de replicación, aquí se mencionan dos de ellas, una basada en el sentido de viaje de los datos y otra por la oportunidad de la replicación.

Los tipos de replicación según el sentido del viaje de los datos son:

3.1.3.1. Unidireccional

Se tiene un nodo actualizable y otro que contiene una copia del principal pero es solo de lectura, tal como se muestra en la figura.

Figura 15. Replicación unidireccional

Fuente: http://www.pucp.edu.pe

3.1.3.2. Bidireccional

Se tienen nodos actualizando los datos, de tal forma que la replicación se produce en ambas direcciones, tal como se muestra en la figura.

Figura 16. Replicación bidireccional

Fuente: http://www.pucp.edu.pe

Los tipos de replicación según la oportunidad de replicación son:

3.1.3.3. Síncrona

Inmediatamente después que una aplicación actualiza una tabla local o réplica, la misma transacción actualiza las otras tablas en los otros nodos.

3.1.3.4. Asíncrona

Cuando se produce una actualización se guarda la información en una cola y luego se envía la información modificada a otro nodo del sistema de replicación después de cierto tiempo.

Los fabricantes de motores de bases de datos que implementan replicación introducen conceptos de acuerdo a sus beneficios, pero en general las clasificaciones que ellos proponen están dentro de la clasificación antes mencionada.

3.1.4. El porqué de la replicación

Con la replicación, se está maximizando el retorno sobre la inversión haciendo trabajar el sistema secundario, pues es continuamente actualizado, es una copia actualizada de la base de datos de producción que permite ser usada para reportes, consultas, extracciones y *backups*. Removiendo estos procesos, que normalmente compiten por recursos sobre el sistema origen o en producción, mejorando el desempeño de la instancia de producción mientras permite optimizarse la instancia secundaria para propósito de consultas.

3.1.5. Replicación basada en *logs*

La principal función de los *redo log files* en una base de datos Oracle es almacenar los cambios hechos a los datos. Oracle se protege contra fallas en los mismos *redo log* manteniendo dos o más copias de los *redo log* en discos separados. Por ello es que la información en un *redo log file* es usada para recuperar la base de datos en el caso de una falla.

La replicación basada en *logs* captura toda modificación a los objetos seleccionados inmediatamente en la base de datos origen y tan pronto como ellos son escritos al *redo log file* de Oracle, estos son replicados hacia la base de datos destino para aplicarlos.

3.1.6. Replicación usando disparadores

Los disparadores de bases de datos son procedimientos que implícitamente se ejecutan o disparan cuando se realiza una inserción, una actualización o una eliminación en una tabla a la cual está asociado. Es decir, el procedimiento se ejecuta automáticamente cuando se realiza una operación en la tabla.

Un disparador está compuesto de tres partes básicas:

- El evento que dispara el procedimiento: el cual es una instrucción de la base de datos que hace que se ejecute el código del disparador. El evento puede ser insertar datos a la tabla, actualizarlos o eliminarlos.
- La restricción del disparador: es una condición lógica que debe ser verdadera para que el disparador se ejecute. En la restricción se puede hacer referencia a cualquier valor de fila que se está modificando en la acción del disparador.

- El cuerpo del disparador: consiste en el código que se debe ejecutar cuando el evento se realiza. Este código incluye cualquier instrucción del lenguaje de manipulación de datos.

Cuando el evento ocurre el primer paso es evaluar la restricción del disparador, si el resultado es verdadero entonces se ejecuta el código como cualquier procedimiento normal.

Los disparadores pueden ser utilizados para implementar replicación síncrona de dos o más tablas en diferentes localidades. La implementación de replicación utilizando disparadores debe tener las siguientes características:

- Cada copia puede ser consultada y actualizada y todas las actualizaciones son propagadas inmediatamente a todas las localidades donde se tengan copias. Si esto es un requerimiento necesario se debe realizar la replicación utilizando disparadores.
- Actualizaciones remotas realizadas por los disparadores son parte de la transacción que ejecutó el disparador por lo que la instrucción "*commit*" (es una instrucción de sistemas de bases de datos transaccionales que significa cometido o realizado) es realizada como parte de la transacción local utilizando el proceso normal de "*commit* de dos fases".
- Si los disparadores están deshabilitados, la operaciones de replicación no son realizadas lo que provoca que una o más tablas no estén sincronizadas.
- Si existen fallas en la red las localidades que contienen réplicas no serán actualizadas.

3.1.7. Replicación instantánea (*Snapshot*)

Una instantánea es una copia completa o subconjunto de los datos de una tabla que refleja el estado actual de esa tabla. Una instantánea es definida en función de una consulta distribuida que recupera los datos y actualiza la instantánea.

A la tabla de la cual se hace la copia se le llama tabla maestra y a la localidad que contiene la tabla se le llama localidad maestra.

La instantánea es actualizada en forma periódica y el periodo de tiempo en que se debe actualizar es definido al crear la instantánea.

3.1.7.1. Instantáneas simples e instantáneas complejas

Una instantánea simple consiste en una copia exactamente igual a la tabla maestra, es decir que cada fila de la instantánea es igual a cada fila en la tabla maestra. Para cumplir con esta definición al crear la instantánea simple la consulta no puede tener agrupaciones, uniones de varias tablas, operadores de conjuntos o subconsultas. Si la consulta de la instancia tiene cualquiera de estas instrucciones entonces se le llama una instantánea compleja.

3.2. Solución Oracle *Data Guard*

3.2.1. Definición

Oracle *Data Guard* es el *software* para la administración y monitoreo que trabaja con una base de datos de producción y una o más bases de datos *standby* para proteger los datos contra fallas, errores y corrupción que pueda destruir la base de datos.

Una base de datos *standby* es una réplica de la base de datos creada de un *backup* de una base de datos primaria. Aplicando *archived redo logs* de la base de datos primaria hacia la o las bases de datos *standby*, se logra mantener dos bases de datos sincronizadas.

Una base de datos *standby* tiene los siguientes propósitos:

- Protección contra desastre.
- Protección contra corrupción en los datos.
- Abrir la base de datos en modo de solo lectura. Reportes suplementarios.

Si la base de datos es destruida completamente o si los datos comienzan a corromperse, entonces se puede implementar un *failover* hacia la base de datos *standby*, en este caso la base de datos *standby* comienza a ser la nueva base de datos primaria. Esta base de datos también puede abrirse con la opción de solo lectura, por lo consiguiente funciona como una base independiente para propósitos de reportes.

En la versión de Oracle 8i esta solución se conocía como Oracle *Standby* pero en Oracle 9i fue renombrada como *Data Guard*.

3.2.2. Restricciones

Dentro de una base de datos *standby* se puede encontrar ciertas restricciones en cuanto a su configuración se refiere:

- Una configuración *Data Guard* debe usar la misma edición de la base de datos de Oracle *Enterprise Edition* y ésta debe instalarse sobre todos los sistemas.

- La base de datos primaria debe correr en modo *ARCHIVELOG*.

- La misma versión del *software* Oracle debe ser usada sobre ambas, la base primaria y las bases *standby*. El sistema operativo que corre sobre la base primaria y *standby* debe ser el mismo, pero en el sistema operativo no se necesita que sea la misma versión. En adición, la base de datos *standby* puede usar una estructura de directorio diferente al de la base de datos primaria.

- La arquitectura de *hardware* y sistema operativo sobre la localidad primaria y *standby* debe ser la misma. Por ejemplo, esto significa que una configuración *Data Guard* con una base de datos primaria sobre un sistema Sun de 32 bit debe tener una base de datos *standby* con un sistema Sun de 32 bit.

- Cada base de datos primaria y *standby* deben tener su propio *control file*.

- Si actualmente corre Oracle *Data Guard* sobre el *software* de base de datos Oracle 8i, se debe actualizar a Oracle 9i *Data Guard*.

- Las cuentas de usuario para administrar las instancias de bases de datos tanto primaria como *standby* deberán tener privilegios de sistema SYSDBA.

- Se debe usar la misma versión, *release*, y parche (*patch*) de ORACLE RDBMS para las bases de datos primaria y *standby* para que las operaciones de *failover* no estén comprometidas.

- En Oracle *Data Guard* se puede tener hasta nueve bases de datos *standby*.

- En versiones de Oracle para Windows NT previas a Oracle8 *release* 8.0.4 soportan únicamente un Oracle *home*. Se puede instalar productos Oracle7 *release* 7.x y Oracle8 *release* 8.x en el mismo Oracle *home*. Sin embargo, no se puede instalar múltiples *releases* con un tercer dígito del mismo producto. Por ejemplo, no se puede instalar productos Oracle7 *release* 7.3.2 y Oracle7 *release* 7.3.3 en la misma computadora, ya que una instalación escribe sobre la previa.

- En versiones Oracle8 *releases* 8.0.4 a 8.0.6, se puede instalar una o más *releases* de Oracle en múltiples Oracle *homes*. Se puede instalar Oracle8 *release* 8.0.x y Oracle8i *release* 8.1.3 o bien Oracle7 *release* 7.x y Oracle8 *release* 8.0.x en diferentes Oracle *homes* en la misma computadora. También se puede instalar diferentes *releases* de Oracle en el mismo Oracle *home* que tengan diferente el primer o segundo dígito del número de *release*. Por ejemplo, instalar Oracle7 *release* 7.2 y Oracle8 *release* 8.0.x en el mismo Oracle *home*.

- Para instalar Oracle8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) tienen la misma funcionalidad que el inciso anterior pero contienen las siguientes restricciones:

    - No se puede instalar cualquier *release* de Oracle8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) en un Oracle *home* que fue creado con el instalador antiguo llamado Oracle *Installer* y fue usado para instalaciones previas a Oracle8i *release* 8.1.3; el nuevo instalador es llamado Oracle *Universal Installer* y está basado en Java.
    - Los *releases* Oracle 8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) deben ser instalados en Oracle *homes* separados. No se puede tener mas de un *release* instalado en cada Oracle *home*.

3.2.3. Requerimientos mínimos

El *software* que se necesita para instalar Oracle *Data Guard* sobre cada servidor (primario y *standby*) es el siguiente:

Tabla II. Requerimientos mínimos de sistema operativo para la solución Oracle *Data Guard*

Plataforma	Sistema Operativo	Parche requerido
Basada en *Windows*	*Windows* NT 4.0	*Service pack* 5
Basada en *Windows*	*Windows* 2000 (*Server*, *Advanced Server*, *DataCenter*)	*Service pack* 1

- Licencia de *Windows* 2000 (*Server, Advanced Server, Datacenter*) con *service pack* 1 o mayor.

- TCP/IP o TCP/IP con SSL

- Licencia por procesador o número de usuarios de Oracle 8/8i/9i *Enterprise Edition* sobre cada servidor de base de datos.

- Hay que tener en cuenta que siempre se debe tener los últimos parches.

- Idealmente, ambos servidores deben ser idénticos en *hardware*, *drivers* o controladores, *software*, y configuración. La igualdad física de un servidor a otro, evitará que se tenga mayores problemas.

- Netscape Navigator 4.76 o mayor.

- Microsoft Internet Explorer 5.0 o mayor.

Para instalar la base de datos Oracle 9i bajo plataformas intel o compatible, se necesita el *hardware* que se muestra en la siguiente tabla.

Tabla III. Requerimientos mínimos de *hardware* para la solución Oracle *Data Guard* sobre plataforma basada en *Windows*

Hardware	**Requerimientos**
Procesador	Pentium 266 o mayor (para Oracle 9i *Enterprise Edition*)
Memoria	RAM: 128 MB (recomendado 256 MB) Memoria virtual: el doble de la memoria física
Espacio en disco duro	Drive del sistema 140 MB, Oracle *home drive* (NTFS) 2.85 GB para *Enterprise Edition*

Para instalar la base de datos Oracle 9i bajo sistemas Unix, se necesita *hardware* que se muestra en la siguiente tabla.

Tabla IV. Requerimientos mínimos de *hardware* para la solución Oracle *Data Guard* sobre plataforma basada en Unix

Hardware	**Requerimientos**
Memoria	Un mínimo de 512 MB de RAM
Swap Space	Un mínimo del doble de RAM, o 400 MB, cualquiera de los dos
Espacio en disco duro	4.5 GB

- Una consideración muy importante es cuando se selecciona el *hardware* del servidor puesto que éste debe ser certificado en la lista de compatibilidad de Oracle.

3.2.4. Funcionamiento

En Oracle *Data Guard* se introducen nuevos conceptos como son los siguientes:

Base de datos primaria es una base de datos en producción. Ésta es usada para crear una base de datos *standby*. Cada base de datos *standby* es asociada solamente con una base de datos primaria.

Base de datos *Standby*, puede ser una base de datos física o lógica, es creada de una réplica de un *backup* de una base de datos primaria.

Una base de datos ***standby* física** es físicamente idéntica a la base de datos primaria bloque por bloque.

Esto es actualizado mediante la realización de recuperación de los *redo logs* que son generados en la base de datos primaria.

Data Guard mantiene una base de datos *standby* física realizando operaciones de recuperación administrada. Cuando no se está realizando una operación de recuperación, una base de datos *standby* física puede ser abierta para operaciones de solo lectura.

La recuperación administrada de una base de datos *standby* es mantenida aplicando los *archived redo logs* sobre el sistema *standby* usando el mecanismo de recuperación de Oracle. La operación de recuperación aplica los cambios bloque por bloque usando el *row* ID físico. La base de datos no puede ser abierta para operaciones de lectura o escritura mientras los *redo data* están siendo aplicados.

La base de datos *standby* física puede ser abierta para operaciones de solo lectura en que se puedan ejecutar consultas sobre la base de datos. Mientras está abierta para operaciones de solo lectura, la base de datos *standby* puede continuar recibiendo los *redo logs* pero la aplicación de los datos de los *logs* es en diferido mientras la base de datos cambia a operaciones de recuperación administrada.

Aunque la base de datos *standby* no puede realizar ambas operaciones al mismo tiempo, se puede cambiar entre ellas. Por ejemplo, se puede usar una base de datos *standby* física para realizar operaciones de recuperación administrada, entonces ésta puede ser abierta para que las aplicaciones puedan realizar operaciones de solo lectura para correr reportes y luego cambiarla nuevamente a operaciones de recuperación administrada para aplicar los *archived redo logs*. Se puede repetir este ciclo alternando entre ellos según sea necesario.

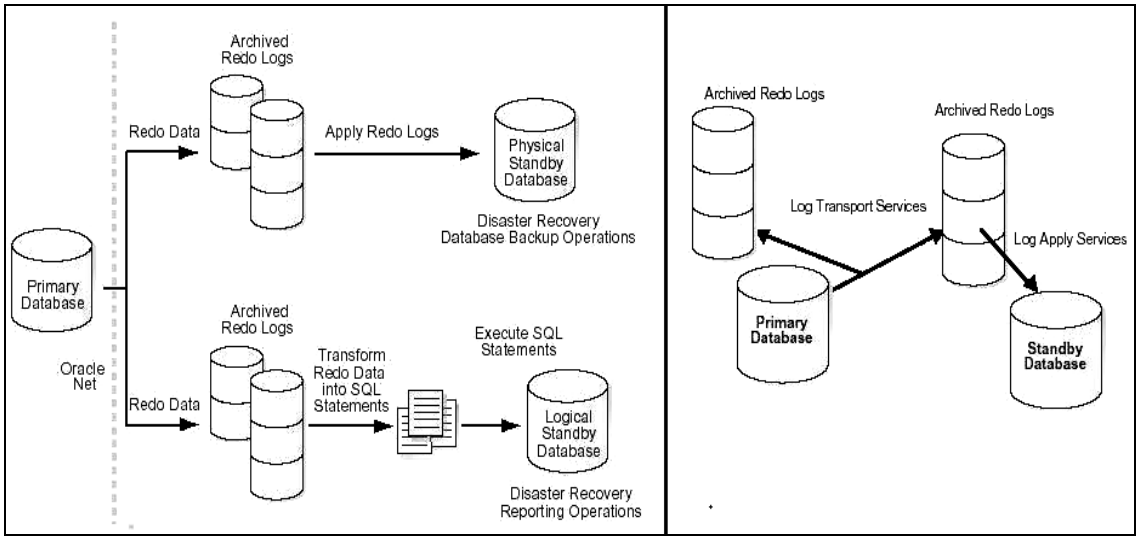
En otro caso, si la base de datos *standby* física está disponible para realizar operaciones de *backup*. Puede continuar recibiendo los *redo logs* incluso si ellos no están siendo aplicados en ese momento.

Una base de datos *standby* lógica contiene la misma información lógica que la base de datos de producción, aunque la organización física y la estructura de los datos puede ser diferente. Esto se mantiene sincronizado con la base de datos primaria transformando los datos recibidos en los *redo logs* de esta base dentro de declaraciones SQL, las cuales son ejecutadas en la base de datos *standby*. Esta base de datos *standby* lógica puede ser usada para propósitos de negocios en adición a requerimientos de recuperación de desastres. Esto permite a los usuarios accesar a la base de datos *standby* lógica para propósitos de consultas y reportes en cualquier momento. Así, una base de datos *standby* lógica puede ser usada concurrentemente para protección de datos y reportes.

A continuación se muestra la figura de una configuración *Data Guard* que contiene una instancia de la base de datos primaria que transmite los redo data hacia las bases de datos *standby* física y lógica y ambas se encuentran en localidades remotas de la instancia de la base de datos primaria.

En esta configuración, una base de datos *standby* física es configurada para operaciones de recuperación de desastres y operaciones de *backup*, y una base de datos *standby* lógica es configurada primariamente para propósitos de reportes, pero ésta también puede ser usada para recuperación de desastres.

Figura 17. Configuración típica de la solución Oracle *Data Guard* y los servicios de transporte y aplicación de los *redo logs*



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96524.pdf

Data Guard utiliza servicios para administrar la transmisión de los *redo data*, la aplicación de los *redo logs* y los cambios de role de la base de datos

Log transport services

Este servicio controla la trasferencia automática de los *redo data* dentro de una configuración *Data Guard, y realiza las siguientes tareas:*

- Transmitir los *redo data* del sistema primario a los sistemas *standby* en la configuración.
- Hacer cumplir los modos de protección de la base de datos.

Log apply services

Los redo data transmitidos de la base de datos primaria son archivados sobre el sistema *standby* en la forma de *archived redo logs*. Este servicio automáticamente aplica los *archived redo logs* sobre la base de datos *standby* para mantener sincronización transaccional con la base de datos primaria y permitir acceso a los datos únicamente de lectura consistentemente transaccionales.

La principal diferencia entre las bases de datos *standby* física y lógica es la manera cómo el servicio *Log Apply* aplica los *archived redo logs*:

- Para una base de datos *standby* física, *Data Guard* usa tecnología *redo apply*, la cual aplica los *redo data* sobre la base de datos *standby* usando técnicas de recuperación estándar del servidor de base de datos Oracle.
- Para una base de datos *standby* lógica, *Data Guard* usa tecnología SQL *apply*, la cual primero transforma los *redo data* recibidos dentro de declaraciones SQL y entonces se ejecutan las declaraciones SQL generadas sobre la base de datos *standby* lógica.

El servicio *Log apply* realiza las siguientes tareas:

- Aplicación automática de los *archived redo logs* sobre la base de datos *standby.*

- Detección automática del extravío de los *redo logs* sobre un sistema *standby* y automáticamente recupera de los redo logs extraviados de la base de datos primaria u otra base de datos *standby*.

Servicios de administración de roles

Una base de datos Oracle opera en uno de dos roles: primario o *standby*. Usando *Data Guard*, se puede cambiar el rol de una base de datos usando una operación de *switchover* o *failover*. Los servicios que controlan esos aspectos son llamados servicios de administración de roles.

Un ***Switchover*** es un rol cambiante entre la base de datos primaria y una de sus bases de datos *standby*. Una operación de *switchover* garantiza ninguna pérdida de datos. Esto es usado para propósitos de mantenimiento planeado sobre el sistema primario. Durante un *switchover*, la transición de la base de datos primaria a el rol de *standby* y la transición de la base de datos *standby* para el rol primario ocurren sin tener que recrear la base de datos.

Un ***failover*** es una transición irreversible de una base de datos *standby* hacia el rol primario. Esto es únicamente aplicado en el evento de una falla catastrófica de la base de datos primaria. El administrador de la base de datos puede configurar *Data Guard* para asegurar ninguna pérdida de datos.

El ***Data Guard broker*** es un marco de trabajo distribuido que automatiza y centraliza la creación, mantenimiento y monitoreo de las configuraciones de *Data Guard*. La siguiente lista describe algunas de las operaciones que el *Data Guard broker* automatiza o simplifica:

- Crea y habilita una o más configuraciones *Data Guard*, incluyendo configuración de los servicios de transporte de log y de aplicación de log.
- Crea una base de datos *standby* física o lógica de una copia de seguridad de la base de datos primaria.
- Agrega nuevas bases de datos *standby* o existentes a una configuración *Data Guard*.
- Administra una configuración *Data Guard* completa de cualquier sistema en la configuración.
- Monitorea las tasas de aplicación de *logs*, capturando información de diagnóstico y detectando rápidamente problemas con el monitoreo centralizado.

Modos de protección de *Data Guard*

En algunas situaciones, los negocios no pueden afrontar la pérdida de datos a ningún costo. En otras situaciones, la disponibilidad de la base de datos quizá sea más importante que la pérdida de datos. Algunas aplicaciones requieren máximo rendimiento de la base de datos y poder tolerar una potencial pérdida de datos. *Data Guard* proporciona tres modos distintos de protección:

El modo de **máxima protección** ofrece el más alto nivel de protección de los datos. Estos son transmitidos de una forma sincronizada de la base de datos primaria a la base de datos *standby*, y las transacciones no son comprometidas sobre la base de datos primaria a no ser que los *redo data* estén disponibles en al menos una base de datos *standby* configurada en este modo. Si la ultima configuración de la base de datos *standby* no está disponible, el proceso se detiene en la base de datos primaria. Este modo garantiza ninguna pérdida de datos.

El modo de **máxima disponibilidad** es similar al modo de máxima protección, incluyendo la garantía de ninguna pérdida de datos. Sin embargo, si una base de datos *standby* llega a ser no disponible (por ejemplo, debido a problemas de conexión de la red), el proceso continua sobre la base de datos primaria. Cuando la falla es corregida, la base de datos *standby* es resincronizada con la base de datos primaria. Si hay necesidad de un *failover* antes de la resincronización con la base de datos *standby*, algunos datos se pueden perder.

El modo de **máximo rendimiento** ofrece ligeramente menos protección a los datos en la base de datos primaria, pero alto rendimiento respecto del modo de máxima disponibilidad. En este modo, las transacciones procesadas en la base de datos primaria, los *redo data* son enviados asincrónicamente hacia la base de datos *standby*. La operación de *commit* en la base de datos primaria no espera para que la base de datos *standby* reciba reconocimiento de que han sido recibidos los *redo data* antes de completar las operaciones de escritura sobre la base de datos primaria. Si cualquier destino *standby* llega a ser no disponible, el proceso continua en la base de datos primaria y hay un pequeño efecto sobre su rendimiento.

3.2.5. Confiabilidad

Una base de datos *standby* es confiable debido a que es una réplica de la base de datos creada de un *backup* o base de datos primaria. Esta confiabilidad se logra aplicando *archived redo logs* de la base de datos primaria hacia la base de datos *standby*, con ello se puede mantener las dos bases de datos sincronizadas.

Si una base de datos primaria es destruida o bien los datos se corrompen, se puede hacer un *failover* hacia la base de datos *standby*, en este caso la base de datos *standby* llega a ser la nueva base de datos primaria. De igual manera, se puede abrir una base de datos *standby* con la opción de solo lectura, lo cual permite que ésta funcione como una base de datos independiente para propósitos de reportes.

3.2.6. Rendimiento

El rendimiento se ve beneficiado al usar la tecnología de *redo apply* la cual es usada por la base de datos *standby* física que aplica los cambios usando mecanismos de recuperación de bajo nivel, lo cual es una vía de paso para todo a nivel de código SQL y de tal modo el mecanismo es más eficiente de aplicar los cambios. Esto hace la tecnología de *redo Apply* un mecanismo altamente eficiente de propagar los cambios entre las bases de datos.

Una base de datos *standby* lógica puede permanecer abierta al mismo tiempo que las tablas son actualizadas desde la base de datos primaria, y esas tablas son simultáneamente disponibles para accesos a lecturas. Esto hace que una base de datos *standby* lógica sea una buena elección para hacer consultas, actividades de reportes, además se puede descargar la base de datos primaria aumentando con ello el rendimiento con lo cual se puede ahorrar valiosos ciclos de CPU y de entradas y salidas a disco.

Esta solución de Oracle *Standby* puede mejorar el rendimiento del servidor de base de datos primario si éste se encuentra demasiado sobrecargado.

Por ejemplo, si un departamento necesita hacer reportes de un cierre del mes, entonces en lugar de hacerlos en la máquina primaria, la cual no solo está procesando transacciones y atendiendo a varios usuarios, al aplicarle el reporte a la misma bajaría el rendimiento suponiendo que éste se lleva aproximadamente 3 horas, con ello se sobrecarga al procesador del sitio primario y obviamente provocaría un bajo rendimiento del sitio primario; mientras que si este reporte se efectúa en la base de datos *standby* posiblemente le tome aproximadamente 30 minutos, con lo cual se puede colocar en modo de solo lectura y con ello se estaría utilizando ese procesador que básicamente no está efectuando nada más que aplicando los *archived redo logs*. Con este ejemplo se hace un mejor uso del sistema primario como del sitio secundario, sin afectar su rendimiento, al contrario, se mejora el rendimiento del sitio primario.

3.2.7. Disponibilidad

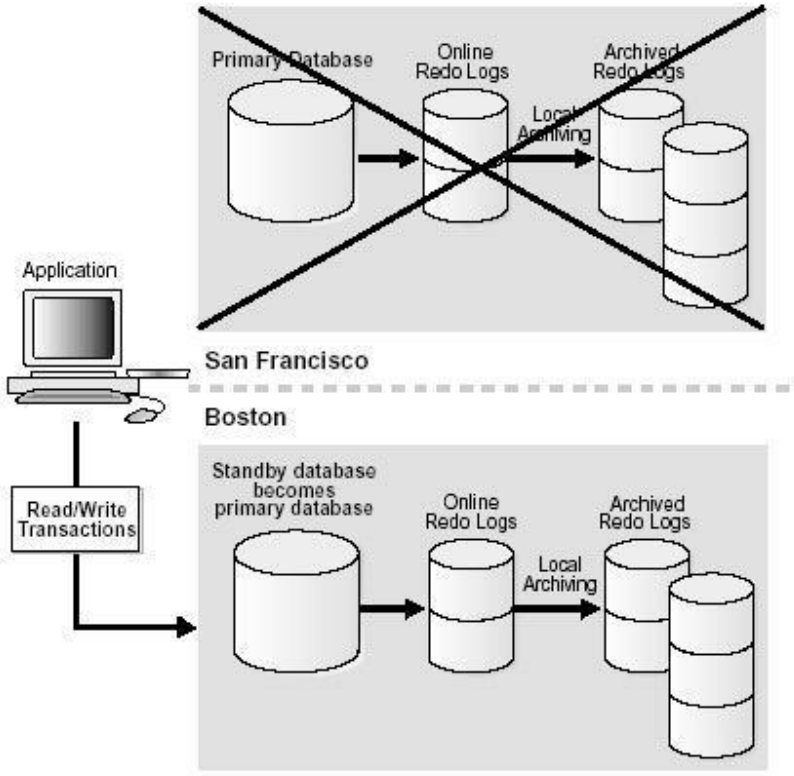
Data Guard proporciona una eficiente y comprensiva solución de recuperación a desastres, protección de datos y alta disponibilidad. Fácil administración de *switchover* y *failover* permitiendo cambiar entre bases de datos primaria y *standby*, minimizando el tiempo fuera de servicio planeados y no planeados (*downtime*) de la base de datos primaria.

En cuanto a disponibilidad se refiere la base de datos *standby* llega a activarse sólo si una falla es detectada por el DBA y éste mismo decide si la activa o se soluciona el problema en el sitio primario. Regularmente lo que más falla en un servidor son los discos y se puede solventar el problema cambiando el mismo y aplicando un *backup*.

Por el contrario, si dentro de ese disco la parte que se dañó corresponde al diccionario de datos de la base de datos el administrador de la base de datos deberá de activar la base *standby* para convertirla en la base de datos primaria. Esta acción no es transparente al usuario y toma cierto tiempo que la base de datos *standby* se convierta en la primaria.

La figura describe una operación de *failover* de una base de datos primaria en San Francisco hacia una base de datos *standby* en modo de recuperación administrada en Boston.

Figura 18. *Failover* de una base de datos primaria hacia una base de datos *standby*



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96524.pdf

3.2.8. Crecimiento

La base de datos *standby* no proporciona el beneficio de escalabilidad horizontal ya que esta solución no está basada en *clusters*.

3.2.9. Ventajas

- *Data Guard* ofrece recuperación a desastre, protección de datos y alta disponibilidad.

- Completa protección de datos, con sus bases de datos *standby*, *Data Guard* garantiza ninguna pérdida de datos, incluso en un desastre imprevisto. Una base de datos proporciona un salvavidas contra la corrupción de los datos y errores de usuario. Las corrupciones a nivel de almacenamiento físico en la base de datos primaria no son propagadas hacia la base de datos *standby*. Similarmente, las corrupciones lógicas o errores de usuario que causan que la base de datos primaria sea dañada permanentemente puedan ser resueltos. Finalmente, los *redo data* son validados cuando se aplican en la base de datos *standby*.

- Eficiente uso de los recursos del sistema, las tablas de la base de datos *standby* que son actualizadas con los redo logs recibidos de la base de datos primaria pueden ser usados para otras tareas tales como operaciones de *backup*, reportes y consultas, de tal modo reduce la carga de trabajo necesaria para realizar esas tareas, ahorrando ciclos de CPU y entrada o salida en la base de datos primaria. Con una base de datos *standby* lógica, los usuarios pueden realizar operaciones de manipulación de los datos normalmente sobre las tablas en esquemas que son actualizados de la base de datos primaria.

- Una base de datos *standby* puede quedar abierta mientras las tablas son actualizadas desde la base de datos primaria, y las tablas son simultáneamente disponibles para accesos de solo lectura.

- Flexibilidad en la protección de los datos para balancear los requerimientos de disponibilidad contra rendimiento, Oracle *Data Guard* ofrece máxima protección, máxima disponibilidad y máximo rendimiento para ayudar a las empresas a balancear los requerimientos de disponibilidad de los datos contra el rendimiento del sistema.

- Administración simple y centralizada, el *Data Guard broker* proporciona la interface gráfica de usuario *Data Guard Manager* y la interface de línea de comando *Data Guard command line* para automatizar la administración y la operación de las tareas a través de múltiples bases de datos en una configuración *Data Guard*. El *broker* también monitorea todo de los sistemas dentro de una sola configuración *Data Guard*.

- Detección y resolución automática de aberturas, si la conectividad se pierde entre la base primaria y una o más bases de datos *standby* por ejemplo, debido a problemas de red, *redo data* generados en la base de datos primaria no pueden ser enviados a las bases de datos *standby*. Una vez es establecida la conexión nuevamente, la secuencia de *log* extraviados (o abertura) son automáticamente detectados por *Data Guard*, y los *redo logs* necesarios son automáticamente transmitidos hacia las bases de datos *standby*. Las bases de datos *standby* son resincronizadas con la base de datos primaria, con ninguna intervención manual por parte del administrador de la base de datos.

- Mantener una base de datos *standby* en una localidad que esté ubicada geográficamente remota de la base de datos primaria o mantener varias bases de datos *standby* en diversas localidades geográficas.

- Hacer una base de datos *standby* la nueva base de datos primaria con mínima pérdida de tiempo y de datos si la base de datos primaria está completamente destruida.

- Provee posible protección contra lotes de trabajos erróneos, errores de usuario o aplicación que corrompa la base de datos primaria por no aplicar *logs* archivados que contienen datos corruptos a la base de datos *standby*. Se puede entonces activar la base de datos *standby* no corrupta, haciéndola base de datos primaria.

- Una base de datos *standby* puede ser un gran beneficio para las estrategias de *backup* y recuperación.

3.2.10. Desventajas

- Falta de escalabilidad.

- Costo, Oracle cobra las licencias que serán usadas en la base de datos *standby*.

- La activación de una base de datos *standby* la debe efectuar el administrador de la base de datos, aunque la misma se encuentre en modo de recuperación administrada.

- Operaciones *UNRECOVERABLE* o *NOLOGGING* no son propagadas.

- Pérdida de los datos debido a la pérdida de los *redo logs* en línea, esto ocurre cuando es manual y no administrada.

3.3. Solución Oracle *Real Application Clusters*

3.3.1. Definición

El *software Real Application Clusters* (*RAC*) y una colección de *hardware* llamada *cluster* utilizan el potencial de cada componente para crear un potente entorno de computación. *RAC* provee escalabilidad y alta disponibilidad debido a que se puede agregar nodos al *cluster* con ello se proporciona la recuperación de falla si los componentes del nodo fallaran. En un entorno de *RAC* todas las instancias activas pueden concurrentemente ejecutar transacciones contra una base de datos compartida. *RAC* coordina cada acceso de las instancias para compartir los datos con lo cual proporciona consistencia e integridad de los datos.

RAC proporciona disponibilidad escondiendo las fallas al usuario final sin interrumpir el acceso a los datos. *Transparent Application Failover* (TAF) es una aplicación que transparentemente redirecciona las consultas de los clientes de la base de datos, hacia un nodo disponible en el *cluster* cuando éste falla. Con esta aplicación el cliente no ve el mensaje de error que describe la pérdida del servicio.

3.3.2. Restricciones

- En versiones de Oracle para *Windows* NT previas a Oracle8 *release* 8.0.4 soportan únicamente un Oracle *home*. Se puede instalar productos Oracle7 *release* 7.x y Oracle8 *release* 8.x en el mismo Oracle *home*. Sin embargo, no se puede instalar múltiples *releases* con un tercer dígito del mismo producto. Por ejemplo, no se puede instalar productos Oracle7 release 7.3.2 y Oracle7 release 7.3.3 en la misma computadora, ya que una instalación escribe sobre la previa.

- En versiones Oracle8 *releases* 8.0.4 a 8.0.6, se puede instalar una o mas *releases* de Oracle en múltiples Oracle *homes*. Se puede instalar Oracle8 *release* 8.0.x y Oracle8i *release* 8.1.3 o bien Orale7 *release* 7.x y Oracle8 *release* 8.0.x en diferentes Oracle *homes* en la misma computadora. También se puede instalar diferentes *releases* de Oracle en el mismo Oracle *home* que tengan diferente el primer o segundo dígito del número de *release*. Por ejemplo, instalar Oracle7 *release* 7.2 y Oracle8 *release* 8.0.x en el mismo Oracle *home*.

- Para instalar Oracle8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) que tienen la misma funcionalidad que el inciso anterior pero contienen las siguientes restricciones:

- No se puede instalar cualquier *release* de Oracle8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) en un Oracle *home* que fue creado con el instalador antiguo llamado Oracle *Installer* y fue usado para instalaciones previas a Oracle8i *release* 8.1.3; el nuevo instalador es llamado Oracle *Universal Installer* y está basado en Java.

- Los *releases* Oracle 8i *release* 8.1.3 a Oracle9i *release* 2 (9.2) deben ser instalados en Oracle *home* separados. No se puede tener más de un *release* instalado en cada Oracle *home*.

- *RAC* puede correr con cualquier base de datos Oracle creada en modo exclusivo (x). Oracle en modo exclusivo puede acceder una base de datos creada o modificada por el *software RAC*. En adición a ello, cada instancia debe tener su propio juego de *redo logs*.

- La base de datos en modo de *cluster*, no soporta las siguientes operaciones:

    - Crear una base de datos (*create database*)
    - Crear un control file (*create control file*)
    - Cambiar el modo de *archiving* de la base de datos (las opciones *ARCHIVELOG* y *NOARCHIVELOG* de *alter database*)
    - Para desarrollar esas opciones se debe apagar (*shutdown*) todas las instancias e iniciar (*startup*) una instancia en modo exclusivo.

- El número de *datafiles* soportados por Oracle es específico en el sistema operativo. Dentro de este límite, el número máximo permitido depende de los valores usados en el comando *create database*. Esto es limitado por el tamaño físico del *control file*.

- Se pueden agregar nodos dinámicamente al *cluster* pero esto depende de las capacidades del sistema operativo. Si se esta usando Oracle *clusterware*, entonces dinámicamente se pueden agregar nodos e instancias sobre diferentes plataformas.

Dentro de las limitaciones de *TAF* se pueden encontrar que el efecto de cualquier declaración *Alter Session* se perderá, lo cual incluye pérdida de las tablas temporales globales: cualquier paquete PL/SQL se perderá y las transacciones que involucren declaraciones de insertar, actualizar o borrar, se perderán ya que no pueden ser manejadas automáticamente por *TAF*.

- Por *default*, cuando un *TAF* ha iniciado y ocurre un *failover*, *Net8* tomará solamente una conexión hacia una instancia de *backup*, pero usando los parámetros *retries* y *delay*, se puede cambiar el comportamiento de *Net8* haciendo múltiples intentos para conectar a un *backup* de la base de datos.

3.3.3. Requerimientos mínimos

Los requerimientos mínimos de *software* en cada nodo de un *cluster* se muestran en la siguiente tabla.

Tabla V. Requerimientos mínimos de sistema operativo para la solución Oracle *Real Application Clusters*

Plataforma	**Sistema operativo**	**Parche requerido**
Basada en *Windows*	*Windows* NT 4.0	*Service pack* 5
Basada en *Windows*	*Windows* 2000 (*Server*, *Advanced Server*, *DataCenter*)	*Service pack* 1
Compaq Tru64 Unix	Tru64 5.1	5.1 *patchkit* 4
Compaq Tru64 Unix	Tru64 5.1A	5.1A *patchkit* 1

- TCP/IP o TCP/IP con SSL

- *Netscape Navigator* 4.76 o mayor / *Microsoft Internet Explorer* 5.0 o mayor

- Se instala Oracle 8/8i *Enterprise Edition* para OPS u Oracle 9i para RAC sobre ambos nodos.

- Licencia por procesador o número de usuarios de Oracle 8/8i/9i *Enterprise Edition* sobre cada nodo del *cluster*.

- Hay que tener en cuenta que siempre se debe tener los últimos parches.

- Idealmente, ambos servidores deben ser idénticos en *hardware*, *drivers* o controladores, *software*, y configuración. La igualdad física de un servidor a otro evitará que se tenga mayores problemas.

Requerimientos de *hardware*
Cada nodo requiere:

- Discos externos compartidos.

- Configuraciones de *hardware* certificado.

Configuraciones de *hardware* y red

Para instalar la base de datos Oracle 9i bajo plataformas intel o compatible se necesita el siguiente *hardware*, que se muestra en la siguiente tabla.

Tabla VI. Requerimientos mínimos de *hardware* para la solución Oracle *Real Application Clusters*

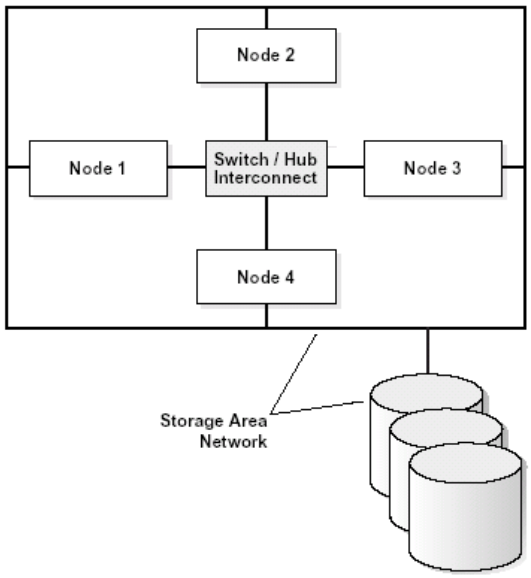
Hardware	Requerimientos
Procesador(es)	Pentium 266 o mayor (para Oracle 9i *Enterprise Edition*)
Memoria	RAM: 256 MB en cada instancia Memoria virtual: el doble de la memoria física
Espacio en disco duro	Drive del sistema 140 MB, Oracle *home* drive (NTFS) 2.85 GB para *Enterprise Edition*

- Nombres de red publica (conocidos como *host* o nombres *TCP/IP*) de cada nodo.
- Si se tiene una interconexión de alta velocidad, en la cual cada nodo tenga su propio nombre de red privada.
- Se usa hardware *Virtual Interface Architecture* (VIA) y si hay disponibilidad de *NIC* (*Network Interface Card*).

3.3.4. Funcionamiento

Una base de datos en *cluster* debe tener al menos dos nodos que son enlazados por una interconexión que comunica cada nodo en el *cluster* de la base de datos. Cada instancia de Oracle usa la interconexión para los mensajes que sincronizan cada uso de instancia de los recursos compartidos. Oracle también usa la interconexión para transmitir bloques sobre múltiples instancias compartidas. El tipo primario de recurso compartido son los *datafiles* que son accesados por todos los nodos. La figura de abajo es una vista de cómo los nodos se interconectan en un *cluster* de base de datos y cómo éste accesa los *datafiles* compartidos sobre los dispositivos de almacenamiento.

Figura 19. Interconexión de los nodos en un cluster de base de datos para Oracle RAC



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96597.pdf

El *cluster* y su interconexión son enlazadas al dispositivo de almacenamiento o subsistema de discos compartidos, por medio de *storage area network*.

RAC usa alta velocidad para el componente de interproceso de comunicación (*IPC*) para la comunicación entre nodos. Usualmente esta interconexión usa un ancho de banda grande, facilitando la comunicación entre los nodos enlazados y proporcionando baja latencia. Se puede usar *Ethernet*, un *Fiber Distributed Data Interface* (*FDDI*), u otro hardware para su interconexión. El IPC define protocolos e interfaces requeridas en un entorno *RAC* para transferir mensajes entre las instancias. Los mensajes son la unidad fundamental de comunicación entre las interfaces. El *IPC* es funcionalmente construido sobre modelo de mensajes en colas asincrónico.

Generalmente, cada nodo en un *cluster* de base de datos tiene uno o más CPU. Los nodos con múltiples CPU son configurados típicamente para compartir la memoria principal.

RAC requiere que todos los nodos tengan acceso simultáneo a los discos compartidos para dar a las instancias acceso concurrente a la base de datos. La implementación del subsistema de discos compartidos está basado en su sistema operativo o solución de *hardware.*

Las configuraciones para *RAC* son típicamente uniformes. Esto significa que la sobrecarga para cada nodo en el *cluster* al accesar la memoria es la misma.

Una base de datos *RAC* tiene los mismos procesos que una sola instancia de base de datos Oracle tales como monitor de proceso (*PMON*), escritor de base de datos (*DBWRn*), escritor de log (*LGWR*), etc. Pero hay procesos adicionales para *RAC* como se muestra en la figura. Los nombre son exactamente los mismos que pueden ser creados sobre plataformas dependientes.

Global Cache Service Process (*LMSn*), donde n es el rango que puede ir de 0 a 9 dependiendo de la cantidad de tráfico en mensajes, controla el flujo de mensajes a instancias remotas y maneja el acceso al bloque de datos global. Éste también transmite imágenes del bloque entre los *buffer* de cache de las diferentes instancias. Esta parte de procesamiento es conocido como *CACHE FUSIÓN*.

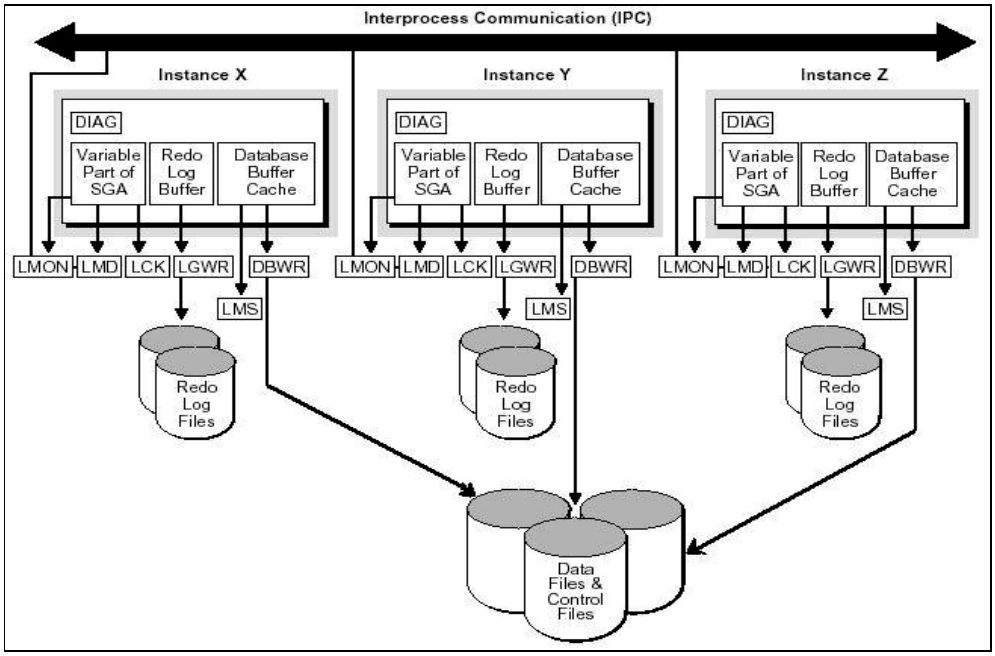
Global Enqueue Service Monitor **(***LMON*) monitorea las colas globales y los recursos a través del *cluster* y realiza operaciones de recuperación de colas globales. Las colas son estructuras de memoria compartida que serializan actualizaciones de línea.

Global Enqueue Service Daemos (*LMD*) administra las colas globales y el acceso al recurso global. El proceso *LMD* administra solicitudes de recursos remotos.

Lock process (*LCK*) maneja las solicitud de recursos no basados en *cache fusion* tales como librerías y solicitudes de *row* cache.

Diagnosability Daemon (*DIAG*) captura los diagnósticos de datos a cerca de las fallas de procesos entre instancias.

Figura 20 . Proceso de instancia específica de la solución Oracle *RAC*



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96597.pdf

Servicio *Global cache* y *Global enqueue*

El ***Global Cache Service*** (*GCS*) y ***Global Enqueue Service*** (*GES*) son componentes integrados de *RAC* que coordinan simultáneamente el acceso a la base de datos compartida y comparten los recursos entre la base de datos. Esos servicios mantienen consistencia e integridad de los datos. Estos servicios en cada instancia usan el *IPC* para comunicarse entre las instancias y dentro del *cluster*.

Estos servicios mantienen un *Global Resource Directory* para registrar la información acerca de los recursos. El *GRD* reside en memoria y está disponible para todas las instancias. En esta arquitectura distribuida, cada nodo participa en la administración de información en el directorio. Este esquema distribuido proporciona tolerancia a fallos y mejora el rendimiento en tiempo de corrida.

El *GCS* y *GES* aseguran la integridad del *GRD* incluso si múltiples nodos fallaran. El almacenamiento de la base de datos estaría accesible si al menos una instancia está activa después de que una recuperación sea completada.

El proceso *cache fusion*

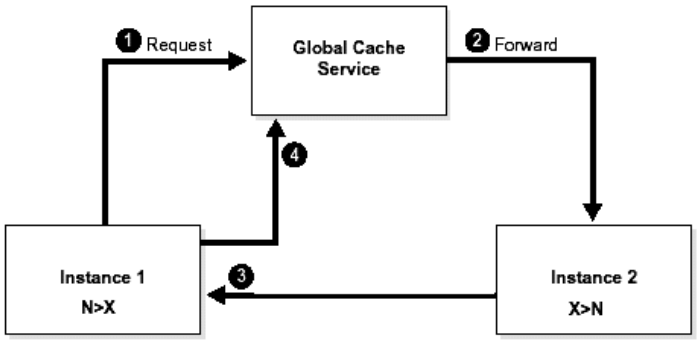
Debido a *cache fusion* y la eliminación de escrituras a disco que ocurren cuando otras instancias solicitan modificación para los bloques, el rendimiento es grandemente disminuido debido a la sobrecarga para administrar los datos compartidos entre las instancias.

Las lecturas concurrentes sobre múltiples nodos ocurren cuando dos instancias necesitan leer el mismo bloque de datos.

RAC resuelve esta situación sin sincronización porque múltiples instancias pueden compartir los bloques de datos para accesos de lectura sin conflictos de coherencia de *cache*.

Una solicitud de lectura de una instancia para un bloque que fue modificado por otra instancia y aún no ha sido escrita a disco puede ser una solicitud para una versión actual de el bloque o para una versión de lectura consistente. En ese caso, el (*LMSn*) transfiere el bloque de la instancia que lo tiene en cache hacia la instancia que lo solicita sobre la interconexión. La figura muestra que un bloque de datos ha sufrido modificación, por una instancia y está en modo exclusivo (X). Además, en este escenario se asume que el bloque ha sido accesado solo por la instancia que los cambió. Ésta es, la única copia existente a lo largo del *cluster*. En otras palabras, el bloque tiene un rol local (L).

Figura 21. Solicitud de cambio a un bloque para una operación de modificación



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96597.pdf

1. La instancia modifica el bloque, la instancia 1 envía una solicitud al *GCS*.

2. El *GCS* transmite la solicitud para mantenerla en la instancia 2.

3. La instancia 2 recibe el mensaje y el proceso *LMS* envía el bloque a la instancia 1. Antes de enviar el bloque, el recurso es bajado en la instancia 2 de modo exclusivo a nulo (N) y la instancia 2 retiene el *buffer* cambiado como una *PI*. El rol cambia a global (G) porque quizá el bloque permanezca diferente en más de una instancia. A lo largo del bloque, la instancia 2 comunica hacia las instancias solicitantes que la instancia 2 retiene en una PI (*past image* o imagen pasada) en modo nulo (N). En el mismo mensaje, la instancia 2 especifica que el solicitante debe retener el bloque en modo exclusivo (X) y con un rol global (G).

4. Al recibir el bloque, la instancia 1 informa a el *GCS* que lo mantiene en modo exclusivo y con un rol global. Hay que notar que el bloque no ha sido escrito a disco antes que el recurso no le sea permitido a la instancia 1.

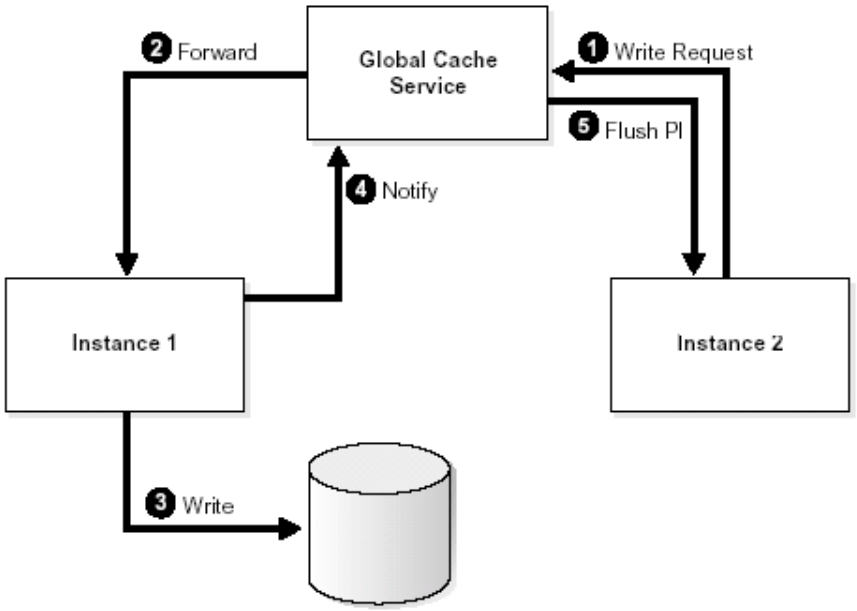
Las escrituras concurrentes sobre diferentes nodos ocurren cuando el mismo bloque de datos es modificado frecuentemente por diferentes instancias. En tal caso, la instancia que tiene el bloque completa su trabajo sobre éste después de recibir su solicitud. El *GCS* entonces convierte los recursos sobre el bloque para su administración global y el proceso *LMSn* transfiere una copia del bloque a la cache de la instancia que lo solicitó. En detalle seria de la siguiente manera:

- El *GCS* rastrea cada versión del bloque de datos y cada versión es referida como una *past image* (*PI*). En el evento de una falla, Oracle puede reconstruir la versión actual de un bloque, usando la información en un *PI*.

- La transferencia de datos *cache* a *cache* es efectuada por medio de interconexión de alta velocidad *IPC*, así elimina entradas o salidas a disco.

En la figura se ilustra cómo una instancia puede realizar un *checkpoint* en cualquier momento o reemplazar los *buffer* en el *cache* debido a solicitudes de liberación del *buffer*. Porque pueden existir múltiples versiones del bloque de datos con cambios en las instancias sobre el *cluster*, el *GCS* se asegura que únicamente la versión actual de los datos sea escrito a disco por medio de un protocolo que administra la escritura. Y también debe asegurarse que todas las versiones previas sean borradas en todas las otras caches.

Figura 22. Escritura de bloques a disco



Fuente: http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96597.pdf

1. La instancia 2 envía una solicitud de escritura a *GCS*.

2. El *GCS* envía la solicitud a la instancia 1, mantener el bloque actual.

3. La instancia 1 recibe la solicitud de escritura y escribe el bloque a disco.

4. La instancia 1 registra completamente la operación de escritura con el *GCS* y le informa que el rol del recurso se convirtió en local porque la instancia 1 realizó la operación de escritura del bloque actual.

5. Después de recibir la notificación, el *GCS* ordena que todas las *PI* mantenidas en las instancias sean descartadas. El *buffer* es liberado y el recurso previamente en modo nulo es cerrado.

En lo referente a *TAF*, éste proporciona los siguientes tipos de *failover*

- *Select failover*. Cuando es usado el *Select failover*, *Net8* mantiene la pista de cualquier declaración *Select* emitida en la transacción actual. Si la conexión de la instancia se pierde, *Net8* establece una conexión a la instancia de *backup*, ejecuta de nuevo la declaración *Select* y posiciona el cursor en donde sucedió la interrupción, entonces el cliente puede seguir leyendo líneas como si nada hubiese sucedido.

- *Session failover*. Cuando se pierde la conexión de una instancia, *SESSION failover* es el resultado de establecer una nueva conexión hacia una instancia de *backup*. Cualquier trabajo en progreso se pierde.

Dentro de este tipo de *failover*, Oracle ofrece dos submétodos:

- *Basic failover*: En este *failover* Oracle conecta hacia una instancia de *backup* únicamente después que una conexión primaria falla.

- *Preconnect failover:* En este *failover* Oracle conecta hacia un *backup* base de datos y una base de datos primaria.

3.3.5. Confiabilidad

Oracle provee continuidad ocultando las fallas al usuario final. Esto provee acceso a los datos continuamente e ininterrumpidamente, gracias a *TAF* en la base de datos ya que transparentemente la aplicación redirecciona la consulta (*query*) del cliente hacia un nodo disponible en el *cluster* cuando el nodo al que está conectado el cliente falla. Los clientes no ven el mensaje de error en la cual se pierde el servicio (Oracle 9i). Las fallas también son ocultadas debido a la redundancia de conexiones con otra base de datos evitando retardos si miles de usuarios deben migrar sus servicios durante una falla en el nodo con el cual estaban conectados. Esas capacidades proveen grandes beneficios para las aplicaciones que no pueden enfrentar un tiempo fuera de servicio (*downtime*).

3.3.6. Rendimiento

La tecnología de *Cache Fusion* implementada en Oracle *RAC* provee varios puntos clave que la hacen tener un mejor rendimiento.

- Utilización de caches de las bases de datos de todos los nodos en el sistema para satisfacer las solicitudes de la aplicación hacia cualquiera de los nodos.

- Remueve las operaciones de disco de un camino crítico en la sincronización entre nodos.

- Reduce grandemente el número requerido de mensajes de sincronización entre los nodos.

- Baja latencia entre los protocolos de interconexión del *cluster* para los mensajes y el envío de datos con un alto ancho de banda.

Otra característica importante que ayuda al rendimiento es la ejecución paralela de consultas sobre *RAC* en esta solución debido a que puede distribuir porciones de una gran declaración SQL a través de múltiples instancias. Con ello la transacción es completada más rápidamente puesto que la misma se ejecuta sobre múltiples CPU. En Oracle *RAC*, el *software* determina en tiempo de corrida si se aplicará un proceso de ejecución en paralelo en el servidor o únicamente en una instancia, o si esos procesos correrán sobre múltiples instancias. En general, Oracle intenta usar sólo una instancia cuando hay suficientes recursos disponibles. Esto reduce el tráfico de mensajes entre las instancias cruzadas.

3.3.7. Disponibilidad

RAC proporciona disponibilidad ya que su arquitectura está basada en *clusters*, los cuales proporcionan redundancia. La falla de un nodo no afecta la forma de procesar transacciones, mientras haya un nodo sobreviviente a lo largo del *cluster*, todos los clientes de la base de datos pueden procesar transacciones, aunque ellos posiblemente necesiten incrementar los tiempos de respuesta pero esto es debido a la capacidad de las reglas de validación sobre el nodo sobreviviente. Esta arquitectura también permite agrupar nodos para que estos se coloquen fuera de línea para propósitos de mantenimiento mientras el resto del *cluster* continúa proporcionando servicios en línea.

RAC puede agregar mayor disponibilidad si el sistema de almacenamiento compartido proporciona una solución basada en *hardware* que se conoce como *mirroring*, con la cual se dispone de redundancia en caso de que ocurra una falla en los discos.

En lo que se refiere a recuperación de desastres, estos sistemas de *cluster* no pueden ser usados contra ellos, ya que los desastres tales como terremotos, incendios, atentados, etc. Pueden destruir el sitio físicamente, y con ello se perdería por completo la base de datos.

Por lo anteriormente expuesto, *RAC* sobresale en áreas primarias con respecto a la alta disponibilidad:

- Oculta las fallas al usuario final.

- Entrega disponibilidad con N-1 nodos en un *cluster* con N nodos.

3.3.8. Crecimiento

Oracle *RAC* provee un entorno en el cual mejora el rendimiento y adiciona capacidad al agregar nodos. En algunas plataformas se puede agregar nodos dinámicamente mientras el *cluster* está corriendo.

El número de nodos que *RAC* puede soportar es significativamente mayor que cualquier otra aplicación. Los sistemas pequeños que están configurados para alta disponibilidad deben tener únicamente dos nodos. Sin embargo, mayores configuraciones permiten tener 32 y hasta 64 nodos.

3.3.9. Ventajas

Esas ventajas incluyen rendimiento de procesamiento y escalabilidad sobre sistemas con instancias únicas y tiempo de respuesta mejorado. Una solución *RAC* también provee una solución ideal de alta disponibilidad resolviendo la falla de un nodo en un ambiente de *clusters*.

- Escalabilidad sobre sistemas de una sola instancia, al agregar nodos al *cluster* para un mejor desempeño.

- Tiempo de respuesta.

- Transparencia.

- Debido a sus componentes redundantes, los cuales proveen servicio ininterrumpido, siempre en el evento de fallas de *hardware* o *software*, *RAC* provee alta disponibilidad si un nodo falla en el *cluster*, entonces el sistema no se ve afectado. En otras palabras, todos los usuarios tienen acceso a todo dato en donde haya un nodo disponible en un *cluster*. Esto significa que los datos están consistentemente disponibles.

- Debido a *cache fusión* de *RAC* se evita la entrada o salida hacia los discos, ya que si un bloque de datos fue actualizado por otro nodo, lo traslada del nodo remoto hacia el nodo que está procesando la consulta, de esa manera se ahorra la entrada o salida del disco.

3.3.10. Desventajas

- También es importante notar que hay serias limitaciones para la herramienta Oracle *TAF*. La limitación más seria es que Oracle *TAF* no soporta el reinicio de cualquier declaración de *DML* incluyendo *insert*, *update* y *delete*.

- Dentro de TAF se encuentran ciertas limitaciones que implican pérdida de tablas globales temporales, pérdida de paquetes PL/SQL. Las transacciones que involucren declaraciones *insert, update o delete* no pueden ser manejadas automáticamente por *TAF*, el efecto de cualquier declaración *alter session* se perderá.

- Esta solución *RAC* requiere tiempo fuera de servicio para actualizar el *software* de Oracle. Por lo tanto, los sistemas se deben detener para aplicar las actualizaciones de Oracle.

- *RAC* no proporciona recuperación de desastre, si el *cluster* en el que reside *RAC* se encuentra dentro de la misma habitación o dentro del mismo edificio y éste se destruye completamente.

- Oracle *RAC* requiere licencias adicionales, éstas pueden ser de acuerdo al número de procesadores con que cuente el nodo, es decir, que si cada nodo puede contener dentro de él dos procesadores y el *cluster* se compone de dos nodos ello significaría pagar cuatro licencias por procesador con lo cual esta solución en lo referente al costo de licencias es alta.

3.4. Solución *Shareplex* para Oracle

3.4.1. Definición

Shareplex para Oracle proporciona alta velocidad de replicación basada en *log* entre instancias Oracle sobre las siguientes plataformas *Windows* NT/2000, *Unix* y otras.

Shareplex replica únicamente los cambios realizados en el sistema de producción hacia el sistema alterno que puede ser remoto, esto es continuamente sin interrumpir el proceso de producción, con lo cual los datos son continuamente accesibles. Con Shareplex se pueden correr reportes contra estos datos y obtener los mismos resultados como si se estuviera corriendo un reporte sobre el sistema de producción, sin afectar la productividad de los usuarios en línea.

3.4.2. Restricciones

- Las bases de datos origen y destino deberían usar la misma versión de *SharePlex*. Para ejemplo se supondrá que está replicando la instancia "ora7" usando *SharePlex* 3.0, y está replicando la instancia "ora8" usando *SharePlex* 3.2. la versión *SharePlex* para la instancia destino ora7 debería ser *SharePlex* 3.0, y la versión *SharePlex* para la instancia destino ora8 debería ser *SharePlex* 3.2.

- *Shareplex* no puede replicar instancias corriendo sobre *OPS*, porque *OPS* usa múltiples *redo logs*.

- *Shareplex* no puede replicar ciertos componentes que son usados por Oracle, dentro de ellos se pueden mencionar los siguientes: *Alter Table* para borrar una columna, agregar una columna de objetos largos (*LOB, LARGE OBJETCS*), agregar una columna con una clave primaria o un *constraint* único, mover líneas, redefinir una columna *LOB, ALTER TABLE* comando que cambia el tamaño de una columna cuando hay dato en la misma. Si no hay dato en la tabla, si lo soporta, *DDL* sobre tablas particionadas, *DDL* para agregar una columna *LOB*, actividades que no están en los *redo* o *archive logs*, Tablas de *SYSTEM* tales como diccionarios de datos. La replicación del mismo objeto hacia otra base de datos podría causar corrupción . *Shareplex* regresará un mensaje de error, si se intenta activar una configuración que incluya una tabla del sistema, índices, *Index organized tables* (*IOT*). Esos pueden ser convertidos en tablas regulares si se desea replicar vistas, vistas de objetos, *Snapshot* y vistas materializadas, sinónimos, métodos, *stored procedures*, funciones, y paquetes, *triggers,* tipos de datos *NCLOB* Y *BFILE*, *VARRAY* y otro tipo de datos, tablas *clustered*.

3.4.3. Requerimientos mínimos

En sistemas con *Windows*:

- Mínimo de 300 MB de RAM.
- Tamaño del archivo de página adicional de 200 MB si el tamaño usado es 80% o más.
- Cliente Oracle 8.1.6.
- Servidor de Terminal MS o PC para proveer soporte técnico por medio de *login* remoto.

3.4.4. Funcionamiento

Cuando se activa una configuración *SharePlex* comienza la replicación, los datos son transportados a lo largo de una serie de colas por medio de una serie de cinco procesos, comúnmente llamados servicios, mientras llega a su destino en el sistema designado. Esos procesos se ejecutan automáticamente conforme sean necesitados, pero también pueden ser detenidos y reiniciados mediante comandos emitidos por un usuario de *SharePlex*.

- **El proceso de captura** lee uno o varios *redo log* para obtener los cambios a los objetos seleccionados para replicación, como se dijo en la configuración activa. El proceso de captura escribe los datos para los cambios a la cola de captura, donde se acumula mientras el siguiente paso del proceso *SharePlex* (lectura) está listo para ello. Hay un proceso de captura separado para que cada instancia a ser replicada, funcione de manera concurrente e independientemente.

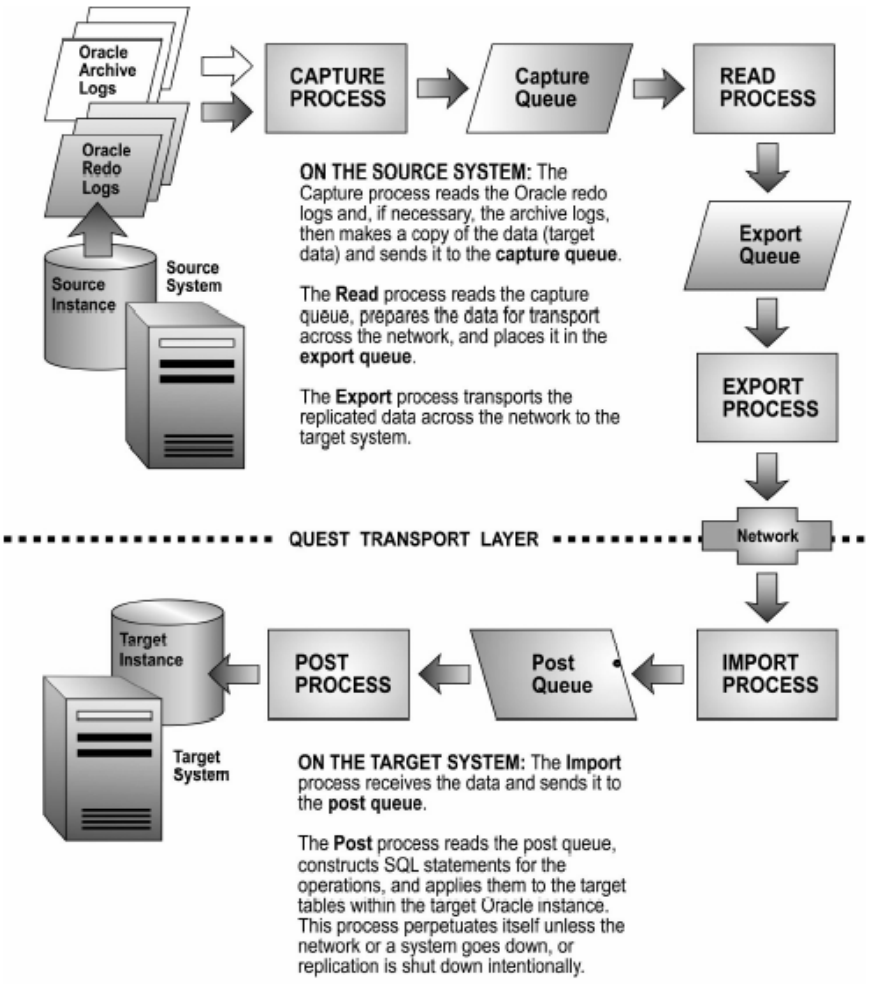
- **El proceso de lectura** es el siguiente paso en la replicación, agrega información de direccionamiento a la transacción (basada en el archivo de configuración) y construye una cola de exportación. Hay un proceso separado de lectura para cada origen de datos, cada uno funcionando concurrentemente e independientemente. En general, todos los procesos de lectura comparten la misma cola de exportación.

- **El proceso de exportación** opera sobre el sistema origen para leer los datos de la cola de exportación y lo envía a través de la red hacia un proceso de importación sobre el sistema destino. No importa cuántas configuraciones activas se hallan directamente replicando al sistema específico destino. El replicador crea un proceso de exportación para el sistema destino, compartido por todos los datos. Si hay dos sistemas destino, habrá dos procesos de exportación y así sucesivamente. El proceso de exportación es la primera parte de la exportación-importación "par de servicio", los cuales actúan como un transporte para mover los datos entre los dos sistemas sobre una red TCP / IP.

- **El proceso de importación** es el segundo medio del par de servicios exportación-importación, para cada proceso de exportación hacia un sistema destino hay un correspondiente proceso de importación sobre ese sistema. La función del proceso de importación recibe los datos del sistema origen y construye una cola post sobre el sistema destino una cola para cada fuente de datos comienza a ser replicada a ese sistema.

- **El proceso post** en cada sistema destino lee los datos que esperan en la cola post, construyendo declaraciones SQL para las operaciones de replicación y aplicando lo mismo a las tablas destino sobre el sistema destino. Múltiples procesos post, cada uno asignado a una combinación de instancias diferentes origen-destino, pueden operar simultáneamente sobre el sistema destino.

Figura 23. Muestra el proceso básico de replicación usado por el producto *Shareplex* para Oracle



Fuente: http://www.dlt.com/quest/solutions-availability.asp?sol=shareplx

Entendiendo la sincronización

Es importante configurar los sistemas y bases de datos para asegurarse que los datos origen y destino queden sincronizados. Resolviendo condiciones "fuera de sincronía" que pueden ser consumidas en tiempo y disociador para la actividad del usuario.

Para establecer y mantener un entorno de replicación sincronizada, se necesita un entendimiento básico de cómo replica *SharePlex* y cómo determina una condición de fuera de sincronía.

La definición básica de sincronización es que todas las tablas correspondientes en el origen y destino en la configuración de la replicación sean idénticas. Esto significa:

- Tengan la misma estructura, los mismos nombres de columna y el mismo tipo de dato (incluyendo atributos idénticos) dentro de las columnas correspondientes.

- Todas las líneas emparejadas, si existe una línea en una base de datos, existe en todas las otras.

- Los valores en las líneas correspondientes son idénticos.

Eso es, después de todo, el punto de replicación, tener una réplica en una base de datos de algún objeto o todos los objetos en otra base de datos.

3.4.5. Confiabilidad

Construye declaraciones estándar SQL y cambios post replicados a la base de datos destino. Replica únicamente los cambios hechos en los datos de origen, proveyendo una rápida y fiable solución de replicación. Por defecto, si los cambios están en un *insert*, se usa todo de las columnas en la línea a construir en una declaración *insert*. Si los cambios están en un *delete*, se usa únicamente el valor llave para construir una cláusula *where* que localiza la línea correcta.

Si los cambios son un *update*, se usa una llave única y únicamente los valores de las columnas cambiadas en la cláusula *where*. Antes de mandar esos cambios, el proceso post compara una imagen previa de los valores de las columnas de origen para los valores existentes en la columna destino. Si ellos concuerdan, confirman un estado de sincronización, post aplica los cambios; si no, los cambios son llevados hacia un archivo de error y *SharePlex* regresa un error de "fuera de sincronización". *SharePlex* valida constantemente el dato destino.

3.4.6. Rendimiento

Toda comunicación y movimiento de datos es manejado por sistemas de transporte y mensajes internos de *Shareplex*, usando protocolos asincrónicos con conexiones *TCP/IP* que son muy eficientes para grandes transferencias de datos. Este sistema entrega alto rendimiento, confiabilidad, mientras se usa menos ancho de banda para la comunicación. *Shareplex* puede replicar sobre cualquier red *TCP/IP*, incluyendo entornos *WAN*.

Los cambios son replicados conforme ocurren, más bien sobre un *commit*; *Shareplex* reduce el impacto que tiene la replicación sobre la red. Esto no causa picos en el rendimiento de la red. Haciendo esto mantiene mínima latencia entre los sistemas origen y destino.

Shareplex es diseñado para recuperarse de fallas en la red. Cuando la red no está disponible, los registros son almacenados en una cola del sistema origen. *Shareplex* automáticamente detecta cuándo la red se encuentra disponible nuevamente y transmite los datos.

También puede controlar cuando *SharePlex* usa la red. Ordinariamente, los datos son transferidos del origen al destino continuamente. Sin embargo, sí quiere un límite cuando esto ocurra para enviar datos cuando el costo de conexión de red sea menor; por ejemplo, puede detener el proceso de exportación en el sistema origen. Los datos quedarán seguramente en la cola de exportación mientras inicia la exportación nuevamente.

3.4.7. Disponibilidad

Se puede configurar *SharePlex* para replicar sobre conexiones *LAN* y *WAN* que guarden una segunda instancia constantemente actualizada y lista para tomar la producción de la instancia cuando el sistema origen está programado para mantenimiento o un desastre.

La instancia alternativa es una imagen espejo del entorno de producción. Para optimizar la efectividad de esta alternativa, todos los objetos replicados deberían tener el mismo propietario y nombres sobre el sistema primario y secundario. El periodo físico se refresca usando para actualizar cambios a la estructura de la base de datos, mientras SharePlex replica los cambios a los datos.

Puede configurar replicación bidireccional usando una configuración reversa, mientras el sistema primario está fuera de línea los usuarios están trabajando sobre el sistema secundario, las colas *SharePlex* cambian sobre el sistema secundario. Entonces, cuando el sistema primario es restaurado, *SharePlex* puede actualizarse con esos cambios.

3.4.8. Ventajas

- Mantiene completamente accesible a la instancia destino, no está forzado a elegir entre replicación y acceso, puede acceder a la instancia mientras *SharePlex* está actualizándola.

- Diseñado para entornos *OLTP* de alta intensidad está diseñado para los negocios con volúmenes de datos. Es capaz de replicar millones de transacciones por día para miles de tablas.

- Conserva los recursos del sistema *SharePlex* logra esta replicación sin impactos significativos a la instancia origen, el sistema origen o la red. Este diseño basado en *log* permite replicar con muy baja sobrecarga u *overhead*.

- Velocidad y exactitud con ambas replicación continua, *SharePlex* es rápido, minimiza la latencia entre las instancias origen y el destino capturando las modificaciones a las tablas seleccionadas y secuencias de *redo logs* Oracle continuamente, dándole prioridad a la replicación de la transacción que les ha dado *commit*. Si la transacción es cancelada, *SharePlex* replica el *rollback* tanto que la instancia destino es una representación exacta de la base de datos origen.

- *Shareplex* es rápido, sin embargo, la velocidad no sacrifica la exactitud del dato. Shareplex estrictamente adhiere al modelo consistencia de lectura, manteniendo ambas operaciones en orden y el contexto de la sesión toda la manera al de la instancia designada, donde usa el estándar SQL para aplicar la transacción.

- *SharePlex* mantiene tolerancia a fallos, puede colocar en una cola si la instancia destino está abajo, permitiéndole a las transacciones acumularse en el sistema destino mientras la comunicación puede ser restablecida con la instancia destino. Si el sistema destino en la red está abajo, *SharePlex* almacena las transacciones en el sistema origen mientras las operaciones son restauradas.

- Provee alto nivel de control de usuario con este diseño, se puede tener opciones adicionales; si se prefiere controlar cuando *SharePlex* envíe los datos sobre el enlace de la red, se puede hacer.

- Instalación fácil y rápida *SharePlex*, por todo lo sofisticado y poderoso que es, es relativamente simple de instalar. El promedio de instalación toma alrededor de una hora sobre dos sistemas.

- Replica en un entorno de alta disponibilidad provee una ventaja significativa en este entorno y otras operaciones de misiones críticas, donde el acceso a los datos es crítico, y el *downtime* significa pérdida de oportunidad en los negocios. *Shareplex* permite hacer la sincronización inicial de replicación, como la resincronización después de una falla o cambios en la aplicación-base de datos, para lograr con una mínima interrupción la actividad del usuario. En adición, usando replicación Shareplex con otras tecnologías de alta disponibilidad permite a la base de datos secundaria ser usada para consultas y reportes.

- *Shareplex* entrega esos beneficios por medio de la función *reconcile*, la cual coordina los resultados de una sincronización o resincronización física, como proveer con un *backup* o disco *mirroring* y soluciones refrescadas, con replicación basada en transacciones. Después que la base de datos destino ha sido recuperada, la función *reconcile* compara la copia estática proveída por el *backup* con las transacciones replicadas en las colas de replicación para traer las dos bases de datos dentro de sincronización.

- *SharePlex,* que es una solución liviana en uso de recursos (CPU, ancho de banda de comunicación, memoria, sobre carga moderada a la base de datos, etc.), sin un solo punto de falla y facilidad de manejar ambientes heterogéneos en versiones de base de datos Oracle y sistemas operativos distintos y diferentes versiones del mismo.

3.4.9. Desventajas

- Posiblemente precio; sin embargo, cuando es misión crítica no importa pagar un poco de dinero con tal de tener disponibilidad, rendimiento, etc.

3.5. Solución *SQL Server* 2000 *Cluster*

3.5.1. Definición

Esta solución de la base de datos *SQL Server* 2000 *cluster* con acceso compartido a una sola base de datos, utiliza las ventajas proporcionadas por los *cluster* ya que si un nodo en el *cluster* falla, los nodos restantes continúan dando servicio a los usuarios conectados a los mismos.

3.5.2. Restricciones

- *SQL Server* 2000 *Cluster* solo corre bajo el sistema operativo Microsoft *Windows Server*.

3.5.3. Requerimientos mínimos

El *software* que se necesita para SQL *Server cluster* depende de cuántos nodos tenga el *cluster*, si tiene dos o más que dos.

Para el *cluster* de dos nodos, se necesita lo siguiente:

- Dos licencias para *Microsoft Windows* 2000 *Advanced Server*
- Una licencia para SQL Server 7.0 *Enterprise* o SQL *Server* 2000 *Enterprise* para activo-pasivo, o dos licencias para activo-activo.
- Los últimos *Service Packs* para *Windows* 2000 y SQL *Server*.

Hay que tener en cuenta que siempre se debe tener los últimos *Service packs*, debido al problema de los *bugs* o errores relacionados al *cluster* y que estos hayan sido resueltos por ellos mismos.

Hardware necesario para *cluster*

Se asumirá un *cluster* de dos SQL *Servers*, para ello se necesitara como mínimo lo siguiente:

- Dos servidores con un mínimo de 256 MB de RAM y un solo CPU Pentium III.

- Un arreglo de discos compartido que soporten *RAID* 5, cualquiera de los dos *SCSI* o canal de fibra.

- Cada servidor debe tener al menos un disco duro local *SCSI* y su propia controladora *SCSI*.

- Cada servidor debe tener un adaptador *SCSI* o canal de fibra para comunicarse hacia el arreglo de discos compartidos. El arreglo de discos compartidos no puede ser usado por la controladora *SCSI* usada para el disco duro local o el CD-ROM.

- Cada servidor debe tener 2 tarjetas de red PCI (una para conexión privada y la otra para la conexión pública).

- Idealmente, ambos servidores deben ser idénticos en *hardware*, *drivers* o controladores, *software*, y configuración. La igualdad física de un servidor a otro, evitará que se tenga mayores problemas.

- Otra consideración muy importante es cuando se seleccione el *hardware* del *cluster* y es que éste debe ser aprobado por la lista de compatibilidad de hardware de Microsoft como un sistema soportado. Esto significa que antes que Microsoft soporte el *cluster*, todo el *hardware* del *cluster* seleccionado (servidores, tarjetas, arreglo de discos, etc.) debe haber sido probado como un sistema completo y aprobado por Microsoft. Si el sistema que se compraría no ha sido probado, entonces Microsoft no podrá ayudar incluso si se les llamara o se les pagara por soporte.

3.5.4. Funcionamiento

En un *cluster* de dos nodos, uno de SQL Server es referido como el nodo primario y el segundo como el nodo secundario. En un diseño de *cluster* activo-pasivo, SQL Server correrá sobre el nodo primario, y cuando el nodo primario falle, entonces el secundario entrará en funcionamiento.

Cuando se construye un *cluster* de dos nodos usando Windows 2000 *Advanced Server* y el servicio Microsoft *Clustering*, cada nodo debe estar conectado a un arreglo de discos compartidos usando cables *SCSI* o un canal de fibra.

Usualmente, este arreglo de discos compartido usa arreglos de discos con nivel RAID 5. Todos los datos compartidos en el *cluster* deben ser almacenados en este arreglo de discos, de tal manera que cuando ocurre una falla, el nodo secundario en el cluster no puede accesarlo.

Como se ha dicho, el *cluster* no ayuda para la protección de los datos o el arreglo de discos en los que esté almacenada la información. Por ello, es muy importante que se seleccione un arreglo de discos compartido que sea confiable e incluya tolerancia a fallos.

Ambos servidores deben estar conectados a un arreglo de discos compartido y deben estar conectados por medio de una red privada, que es usada para cada nodo para mantener el estado del otro nodo. Por ejemplo, si el nodo primario experimenta fallas de *hardware*, el nodo secundario detectara esta falla y automáticamente iniciará un *failover*.

¿Cómo se dan cuenta los clientes que están usando SQL Server que ha ocurrido una falla en el *cluster*? Esto es parte del Microsoft *Cluster Service*. Esencialmente SQL Server tiene asignada su propio nombre y dirección virtual *TCP/IP*. Este nombre y dirección son compartidas por ambos servidores en el *cluster*.

Usualmente, un cliente se conectara a SQL *Server cluster* usando el nombre virtual usado por el *cluster*. Y esto va mas allá de lo que al cliente le concierne, ya que hay un único SQL *Server* físicamente, no dos. Asumiendo que el nodo primario de SQL Server *cluster* es el nodo que está corriendo SQL *Server* sobre un diseño de *cluster* activo-pasivo, entonces el nodo primario responderá a las solicitudes de los clientes. Si el primario falla, entonces ocurrirá un *failover* hacia el nodo secundario, el *cluster* conservará el mismo nombre virtual de SQL *Server* y la misma dirección *TCP/IP*, aunque ahora un nuevo servidor físico responderá a las solicitudes de los clientes.

Durante el periodo de *failover*, el cual puede ser varios minutos porque la cantidad de tiempo depende del número y tamaño de la base de datos SQL Server, y de cómo fueron activados, los clientes no podrán acceder a SQL Server, entonces hay una pequeña cantidad de tiempo fuera de servicio cuando ocurre un *failover*.

Cómo reacciona el cliente ante el proceso de *failover*. Algún *software* solo esperará a que ser recupere del *failover*, cuando éste se complete, continuará como si nada hubiera sucedido. Algún *software* presentará una ventana con un mensaje en pantalla, describiendo pérdida de conexión. Otro *software* cliente no sabrá qué hacer, por lo que los usuarios tendrán que salir y recargar a los clientes antes que puedan acceder nuevamente a SQL *Server*.

Como parte de este proceso de prueba, cuando se implementa un SQL Server *cluster*, es importante encontrar las reacciones de los clientes del *software* que se conectan a SQL Server cuando ocurre un *failover*. De esta manera, se puede informar a los usuarios que tienen que esperar, con lo cual habrá una mejor comunicación cuando esto ocurra.

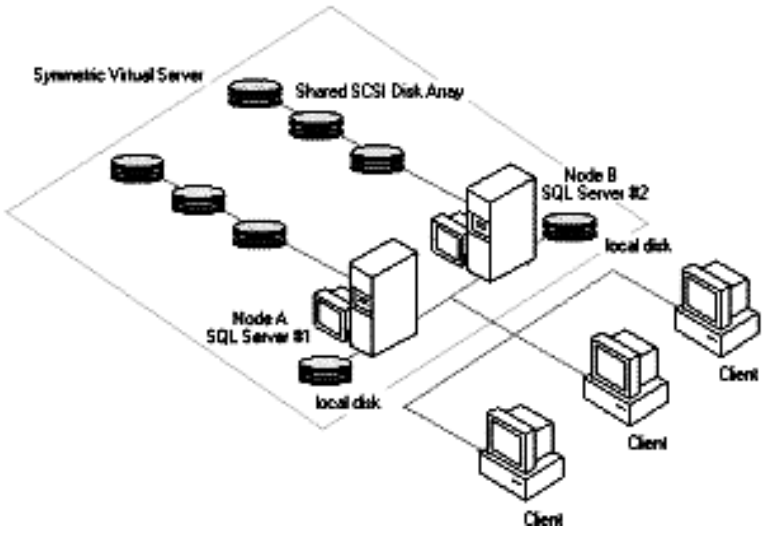
Una vez que ocurre un *failover*, se querrá encontrar que lo causó para tomar la acción necesaria y corregir el problema. Una vez que el problema ha sido resuelto, el siguiente paso es *failover* SQL *Server back* hacia el nodo primario del nodo secundario. Esto se puede programar en cualquier tiempo, de preferencia cuando la actividad de usuarios sobre el sistema es baja.

3.5.5. Disponibilidad

SQL *Server Enterprise Edition* ofrece tolerancia a fallas y alta disponibilidad proporcionando *failover* de un servidor a otro cuando el primero falla o se toma fuera de servicio para darle mantenimiento. El mecanismo de failover funciona sobre dos servidores configurados como un *cluster* y esos dos servidores pueden soportar dos instancias de SQL *Server*. Cada servidor administra una partición de la base de datos. La base de datos SQL *Server* es colocada sobre discos *SCSI* compartidos los cuales son accesibles para ambos servidores. Si uno falla, el otro toma los discos y reinicia el servidor fallado como el nodo sobreviviente. El servidor reiniciado recupera la base de datos y comienza aceptando las conexiones de los clientes. Los clientes se reconectan al servidor virtual cuando el servidor primario falla. Una función de Windows 2000 Server, Microsoft *cluster Service*, permite que el nombre del servidor virtual y la dirección IP migren entre los nodos, entonces los clientes no sabrán que el servidor ha sido movido. Este servicio de Microsoft *cluster* está disponible en *Windows* 2000 *Server*.

SQL *Server Enterprise Edition* proporciona soporte para servidores virtuales. Una vez configurado, el servidor virtual se mira como otro servidor en la red, excepto que puede tolerar fallas en el *hardware*.

Figura 24. Configuración de SQL *Server* 2000 *cluster*



Fuente: http://www.microsoft.com/technet/prodtechnol/sql/2000/plan/default.mspx

SQL *Server failover* es completamente automático. La detección y recuperación de la falla tomará únicamente pocos minutos. Cuando el nodo fallido es reparado, es reiniciado y comienza como un nuevo servidor de *backup*. Si se desea, el nodo fallido puede regresar a ser el servidor original cuando sea reparado.

3.5.6. Crecimiento

SQL *Server* proporciona crecimiento, debido a que se basa en la arquitectura de *clusters* y como bien se conoce con esta arquitectura se pueden agregar nodos, ya sea para propósitos de reemplazo en el caso de falla de algún nodo como también se pueden agregar procesadores al nodo lo cual implicaría una escalabilidad vertical en el nodo.

3.5.7. Ventajas

- Permite una respuesta automática a una falla en el servidor o *software*. No se requiere de intervención humana.

- Permite realizar actualizaciones sin forzar a los usuarios a salirse del sistema por periodos extendidos de tiempo.

- Permite reducir el tiempo fuera de servicio debido a rutinas de servidor, red o mantenimiento de base de datos.

- El *cluster* no requiere que cualquier servidor sea renombrado. Entonces cuando ocurre un *failover*, esto es relativamente transparente al usuario final.

- *Failing back* es rápido, y puede ser realizado considerando que el servidor primario está completamente reparado y colocado nuevamente en línea.

3.5.8. Desventajas

- Más caro que otras alternativas, tales como servidores *standby*.

- Requiere más tiempo que otras alternativas.

- Requiere más mantenimiento que otras alternativas.

- Requiere más *hardware* especializado que sea compatible con *clusters*.

- No puede tolerar una falla en el almacenamiento compartido.

- Requiere más experiencia por parte de administradores de base de datos y administradores de red.

- Requiere dos nodos que estén físicamente conectados, por ello los nodos no pueden estar en ubicaciones físicas diferentes. Eso es, en caso de un desastre, se pierden ambos nodos.

- *Cluster* es una solución de alta disponibilidad, pero no es efectivo contra recuperación de desastres, a menos que esté combinado con otros acercamientos como *backup*/*restore*, discos *mirroring*, etc.

4. COMPARACIÓN DE SOLUCIONES DE ALTA DISPONIBILIDAD DE BASE DE DATOS POR *HARDWARE* O POR *SOFTWARE*

4.1. Comparación de ventajas y desventajas de soluciones de alta disponibilidad de base de datos por *hardware* y por *software*

Al llegar a este capítulo, en el cual se mostrará un cuadro comparativo sobre lo más relevante de cada una de las soluciones descritas anteriormente, siempre en el entorno de alta disponibilidad de bases de datos por *hardware* y alta disponibilidad de bases de datos por *software*, se podrá apreciar que no todas las soluciones son exactamente adecuadas para un negocio en particular. Esto depende de los requerimientos de cada negocio y por ello no se puede decir que tal solución es óptima para una organización; por otro, lado las organizaciones deberán tener presupuestada la cantidad de dinero para invertir en una solución en particular o una combinación de unas con otras, todo con el único propósito de reducir ese tiempo fuera de servicio que a ninguna entidad le gustaría experimentar en la actualidad, en la que la mayoría de negocios gira en torno a la tecnología de información. Los clientes no están dispuestos a perder acceso a la información tan vital de los negocios ni mucho menos a la pérdida total de los datos en el caso de un desastre, ello conlleva que se tendrá que hacer una mayor inversión al hacer cualquier tipo de combinación de soluciones, cuyo único fin, es el de alcanzar ese porcentaje de cinco nueves, con el respaldo de que su información y que sus clientes no experimenten ningún tipo de fallo o destrucción total de la localidad en cuanto al almacenamiento y manejo de la información se refiera.

Por lo tanto, no se puede concluir cuál es la mejor solución y es por ello que a continuación se hará un análisis de sus características más relevantes.

Tabla VII. Comparación de las características mas relevantes de las soluciones mencionadas

	Oracle *Data Guard*	Oracle *Real Application Clusters*	*Shareplex* para Oracle	SQL *Server* 2000 *Cluster*	RAID 0	RAID 1	RAID 5	RAID 0 + 1
Mecanismo de *Failover*	*Switchover* y *Failover*	*Transparent Application Failover*	*Failover* Manual o Automático[1]	*Failover* Automático	Automá-tico	Automá-tico	Auto-mático	Automático
Recuperación a Desastre	Si	Si	Si	No	No	No	No	No
Escalabilidad	No	Si	Si[2]	Si	Si[3]	Si	Si	Si
Disponibilidad	Si	Si	Si	Si	No	Si	Si	Si
Confiabilidad	Si	Si	Si	Si	Si	Si	Si	Si
Versión de la base de datos	Igual	Igual	Igual o Diferente	Igual	No Aplica	No aplica	No aplica	No aplica
Ubicación de la base de datos físicamente distinta entre el origen y destino	Local o Remota	Local	Local o Remota	Local	Local	Local	Local	Local
Costo[4]	Bajo a Regular	Regular a Alto	Regular a Alto	Regular	Bajo	Bajo	Bajo	Bajo

[1] *SharePlex* conjuntamente con *hardware* y *software* de terceros, puede tener *Failover* Automático, balance de carga completa.

[2] *SharePlex* si ofrece Escalabilidad ya que puede replicarse a mas de un servidor, aumentando la capacidad de procesamiento.

[3] Si, pero es limitado al chásis del arreglo de discos. Igual a los demás niveles de RAID.

[4] La fila de costo (aproximado solo la base de datos), esta se estaría clasificando entre los siguientes rangos, equivalentes a Bajo <= $40,000, Regular = $80,000, Alto = > $120,000.

Por la parte de soluciones basadas en *hardware* se debe tomar en cuenta que para éstas es mejor utilizar controladoras redundantes, las cuales pueden costar entre $1,000 y $2,500 por controladora, a esto hay que sumarle los discos pero estos permanecen adjuntos al servidor. Por otro lado, si se obtiene un arreglo pequeño externo al servidor éste puede ir alrededor de los $8,000 y aun mayor a esa cantidad, sin incluir los discos.

El rendimiento de las soluciones de alta disponibilidad de base de datos no fue incluido en la tabla anterior ya que depende de una serie de factores que están más allá del alcance de la base de datos y los cuales dependen no solo de la base de datos, sino a eso hay que agregarle el rendimiento del sistema operativo, del servidor y cada uno de sus componentes internos, el almacenamiento ya sea compartido o no, las conexiones de red, etc.

La tabla que se muestra a continuación presenta los precios de las bases de datos relacionales Oracle 9i *Enterprise Edition* y SQL *Server* 2000 por procesador.

Tabla VIII. Precios de lista de las bases de datos relacionales utilizadas en el presente trabajo

Número de CPU	Oracle 9i *Enterprise Edition*	SQL *Server* 2000 *Enterprise Edition*
1	$40,000	$19,999
2	$80,000	$39,998
4	$160,000	$79,996
8	$320,000	$159,992
16	$640,000	$319,984
32	$1,280,000	$639,968

Ésta no es una comparación total de precios entre SQL *Server* 2000 y Oracle 9i. Es sólo una comparación corta de sus precios.

Se pueden tener descuentos y los precios pueden ser incrementados o decrementados en el futuro. Para obtener más información es necesario ver en los sitios respectivos de las casas proveedoras.

Para ver el costo de esas soluciones, en este trabajo solo se tomará en cuenta el valor de la licencia de la base de datos, la cual puede ser por medio de procesador o por el número de usuarios que usarán la base de datos; por lo tanto, planteare unos escenarios en los cuales se podrá determinar el costo aproximado sobre la base de datos, mas no exacto ya que a esto se debe sumar todos esos componentes redundantes, sistema operativo y el almacenamiento que son de vital importancia.

Por ejemplo, el primer escenario al querer implementar una solución basada en *clusters* con dos servidores (nodos) y cada nodo con dos procesadores y acceso a una base de datos compartida, al implementarla por medio de la base de datos SQL *Server* 2000, si se toma la licencia por procesador está casi en los $19,999 pero como son cuatro procesadores eso quiere decir que se multiplica esa cantidad por 4 con lo cual esta solución costaría $79,996 por año en un ambiente basado en *Windows*. Cabe resaltar que éstas cantidades son solo para la base de datos relacional y a esto hay que agregarle el costo de los servidores, más *UPS* redundantes, más almacenamiento, etc., esto no lo tomo en consideración en el presente trabajo.

Por otro lado, si en el mismo escenario se implementa la misma solución solo que utilizando la base de datos Oracle 9i, el valor por procesador es de $40,000.00 por año y debido a que son cuatro procesadores, eso equivale a la suma de $160,000.00; sólo el valor de la licencia por procesador con el cual no hay límite a el número de usuarios que accedan a esa base de datos compartida.

Otro escenario sería implementar una solución de bases de datos basada en *software* que estén físicamente separadas y se haría bajo la base de datos Oracle 9i, ya que Oracle cobra por cada base de datos no importando si está activa o en espera. Con una base de datos primaria y una base de datos *standby* el precio sería de $80,000 si se tomara la licencia por procesador durante el primer año para esas dos bases de datos; para proporcionar recuperación a desastre las mismas deberían estar físicamente localizadas en lugares distantes.

Es importante aclarar que los costos de los escenarios anteriormente descritos corresponden solo al primer año y al culminar ese año se debe pagar el 20% de la cantidad, lo cual disminuye considerablemente pero esto puede ser variable y aplica en la base de datos Oracle.

Con esta información se puede tener una idea de costo aproximado y las características más importantes para cada una de las soluciones expuestas en el presente trabajo; siempre recordando que la alta disponibilidad implica redundancia respecto a todos los componentes que puedan hacer que ésta falle o sea destruida en casos extremos, pero reales.

CONCLUSIONES

1. La clave de la alta disponibilidad de base de datos es la redundancia que permite mantener los datos en más de un lugar, con lo cual se logra en un momento dado la recuperación a un desastre.

2. El tiempo fuera de servicio o *downtime* es categorizado como planeado y no planeado; indistintamente cual de los dos ocurra, debe ser minimizado para evitar cualquier rompimiento en el servicio, lo que implicaría pérdidas.

3. Los *clusters* son usados para ocultar las fallas en los nodos del sistema, sin dejar de prestar el servicio a los usuarios de las bases de datos y permiten distribuir la carga de trabajo, proporcionando un buen rendimiento en cualquier momento; además, permiten agregar nodos, lo cual los hace sobresalir en sus principales ventajas que son la disponibilidad y la escalabilidad.

4. La replicación es una solución de *software* que permite tener copias de la base de datos en producción en más de un lugar, ya sea una ubicación local o remota; estas copias actualizadas de las bases de datos de producción pueden ser usadas para reportes, consultas, *backups* y en casos de emergencia cuando el sistema de producción haya fallado o esté sea totalmente destruido.

5. Las soluciones de alta disponibilidad de bases de datos por *hardware* o por *software*, al combinarlas proporcionan un mejor resultado.

6. La solución de alta disponibilidad de base de datos por *software* Oracle *Data Guard* proporciona recuperación a desastres, si la base de datos está en una localidad remota; proporciona disponibilidad si se produce una falla en la base de datos primaria, o bien cuando se necesite operaciones de mantenimiento, permitiendo cambiar entre bases de datos primaria y *standby*.

7. La solución de alta disponibilidad de base de datos por *software* Oracle *Real Application Clusters* proporciona disponibilidad y escalabilidad debido a que está basadá en *clusters*; proporciona recuperación a desastres si los nodos de la misma se encuentran a gran distancia entre ellos.

8. La solución de alta disponibilidad de base de datos por *software Shareplex* para Oracle proporciona recuperación a desastres, manteniendo una base de datos en un lugar remoto constantemente actualizado por medio de la replicación, proporciona escalabilidad y disponibilidad ya que puede replicarse a más de un servidor.

9. La solución de alta disponibilidad de base de datos por *software* SQL *Server* 2000 *cluster* proporciona disponibilidad y escalabilidad, debido a que usa tecnología de *clusters* y proporciona un buen rendimiento. No proporciona recuperación a desastres, ya que los nodos residen en el mismo lugar.

10. La solución de alta disponibilidad de base de datos por *hardware* llamada *RAID* local, proporciona disponibilidad a través de distintos niveles de redundancia en los arreglos de discos, con lo cual se evita la pérdida de los datos si uno o más discos llegan a fallar; proporciona escalabilidad, la cual está limitada al chásis del arreglo de discos pero no proporciona recuperación a un desastre local.

11. La opción de almacenamiento *SAN* proporciona recuperación a desastre, si los sitios están copiados idénticamente y localizados a una distancia segura, debido a su tecnología de canal de fibra para proteger y distribuir datos; ofrece escalabilidad agregando almacenamiento, sin tener tiempo fuera de servicio; reduce el tráfico en la red primaria.

12. La opción de almacenamiento *NAS* comparte archivos a través de una red de área local; mejora el rendimiento, descargando el servicio de archivo proporcionado por el servidor.

RECOMENDACIONES

1. Antes de implementar una solución se necesita determinar si satisface las necesidades de un negocio tales como la continuidad del servicio, costo, requerimientos, etc.

2. Se recomienda ver el modo de las configuraciones disponibles para cada una de las diversas soluciones mencionadas.

3. Es importante resaltar que la redundancia es un punto clave en la implementación de las soluciones de alta disponibilidad de base de datos, por lo cual se debe tener en cuenta que la base de datos son solo un componente para alcanzar esa disponibilidad de los sistemas; por lo tanto, es necesario consultar otras fuentes sobre los demás componentes como las fuentes de energía redundantes, sistemas de ventilación de los ambientes, etc., para lograr la alta disponibilidad.

BIBLIOGRAFÍA

1. *Adding Instances and Nodes*.
   http://technet.oracle.com/doc/windows/server.804/a55925/chap8.htm
   Estados Unidos: 06/08/2002

2. *Administering múltiple Instances.*
   http://technet.oracle.com/doc/windows/server.804/a55925/chap7.htm
   Estados Unidos: 06/08/2002

3. Alta disponibilidad.
   http://www.logica.cl /soluciones_de_integración_TI.htm
   Chile: 16/02/2002

4. Alta disponibilidad.
   http://www.enlace.cl/empresa/anexos/alta_disponibilidad.pdf
   Chile: 16/02/2002

5. *Backing Up and Recovering an Oracle Parallel Server Database*.
   http://technet.oracle.com/doc/windows/server.804/a55925/chap9.htm
   Estados Unidos: 06/08/2002

6. *Backup* de recuperación.
   http://otn.oracle.com/deploy/availability/pdf/backup_recovery_twp
   Estados Unidos: 06/08/2002

7. *Basic RAID Organizations*.
   http://www.ecs.umass.edu/ece/koren/architecture/Raid/raidhome
   Estados Unidos: 02/06/2005

8. *Before you begin*.
   http://technet.oracle.com/doc/windows/server.804/a55925/preface
   Estados Unidos: 06/08/2002

9. *Configuring Oracle Parallel Server.*
http://technet.oracle.com/doc/windows/server.804/a55925/chap5.htm
Estados Unidos: 06/08/2002

10. *Costs and Benefits for High Availability Oracle9i.*
http://www.dba-oracle.com/art_dbazine_high_avail.htm
Estados Unidos: 16/08/2004

11. *Glossary*.
http://technet.oracle.com/doc/windows/server.804/a55925/glos.htm
Estados Unidos: 06/08/2002

12. *High Availability Oracle 8i listing*.
http://otn.oracle.com/deploy/availability/htdocs/ha8i_listing.html
Estados Unidos: 02/08/2002

13. *High Availability Solutions*.
http://www.pafumi.net/replication/High_Availability_Solutions.html
Estados Unidos: 29/05/2005

14. *High Availability*.
http://www.tpc.org/information/other/articles/ha.asp
Estados unidos: 16/02/2002

15. *High Available databases*.
http://www.firstsql.com/highavailability.html
Estados Unidos: 11/05/2005

16. *Installing and configuring Oracle Parallel Server Manager.*
http://technet.oracle.com/doc/windows/server.804/a55925/chap6.htm
Estados Unidos: 06/08/2002

17. *Installing Oracle Parallel Server.*
http://technet.oracle.com/doc/windows/server.804/a55925/chap4.htm
Estados Unidos: 06/08/2002

18. Introducción a Oracle 8i.
http://technet.oracle.com/doc/oracle8i_816/paraserv.816/a76968/psintro.htm Estados Unidos: 02/08/2002

19. *Introduction to OPS*.
http://technet.oracle.com/doc/windows/server.804/a55925/chap1.htm
Estados Unidos: 06/08/2002

20. *Introduction to storage area network*.
http://is.pennnet.com/articles/introsan.cfm
Estados Unidos: 20/05/2005

21. *NAS and SAN cost considerations*.
http://is.pennnet.com/articles/nassancost.cfm
Estados Unidos: 20/05/2005

22. *NAS* y *SAN similarities and differences*.
http://is.pennnet.com/articles/nassan.cfm
Estados Unidos: 20/05/2005

23. *No data loss standby*.
http://otn.oracle.com/deploy/availability/pdf/no_data_loss_sb.pdf
Estados Unidos: 02/08/2002

24. Oracle 8i *High Availability*.
http://www.jeffreyhazelwood.com/Oracle8iHA.pdf
Estados Unidos: 11/05/2005

25. Oracle 9i *deployment and performance.*
http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96598.pdf
Estados unidos: 26/10/2004

26. Oracle 9i *Instalation.*
http://download-east.oracle.com/docs/pdf/A95493_01.pdf
Estados Unidos: 26/10/2004

27. *Oracle Parallel Server Architecture*.
http://technet.oracle.com/doc/windows/server.804/a55925/chap3.htm
Estados Unidos: 06/08/2002

28. *Oracle Prices*.
http://www.oracle.com/corporate/pricing/pricelists.html
Estados Unidos: 02/06/2005

29. Oracle9i *Database Concepts*.
http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96524.pdf
Estados Unidos: 26/10/2004

30. *Parallel Fail Safe*.
http://otn.oracle.com/deploy/availability/pdf/PFS_TWP_53.pdf
Estados Unidos: 02/08/2002

31. *Parallel Hardware Architecture*.
http://technet.oracle.com/doc/windows/server.804/a55925/chap2.htm
Estados Unidos: 06/08/2002

32. *Performing Oracle and SQL Server database recovery*.
http://is.pennnet.com/articles/databaseperf.cfm
Estados Unidos: 20/05/2005

33. *Pre-Installation Tasks for Installing RAC on hp Tru64 UNIX-Based Systems*.
http://download-east.oracle.com/docs/html/B10766_05/pretru64.htm#sthref1641
Estados Unidos: 30/09/2004

34. ¿Qué es *Raid*?
http://www.raidweb.com/whatis.html
Estados Unidos: 30/05/2005

35. *Raid Definitions*.
http://linux.cudeso.be/raid.php
Estados Unidos: 29/05/2005

36. *RAID White Paper*.
http://www.attotech.com/diamond/pdf/RAIDWhitePaper.pdf
Estados Unidos: 02/06/2005

37. *RAID*.
http://rk.8m.com/cgi-bin/i/PIC/raid.htm
Estados Unidos:02/06/2005

38. *Read only standby database.*
http://www.dbatoolbox.com/WP2001/hastandby/Read-only_standby_database.pdf
Estados Unidos: 19/08/2002

39. *Real Application Clusters Concepts*.
http://download-east.oracle.com/docs/cd/B10501_01/rac.920/a96597.pdf
Estados Unidos: 26/10/2004

40. Replicado de datos en tiempo real bajo Linux.
http://www.linuxfocus.orgt/castellano/march2001/article199.htm
Estados Unidos: 16/02/2002

41. *SAN vs. NAS.*
http://www.infrastor.com/downloads/papers/sanvsnas.pdf
Estados Unidos: 19/08/2002

42. *SAN y NAS*.
http://www.nas-san.com/differ.html
Estados Unidos: 19/08/2002

43. *Shareplex for Oracle*.
http://www.quest.com/SharePlex
Estados Unidos: 16/02/2002

44. Sistemas de alta disponibilidad bajo linux.
http://www.linuxfocus.orgt/castellano/november2000/article179.htm
Estados Unidos: 16/02/2002

45. Sistemas de alta disponibilidad.
http://www.optimat.com/soluciones/altadisponibilidad.html
España: 16/02/2002

46. SQL *Server* 2000 *clustering Intro* 2.
http://www.sql-server-performance.com/clustering_intro2.asp
Estados Unidos: 29/05/2005

47. SQL *Server* 2000 *clustering Intro*.
http://www.microsoft.com/technet/prodtechnol/sql/2000/plan/ssmsam.mspx Estados Unidos: 29/05/2005

48. SQL *Server product overview*.
http://databases.about.com/od/sqlserver/a/sqlserver2k.htm
Estados Unidos: 30/05/2005

49. SQL *Server versus Oracle* 9i.
http://www.mssqlcity.com/Articles/Compare/sql_server_vs_oracle.htm Estados Unidos: 10/06/2005

50. SQL *Server*.
http://www.microsoft.com/sql/evaluation/overview/default.mspx
Estados Unidos: 30/05/2005

51. *Standby Concepts*.
http://download-east.oracle.com/docs/cd/A87860_01/doc/server.817/a76995.pdf
Estados Unidos: 26/10/2004

52. *Standby Database Concepts*
http://fncduh1.fnal.gov/oracledoc/v8.1.7/DOC/server.817/a76995/standbyc.htm#25080 Estados Unidos: 19/08/2002

53. *Understanding the Real Application Clusters Installed Configuration*
http://download-east.oracle.com/docs/html/B10766_05/undrstnd.htm
Estados Unidos: 30/09/2004

54. *White paper Building* 24 x 7 *database*.
http://www.quest.com/building_wp.pdf
Estados unidos: 16/02/2002