UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

CONSIDERACIONES PARA EL AFINAMIENTO Y MONITOREO DE BASES DE DATOS RELACIONALES

TESIS

Presentada a la Junta Directiva de la
Facultad de Ingeniería

POR

SERGIO RODOLFO ALONZO LEMUS

Al conferírsele el título de

INGENIERO EN CIENCIAS Y SISTEMAS

Guatemala, Agosto de 1.999

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de tesis titulado:

CONSIDERACIONES PARA EL AFINAMIENTO Y MONITOREO DE BASES DE DATOS RELACIONALES.

Tema que fuera asignado por la Coordinación de Ingeniería en Ciencias y Sistemas, con fecha 15 de mayo de 1,996.

Sergio Rodolfo Alonzo Lemus

Guatemala 20 de Abril de 1998

Ingeniero
Jorge Luis Alvarez Mejía
Coordinador
Carrera de Ingeniería en Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero:

Por medio de la presento, me permito informarle que he procedido a revisar el trabajo de tesis titulado: "**Consideraciones para el afinamiento y monitoreo de Bases de Datos Relacionales**", elaborado por el estudiante Sergio Rodolfo Alonzo Lemus, a mi juicio, el mismo cumple con los objetivos propuestos para su desarrollo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme

Atentamente,

Ing. Carlos Alfredo Azurdia
Revisor

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Escuelas de Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, Técnica y Regional de Post-grado de Ingeniería Sanitaria.

Ciudad Universitaria, zona 12
Guatemala, Centroamérica

Guatemala,
3 de febrero de
1,999
REF.:

Ingeniero
Herbert René Miranda Barrios
Decano, Facultad de Ingeniería

Señor Decano:

Atentamente me dirijo a usted, para informarle que después de conocer el dictamen del Asesor del trabajo de tesis del estudiante SERGIO RODOLFO ALONZO LEMUS, titulado CONSIDERACIONES PARA EL AFINAMIENTO Y MONITOREO DE BASES DE DATOS RELACIONALES, procedo a la autorización del mismo.

Atentamente,

"ID Y ENSEÑAD A TODOS"

Ing. Jorge Luis Alvarez Mejía
COORDINADOR
INGENIERIA EN CIENCIAS Y SISTEMAS

JLAM/edj

c.c. Archivo

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA
DECANATO

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Coordinador de la Carrera de Ingeniería en Ciencias y Sistemas, al trabajo de tesis titulado: CONSIDERACIONES PARA EL AFINAMIENTO Y MONITOREO DE BASES DE DATOS RELACIONALES, presentado por el estudiante universitario SERGIO RODOLFO ALONZO LEMUS, procede a la autorización para la impresión de la misma.

IMPRÍMASE:

Ing. Herbert René Miranda Barrios
DECANO

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
DECANO

Guatemala, 12 de febrero de 1999

ESTA TESIS DEDICO A:

- **DIOS.** Por sus infinita ayuda, que sin él no estaría logrando este título.
- **Mi padre y abuelíta.** Por su constante apoyo y consejo.
- **Mi esposa Betty.** Por su amor y dedicación
- **Mis hermanos**: Carlos, Estuardo y Evelyn por su cariño y apoyo

- **Cooperadores Salesianos.** Por su amistad incondicional y ayuda.
- A mis amigos Raúl Vásquez, Francisco Romero, Fernando Mazariegos, Christianel Sandoval,
  a Datum, S.A. en especial a mis compañeros de soporte técnico.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

"Batch program"

Programa por lotes. Programa no interactivo (no conversacional) como un listado o clasificación de informes.

"Batch proccesing"

Procesamiento por lotes. Procesar un grupo de transacciones a la vez. Las transacciones se reúnen y se procesan frente a los archivos maestros (archivos maestros actualizados) al final del día o en algún otro período. Nótese la diferencia con "transaction proccesing".

Por lo general, los sistemas de información utilizan tanto los métodos de procesamiento por lotes como de procesamiento por transacciones. Por ejemplo, en un sistema de procesamiento de pedidos, el procesamiento por transacciones es la actualización continua de los archivos de clientes y de inventario a medida que se ingresan los pedidos.

Al final del día, los programas de procesamiento por lotes generan listas de despacho para la bodega.

Concluido cierto periodo de tiempo, los programas por lotes imprimen facturas e informes gerenciales.

BBS

(Bulletin Board System) Sistema de tablero de anuncios o de boletines. Sistema de computación que se utiliza como fuente de información y sistema de mensajes para un grupo de interés particulas. Los usuarios se comunicas vía línea telefónica con el BBS, revisan y dejan mensajes para otros usuarios, así como se comunican con otros usuarios del sistema al mismo tiempo. Los BBS se utilizan para distribuir shareware y pueden proveer acceso (puertas) a otros programas de aplicación.

Bind

1. Asignar una dirección de máquina a una referencia lógica o simbólica, o a una dirección.
2. Asignar un tipo o valor a una variable o parámetro..
3. Atar puede utilizarse en vez de términos como vincular o interfaz cuando se hace referencia a un software que está diseñado para comunicarse con otro software o hardware.

"Buffers, database

buffers ó buffers cache"

Memoria intermedia, almacenamiento intermedio, regulador. Segmento reservado de memoria que se utiliza para almacenar datos mientras se procesan. En un programa, los buffers se crean para contener cierta cantidad de datos de cada uno de los archivos que van a ser leídos o grabados. Un buffer puede ser también un pequeño banco de memoria de hardware usado para fines especiales. En un manejador de bases de datos, tiene la misma función, y específicamente la información almacenada en estos buffers, son datos almacenados en la base de datos que son almacenados para alguna modificación en dichos buffers, y que posteriormente serán grabados a disco.

"Bulletin Board"

Tablero de anuncios o de boletines. Ver BBS.

Cilindro

Es un conjunto de tracks. La cabeza de lectura se mueve entre cilindros.

"Commit"

Operación de grabación. Permite grabar a los archivos de datos las transacciones realizadas por las aplicaciones.

Costo

Medida financiera incremental y/o costo de recursos humanos para mejorar el rendimiento del sistema. El costo es medido desde el punto de vista en que la organización use el sistema.

Cursor

En el ambiente de las bases de datos, es una estructura temporal que, de acuerdo a ciertas sentencias, puede ser definido, abierto, cerrado, y utilizado para extraer datos de una tabla (tiene como base una instrucción SQL select); consta de un apuntador que indica la tupla en que se esta desplazando en la tabla (de acuerdo a la selección). En algunos manejadores de bases de datos, dicho apuntador puede navegar hacia atrás o hacia delante.

Daemon

Demonio. Programa UNIX que en un segundo plano se encuentra listo para realizar una operación cuando se requiera. Por lo general es un proceso no observado que se inicia desde el comienzo. Ejemplos de demonios típicos son los integradores de impresión ("print spoolers") y manipuladores de correo electrónico o un planificador que inicia otro proceso en un determinado momento. El término se deriva de la mitología griega y significa "espíritu guardián". Equivale a un agente.

DBA (Data Base Administrator)

Administrador de la base de datos. Persona que se encarga en general de la buena administración de un sistema de bases de datos, que examina su buen funcionamiento, que toma políticas para mejorar su rendimiento, y que afina dicho software.

Deadlock

Punto muerto, interbloqueo. Véase "deadly embrace".

"Deadly embrace"

Abrazo mortal, bloqueo. Estancamiento que ocurre cuando dos elementos en un proceso están, cada uno, esperando que su similar responda. Por ejemplo, si en una red un usuario que trabaja en el archivo A necesita el archivo B para continuar, pero al mismo tiempo otro usuario que trabaja en el archivo B necesita el archivo A para continuar, cada uno espera al otro. Ambos quedan temporalmente bloqueados, el software debe ser capaz de ocuparse del problema.

"Default"

(Por) omisión, defecto. Acción o fijación actual tomada por el hardware o software si el usuario no lo ha especificado de otra manera.

Disco

Es una colección de platos circulares colocados uno encima del otro, rotando alrededor de un eje común. Cada plato, a excepción del mas superior y el mas inferior, tiene dos superficies de lectura y escritura. Cada plato tiene asociado una cabeza de lectura.

Disk Contention

(Contención de disco)

La contención de disco ocurre cuando múltiples procesos tratan de accesar al mismo disco simultáneamente. La mayoría de discos tienen límites en el número de accesos a disco, y en la cantidad de datos que estos pueden transferir por segundo. Cuando estos limites son sobrepasados, el procesamiento generalmente tiene esperas para accesar el disco.

Disk Mirroring

Doble escritura en discos. Grabación de datos redundatentes para operación tolerante a fallas. Los datos se graban en dos particiones del mismo disco, o en dos discos separados dentro del mismo sistema o en dos sistemas informáticos separados.

DSS

1. (Decision Support System) Sistema de apoyo de decisiones. Sistema de información y planeación que proporciona la capacidad de interrogar computadores sobre una base ad hoc, analizar información y predecir el impacto de las decisiones antes de que sean tomadas.

   Los DBMS permiten a los usuarios seleccionar datos y obtener información para elaboración de informes y análisis. Las hojas de cálculo y los programas de modelación proveen tanto análisis como planeación tipo "¿Qué pasaría si?". Sin embargo, cualquier aplicación que apoya la toma de decisiones no es un DSS. Un DSS es un conjunto integrado y sólido de programas que comparten datos e información. Un DSS también puede recuperar datos de la industria obtenidos de fuentes externas que pueden compararse y utilizarse para fines históricos y estadísticos.

   Un DDS integrado afectará directamente el proceso de toma de decisiones de la administración y puede ser una aplicación de computación muy beneficiosa en cuanto a costos. Véase EIS.

2. (Digital Signature Standard) Estándar de firma digital. Estándar de la National Security Administration para autenticar un mensaje electrónico.

"Downsize"

Reducción de tamaño. Convertir sistemas de mainframe y basados en minicomputadores a LAN de computadores personales.

"Dump"

Volcar. Imprimir el contenido de la memoria, disco o cinta sin formato de informe.

"Engine"

Máquina.

1. Procesador especializado, por ejemplo un procesador de gráficos. Al igual que cualquier máquina, cuanto más rápido funcione, más rápido se hace el trabajo.
2. Software que ejecuta una función de rutina elemental y altamente repetitiva, tal como un motor de base de datos, de gráficos o de diccionario.
3. Jerga para referirse a procesador.

"Extent"

Alcance. En un disco, espacio contiguo reservado para un archivo o una aplicación.

Hardware

Maquinaria y equipos (CPU, discos, cintas, modem, cables, etc.). Cuanto más memoria y almacenamiento en disco tiene un computador, más trabajo puede

hacer. Cuanto más rápidos sean la memoria y los discos para transmitir datos e instrucciones a la CPU, más rápido se hará el trabajo. Un requerimiento de hardware se basa en el tamaño de las bases de datos que se crearán y en el número de usuarios o aplicaciones que serán atendidas al mismo tiempo. ¿Cuánto?, ¿qué tan rápido?

"Fail stop crash"

Significa que cuando el procesador falla, este para, y deja de procesar y darle servicio a otros procesos.

"Heap"

Montículo, montón. En programación, grupo común de memoria libre disponible para el programa.

"Heisenbug"

Las fallas de software ocurren solamente una vez y causan que el sistema pare. A esto se le llama Heisenbug. Estos ocurren porque existe alguna interacción no usual entre diferentes componentes.

Hora pico

Horario de mayor sobrecarga transaccional para el sistema, períodos de tiempo en que se realizan varias transacciones en un sistema de bases de datos en un día.

"Hot spot"

Es una pieza de datos que es accesada por muchas transacciones y es actualizada por una. Este causa cuello de botella ya que por cada transacción que actualiza, las demás deben esperar.

Instancia

Se le llama instancia, al conjunto de memoria compartida, y procesos, necesarios para que un manejador de base de datos relacional este levantado.

Se le llama instancia también a una ocurrencia de registro, a una tupla.

"Lock"

Candado o mecanismo de bloqueo. Es un mecanismo utilizado por los manejadores de bases de datos, que permiten mantener consistente la base de datos a pesar de tener transacciones concurrentes.

"Log"

Operación de registro, diario, bitácora. Registro de actividad del computador, que se usa para propósitos estadísticos, así como para seguridad y recuperación.

Memoria compartida
(Shared Memory)

Espacio de memoria principal (RAM), utilizado por un conjunto de procesos (programas) para compartir información.

"Memory Dump"

Volcado de memoria, vaciado de memoria. Exhibición o impresión del contenido de la memoria. Cuando un programa tiene una terminación anormal, puede hacerse un volcado de memoria para examinar el estado del programa en el momento del estallido. El programador mira dentro de los buffers para ver con qué elementos de datos estaba trabajando cuando falló. Los contadores, las variables, los conmutadores y los indicadores también son inspeccionados.

"Mirroring"

Mecanismo utilizado para grabar datos de manera redundante; esto tiene como objetivo el poder proteger el sistema de fallas. Ver Disk Mirroring.

Módulo

Un segmento de un programa. En un sistema completo para poder construirlo, se subdivide en distintos módulos, los cuales construyen parte del sistema

completo, entonces se dice que el sistema es modularizado. Otra opción para realizar programación "Top Down".

Operación matemática (aritmética de módulo) en la cual el resultado es el resto de la división. Por ejemplo, 20 MOD 3 da 2, (20/3 = 6 con un resto de 2).

Monitor

Software especial utilizado para revisar ciertas áreas del manejador de bases de datos, del sistema operativo o del hardware.

"Multithreading"

Multilectura. Multitarea dentro de un solo programa. Se utiliza para procesar múltiples transacciones o mensajes en forma paralela. También se requiere para crear aplicaciones de audio y de video sincronizadas. Con frecuencia, las funciones de multilectura se escriben en código reentrante.

OLTP: (OnLine Transaction Processing)

Procesamiento de transacciones en línea.

"Oncheck"

Utilitario de Informix OnLine que sirve para chequear índices, por medio de este se puede determinar si el índice esta corrupto (tbcheck en la versión 5.0). Si el índice estuviera corrupto, Informix no lo utilizará, y se deberá borrar y luego re-crearlo.

"Overhead"

1. Carga general.
2. Cantidad de tiempo de procesamiento empleada por el software de sistema, como el sistema operativo, el monitor de TP o el administrador de la base de datos.
3. En comunicaciones, códigos adicionales transmitidos para fines de control y verificación de errores, que requieren más tiempo para su procesamiento.

"Paging"

Paginación, foliación. En memoria virtual, transferencia de segmentos de programas (páginas) dentro y fuera de la memoria.

"Parse, parsear"

Análisis gramatical. Analizar una sentencia o instrucción de lenguaje. Reduce palabras a unidades

funcionales que pueden convertirse a lenguaje de máquina.

Planificador de procesos o "Scheduler"

Parte del sistema operativo que inicia y termina las tareas (programas) en la computadora. También llamado dispatcher (despachador), el planificador mantiene una lista de las tareas a ejecutar y asignar los recursos de la computadora según se requieran.

"Pool o buffer pool"

Grupo de buffers. Area de memoria reservada para buffers.

"Quantum"

Período de tiempo en que el proceso planificador (ver scheduler) da para cada procesos ejecutado concurrentemente (sistema de multiprogramación).

Read/Write time (tiempo de lectura y escritura)

Es el tiempo para leer o escribir los datos en un track (entre 1 y 10 kilobytes por milisegundo).

Respuesta

Medida de tiempo tomada por el sistema para ejecutar comandos. La respuesta mide el rendimiento desde el punto de vista de usuarios individuales del sistema. En algunas situaciones el termino tiempo elapsado (elapsed time) denota lo que nosotros denominamos respuesta, y el termino respuesta denota el tiempo tomado por el sistema para retornar el primer byte del resultado.

"Rollback"

Operación de des-hecho. Permite deshacer las transacciones realizadas por una aplicación.

"Rotational delay"
(retardo rotacional)

El tiempo que espera hasta que la porción apropiada del disco (track) gira para colocarse debajo de la cabeza de lectura.

"Scheduling algorithm"

Algoritmo de planificación. Método utilizado para planificar la ejecución de tareas. La prioridad, el intervalo de tiempo en la cola de tareas y la

disponibilidad de recursos son ejemplos de criterios usados.

"Seek"

Localizar. Mover el brazo de acceso a la pista solicitada de un disco.

"Seek time"

El tiempo que toma mover la cabeza del disco hacia el apropiado track (cerca de 10 milisegundos).

Selección, "select" o "query"

Es aquella operación realizada por una aplicación que inter-actua con una base de datos, cuyo objetivo es retornar la información que está almacenada en dicha base de datos.

SMP "Symmetric multiprocessing"

Multiprocesamiento simétrico. Diseño de multiprocesamiento en el que a cualquier CPU puede asignársele cualquier tarea de aplicación. Una CPU actúa como un procesador de control, o planificador, que inicia el sistema, distribuye el trabajo a la siguiente CPU disponible y gestiona las peticiones de E/S. Por lo

general se ejecuta una copia del sistema operativo en cada CPU.

Software

Instrucciones para una computadora. Una serie de instrucciones que realizan una tarea en particular se llama programa. Las dos categorías principales son software de sistemas y software de aplicaciones.

"Spindle"

Eje sobre el cual giran los platos de un disco.

Swapeo ("Swap")

Intercambio. Reemplazo de un segmento de un programa en memoria por otro, y su restablecimiento a su estado original cuando se requiera. En los sistemas de memoria virtual, se denomina "paging".

Transaction processing

(Procesamiento de transacciones). Procesamiento de las transacciones en el momento de ser recibidas por el computador. También llamados sistemas en línea o de tiempo real, los archivos maestros se actualizan tan pronto como las transacciones se introducen en las terminales o llegan por las líneas de comunicaciones.
Si alguien guarda los recibos en una caja de zapatos, y los suma a fin de año por razones de impuestos, eso es procesamiento por lotes. Sin embargo, si usted compra

algo e inmediatamente añade la suma al total corriente, eso es procesamiento de transacciones.

"Threshold"

Es el máximo tiempo que una transacción esperará por un "lock" antes que sea abortada.

"Throughput"

Mide el número de tareas ejecutadas por el sistema por unidad de tiempo. "Throughput" mide el rendimiento desde el punto de vista del sistema.

"Tracks"

Son los círculos concéntricos que conforman el plato de un disco. En estos se almacenan los datos.

"Tupla"

Registro, instancia.

"Upgrade"

Sustituir algo ya instalado por una versión de lo mismo, pero mas reciente.

INTRODUCCIÓN

En Guatemala, se puede observar la proliferación de sistemas computacionales cada vez más complejos ya que el crecimiento de las empresas, tanto en su recurso humano como en su capacidad de procesar información rápidamente es inminente.

Debido a esto, dichos sistemas computacionales deben responder integra y rápidamente a las necesidades de información de sus empresas. Una forma de que los sistemas de información además de que sean eficaces, sean eficientes, es precisamente uniendo dos tipos de actividades: *el monitoreo y el afinamiento del sistema.*

En este trabajo se desea proporcionar al lector un concepto claro de lo que implican estas dos actividades. Esto incluye, saber quién es el responsable de realizar dichas actividades, cuando realizarlas, cómo realizarlas, por qué realizarlas; saber cuales son los recursos que pueden provocar los llamados cuellos de botella, y que potencialmente podrían disminuir la eficiencia del sistema (todo lo indicado es parte del capítulo uno).

En el capítulo dos se introduce a la mayoría de elementos que se pretende examinar y afinar, para dar una idea de posibles causas que pueden afectar el rendimiento de una base de datos. Este capítulo es genérico para cualquier manejador de base de datos relacional.

En el capítulo tres, se presentan siete manejadores de bases de datos relacionales y sus características más importantes, para que el lector esté familiarizado con la mayoría de software de este tipo que esta en el mercado, para que conozca sus características, lo cual podría ser útil en algún momento para la toma de decisiones de cuál manejador de base de datos se adecua mejor a los requerimientos de su empresa.

En los siguientes tres capítulos (cuatro, cinco y seis) se presentan los métodos de afinamiento de tres distintos manejadores de bases de datos (los mejor colocados en el mercado actual), estos capítulos no persiguen la comparación en calidad de productos, sino que su objetivo es presentar al lector las diferentes formas de afinamiento entre manejadores, ya que aunque la mayoría de estos programas contienen los mismos elementos en su arquitectura (memoria compartida, manejo de discos, candados, bitácoras, etc), estos mismos elementos se afinan de manera diferente en cada uno de los manejadores, e incluso hay elementos que en algunos manejadores no son afinables y en otros sí. Esta información se incluye para que el lector tome conciencia de que existen distintos manejadores, y que cada uno tiene sus propia formas de configuración.

En el capítulo siete se presenta un caso, en el que se hacen comparaciones entre una base de datos que no está afinada, y otra que sí.

1. AFINAMIENTO DE BASES DE DATOS

1.1. Importancia del afinamiento de una base de datos

En este capítulo, se explican las razones por las cuales se debe realizar el afinamiento de una base de datos, los recursos que se necesitan para realizar dicho proceso, como personal y tiempo, y además los factores que deben de incluirse para tomar una decisión sobre que se debe afinar.

1.2. ¿Por qué afinar una base de datos?

Los manejadores de bases de datos relacionales, en su mayoría, es un software altamente afinable. Generalmente, cuando dicho software es instalado, sus parámetros de configuración son colocados por "default", o son obtenidos de otras instalaciones de acuerdo a la experiencia de la persona que realiza la instalación, o de la cantidad de información, de usuarios y otros factores, que son calculados a groso modo.

Las bases de datos lógicas que han sido creadas en el manejador de bases de datos, van creciendo mientras los usuarios ingresan datos; cuando la cantidad de datos almacenada es considerablemente voluminosa, es necesario aumentar los parámetros de configuración del manejador de bases de datos, debido a que los recursos que utiliza el mismo son tomados del hardware o del

sistema operativo; Estos ya no son suficientes para manejar las grandes cantidades de información; sus resultados pueden ser notados en un decremento en la velocidad en que se realizan las transacciones. Por ejemplo, los manejadores de bases de datos por lo general apartan un espacio de memoria compartida, que al inicio podría ser suficiente para extraer datos de una manera rápida, sin embargo, cuando la cantidad de datos extraídos crece, dichos datos ya no pueden ubicarse en esta porción de memoria, por lo que empieza a grabar estos datos que están en memoria principal a disco, para poder subir el resto, lo que produce que los datos sean extraídos lentamente. A esto se le denomina "swapeo".

Si en una empresa sucede el siguiente caso: tiene aproximadamente un volumen de unos 100,000 registros diarios actualmente en esa aplicación, lo que hace que el sistema retorne un registro, 2 minutos después que se ha presionado la tecla para ejecución de selección. Además, en horas pico atienden aproximadamente a 500 personas; suponiendo que se ingresa un registro por persona, se tendría en horas pico 500 registros, esto significa que en un mes tendrían 15,000 registros más, que aumentaría el tiempo de retorno de registros por persona.

Todas estas situaciones hacen que se necesite cada vez más los conocimientos necesarios para afinar la base de datos.

En general, afinar una base de datos es: *La actividad de hacer que las transacciones realizadas por las aplicaciones de la base de datos sean ejecutadas con mayor velocidad.* "Mas rápidamente" significa usualmente un alto rendimiento, o sea que tenga un tiempo de respuesta bajo.

Para hacer que un sistema se ejecute rápidamente, el afinador de la base de datos debe de cambiar la manera en que la aplicación esta construida, las estructuras y parámetros del sistema de base de datos, la configuración del sistema operativo, o el hardware. Debido a esto el afinador del sistema debe de tener conocimiento de las aplicaciones, y del sistema de computación utilizado.

Entre los beneficios que una empresa puede tener, cuando se realiza afinamiento en su sistema, están:

1.2.1. Beneficios del afinamiento

1. Si se realiza un afinamiento completo del sistema, se podrá evitar comprar equipo adicional (como memoria, un procesador más potente, etc.).

2. A menudo hay empresas que desean actualizar su sistema, hacerlo más pequeño en tamaño ("downsize"), pero al mismo tiempo, obtener altos beneficios. Estas empresas pueden lograr un rendimiento similar, o mejor del que tenían con sus sistemas antiguos y de configuración costosa, realizando afinamiento a su sistema.

3. Si se tiene hardware sencillo y barato, entonces tendrá que pagar poco por mantenimiento y por "upgrades" de "software" y de "hardware".
4. Un alto rendimiento y un sistema adecuadamente afinado produce un tiempo de respuesta rápido; ésto hace que sus usuarios sean más productivos.

5. En adición a los beneficios financieros del afinamiento, hay beneficios humanos considerables. No hay nada más frustrante para un empleado que trata de ser productivo, que tener que esperar por recursos computacionales, o tener un tiempo de respuesta lento.

1.3. ¿Quién afina un sistema?

En general, varias personas deben intercambiar información, por lo que deben estar envueltas en el proceso de afinamiento para un sistema, por ejemplo:

1. Las personas que planean y dirigen el sistema deben de supervisar el proceso de afinamiento por completo, desde la planeación hasta la producción, incluyendo el monitoreo.

2. Los administradores de hardware y de software deben documentar y comunicar los cambios en la configuración de hardware y de software del sistema, puesto que estos cambios pueden repercutir en el diseño y administración de un sistema efectivo.

3. El administrador de la base de datos debe monitorear cuidadosamente y documentar la actividad del sistema, de manera que un síntoma no usual del sistema pueda ser identificado y corregido.

4. Los desarrolladores de aplicación deben de comunicar las estrategias de implementación, que seleccionen para que los módulos y las sentencias SQL puedan ser rápidas y de fácil identificación durante su afinamiento.

5. Los diseñadores de aplicación deben comunicar el diseño del sistema, para que todos puedan entender el flujo de datos en una aplicación.

Como puede observarse, todos están envueltos en el sistema, ya que tienen algún rol en el proceso de afinamiento. Cuando las personas mencionadas arriba, comunican y documentan las características del sistema, el afinamiento se vuelve significativamente fácil y rápido.

1.4. ¿ Cuándo se debe afinar una base de datos?

La mayoría de personas creen que el proceso de afinamiento inicia cuando los usuarios detectan que el tiempo de respuesta es lento. Esta es una estrategia pobre, ya que es demasiado tarde para aplicar algunos de los elementos de las estrategias de afinamiento efectivas.

Los diseñadores de aplicación necesitan colocar algunas políticas de rendimiento de aplicaciones, en la fase de diseño. De esta manera, durante el diseño y el desarrollo, los diseñadores de aplicación podrán considerar qué características del manejador podrán beneficiar el sistema.

Para diseñar un sistema con un rendimiento bueno, se debe eliminar costos, que más tarde puedan frustrar el rendimiento en la vida de la aplicación. Las figuras 1 y 2 muestran el costo y el beneficio relativo de afinar durante la vida de una aplicación. Como se puede ver, el tiempo más efectivo para afinar es durante la fase del diseño. Afinar durante esta fase permite beneficio máximo a menor costo.

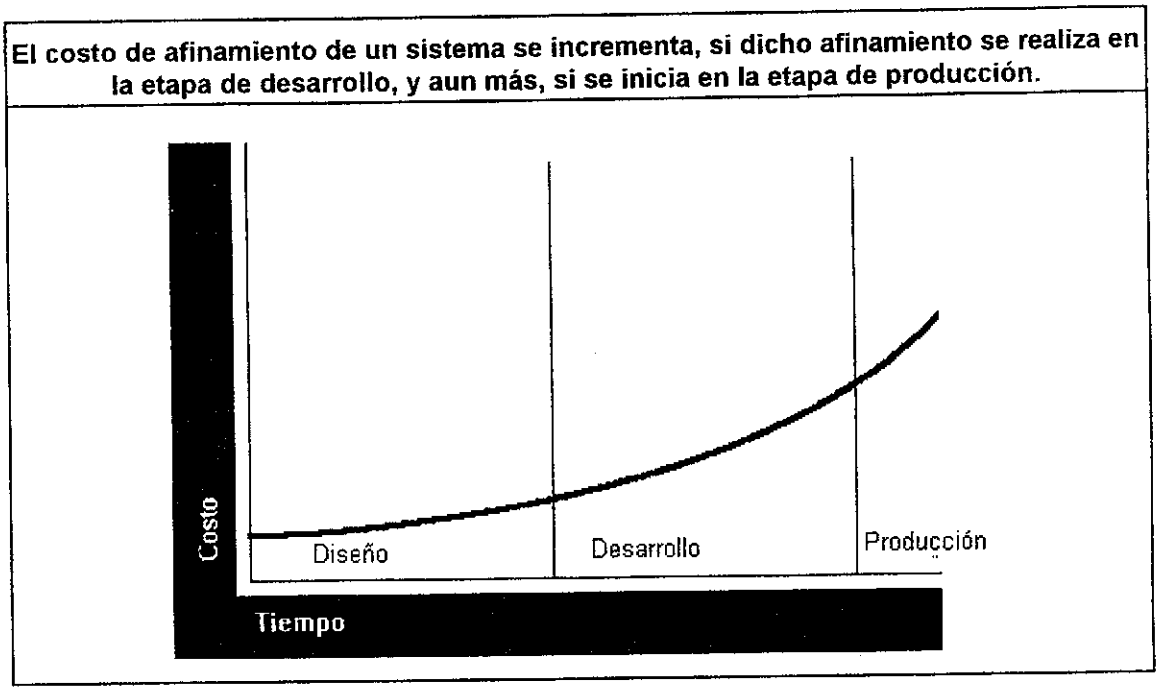


Figura 1. Costo de afinamiento de las etapas de diseño a la de producción

El beneficio al realizar un afinamiento desde la etapa de diseño es mayor que si se iniciara en la etapa de desarrollo y aun más que en la de producción, ya que en las siguientes etapas ya no se acarrean con problemas de afinamiento que han sido solventados en las etapas anteriores.

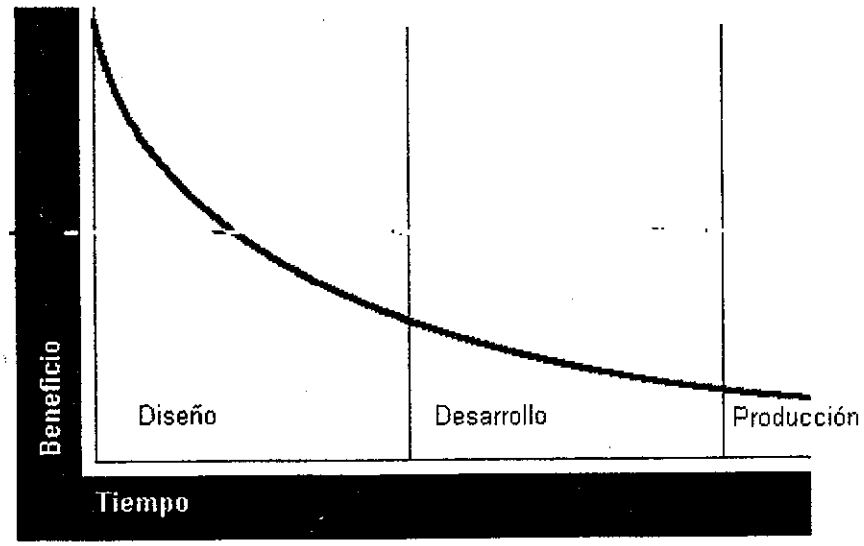


Figura 2. Beneficio de afinamiento de las etapas de diseño a la de producción

1.4.1. Afinamiento en la planeación del sistema

Una decisión equivocada durante la primera fase del ciclo del desarrollo del sistema, puede ser crucial. Definitivamente, existe una lista de actividades que deben realizarse antes de iniciar el diseño de una base de datos y el desarrollo de los programas; sin embargo, muchas de ellas son llevadas a cabo por el jefe de planeación del proyecto. Debe existir la oportunidad de compartir información y opiniones durante esta fase. Un ejemplo de esto podría ser el siguiente:

Para la selección de recursos en hardware y software, el jefe de planeación del proyecto debe de coordinarse definitivamente con el administrador del sistema. Debería de seguir los siguientes pasos:

1. Describir todos los sistemas y aplicaciones que requiere la organización. Debe pensarse globalmente durante esta etapa, además debe asegurarse que todas las aplicaciones sean pensadas de una manera integrada, de tal forma que diferentes aplicaciones tengan su información en una sola base de datos, ya que de esa manera no es necesario intercambiar datos entre aplicaciones, lo cual produce un consumo innecesario de recursos del sistema.

2. Seleccionar su configuración de hardware y de software. Lo más común es que se investigue en el mercado los diferentes productos que existen, algo importante es el valuar las implicaciones en el rendimiento si se decide por comprar cierto hardware o software, y no únicamente valuar las posibilidades por el precio. Algo que es efectivo, es preguntar a otras personas sobre el rendimiento que han tenido al comprar cierto producto, cómo les ha afectado.

3. Debe tomarse en cuenta, la proyección de un futuro crecimiento del hardware o software.

4. Realizar un calendario, incluyendo tiempos de holgura. Si se tiene planeado comprar hardware/software que necesita tiempo considerable para ser entregado, pídalo incluso con seis meses de anticipación, para evitar que los vendedores le hagan quedar mal.

5. Formar estándares para el diseño y desarrollo de su sistema.

6. Determinar que tiempo de respuesta será aceptable para su sistema.

1.4.2. Analizando y diseñando sus aplicaciones

La etapa de análisis y diseño es crucial para un buen rendimiento en el futuro. Todo el afinamiento realizado por programadores, DBA, administradores del sistema, no podrán arreglar un sistema que tenga errores en el análisis y en el diseño.

Algunas de las cosas que se deben de realizar en esta etapa son:

1. Diseñar el modelo de datos con un buen rendimiento; El diseño es una combinación de conocimientos que se encuentran en un libro, y además decisiones que realiza el diseñador. Esto significa que se diseña con los conceptos del libro, obteniendo entidades, relacionándolas, obtener atributos, etc. y el diseñador toma ciertas decisiones, las cuales le permitirán que el sistema sea más rápido, como por ejemplo el de-normalizar cuando sea necesario, el hecho de añadir redundancia en los datos para incrementar el rendimiento, etc. Hay una gran cantidad de decisiones en el mundo real que se deberían toman en cuenta para mejorar tiempos de respuesta a usuarios.

2. Diseñar los programas con un buen rendimiento; el diseño de los programas deben realizarse de manera modular, para permitir cambios y afinamientos. Por ejemplo, si varios programas necesitan llevar a cabo una función común, tal como actualizar el detalle de una cuenta contable, el diseñador debe pensar el código de la actualización como un módulo separado, que pueda ser llamado por cualquier otro programa que lo necesite.

El hecho de modularizar trae otros beneficios: Programas pequeños generalmente requieren poca memoria, también le reduce el trabajo en la etapa de mantenimiento, porque cualquier cambio al módulo es hecho solo una vez, en vez de realizarlo cada vez que el programa es incluido en el código.

3. Decidir cuando usar índices en tablas. Aprender a diseñar índices de manera efectiva podría ser una tarea prioritaria para el diseñador. Generalmente, el seleccionar datos con índices es más rápido que el realizar búsquedas completas en las tablas. Cuando se usan índices, frecuentemente se realizan unas pocas lecturas físicas a disco, mientras que búsquedas completas pueden realizar miles de accesos a disco, lo cual vuelve ineficiente el proceso. El problema básico consiste en consultar tablas que tienen funciones de agrupación, "joins", etc, y que por esto no utilicen el índice.

1.4.3. Desarrollando las aplicaciones

Durante la fase de desarrollo, se crean las tablas, índices y programas necesarios para la aplicación. En esta etapa se deben de realizar las siguientes actividades:

1. Escoger el optimizador, dependiendo del manejador de bases de datos con que se cuente. El optimizador examina las sentencias SQL y selecciona el plan de ejecución optimo, o regresa el camino, para la sentencia. La ejecución del plan, es la secuencia de pasos físicos que el manejador debe de realizar para retornar la información requerida.

2. Afinar las sentencias SQL, y los programas que han sido realizados en los distintos lenguajes con que cuentan los manejadores de base de datos.

3. También se debe de seleccionar la estrategia adecuada para el manejo de candados de su aplicación (aunque existen algunos manejadores de bases de datos que lo hacen en su mayoría, casi automáticamente).

1.4.4. Prueba y verificación de la calidad en el sistema

Durante esta fase se debe probar el sistema antes de trasladarlo a la etapa de producción. Inicialmente los módulos son probados de manera individual, sin embargo, se debe asegurar de que dichos módulos funcionan en conjunto, y que su rendimiento es aceptable. Los siguientes aspectos se deben tomar en cuenta:

1. Asegurarse que el volumen de datos de prueba es lo suficientemente largo. Si se tienen pocos datos, los problemas de rendimiento no se podrán observar, hasta que estén en producción.

2. Asegurarse que se está ejecutando los programas con el mismo tipo de configuración, que será utilizada en producción, por ejemplo: Si en producción utilizarán terminales caracter, de esa manera se realizarán las pruebas, o si en producción funcionará como cliente/servidor, así se harán las pruebas también.

3. Asegurarse que la configuración de hardware en que se ejecutan los programas como prueba sea más lenta, o igual con la que se correrá en producción. Muchas personas hacen lo contrario, prueban el software, en ocasiones en máquinas más rápidas, y cuando éste se libera en producción, entonces baja la velocidad.

4. Se debe informar al administrador de la base de datos sobre los problemas de rendimiento que se encuentren. Realmente el DBA es quien tiene el trabajo final, el de realizar el afinamiento de la base de datos para un buen rendimiento. Esto se hace mucho más pesado en la etapa de producción. Si se le puede dar buena información al DBA durante este estado, entonces ellos serán capaces de utilizar esos datos para afinar la base de datos de una manera eficiente, antes de que inicie la etapa de producción.

1.4.5. Monitoreo del rendimiento durante la etapa de producción.

Es recomendable que mientras el sistema esté activo en la etapa de producción, su rendimiento sea examinado, considerando las necesidades de los usuario. Idealmente, los sistemas liberados en producción, están listos para funcionar a la perfección. Posiblemente, inicie bien, pero poco a poco se van requiriendo más recursos, como memoria, espacio en disco, etc., debido a que existe crecimiento en el número de usuarios, etc. El DBA y el administrador del sistema comparten la responsabilidad de examinar los sistemas y los datos, tratando de asegurar que el manejador de bases de datos utilice bien sus recursos: memoria, disco, CPU y la red.

Entre las responsabilidades que tiene el DBA están:

1. Estar atento a los problemas de rendimiento, ser previsor. El DBA no debería esperar las quejas de los usuarios y reaccionar a estas. Muchos de los usuarios no reportan todos los problemas.

2. Tratar de utilizar todas (o por lo menos la mayoría) de las herramientas que están disponibles por el manejador de bases de datos.

3. De vez en cuando, correr procedimientos voluminosos que hagan transacciones a la base de datos en producción, solamente con usuarios que podrían hacerlo. Registrar tiempos de respuesta en intervalos discretos usando la misma forma que los usuarios de producción podrían hacerlo.

Las principales responsabilidades de monitoreo para los administradores incluyen:

1. Asegurar la velocidad del hardware y de la red sean aceptables para la cantidad y el tipo de procesamiento que se está realizando.
2. Examinar rendimiento, revisando memoria libre, actividad de pagineo y "swapeo", actividad de CPU, disco y red.

3. Asegurarse de la buena funcionalidad de los sistemas, asi como la actualizacion de los programas, como prevención para el buen funcionamiento de nuevos versiones del software que se tiene (nuevos versiones de manejadores de bases de datos, de software de red, etc.).

4. Coordinar el afinamiento del sistema operativo con el del afinamiento de la base de datos. Esto es debido a que en ocasiones es necesario reorganizar archivos a través de los discos, en general para evitar que el trabajo de uno, traiga elementos negativos al trabajo del otro, y viceversa.

1.5. Metas para el afinamiento.

Si se está diseñando o dándole mantenimiento a un sistema, se deben adoptar ciertas metas de rendimiento en el proceso de afinamiento, las cuales deben ser bien conocidas. Puede darse el caso, en que se gaste tiempo innecesariamente, tratando de afinar un sistema sin ganar nada, cambiando los parámetros de inicialización del manejador de la base de datos aleatoriamente (de manera fortuita). Uno de los métodos más eficientes para afinar un sistema es el siguiente:

1. Considerar el rendimiento en la fase de diseño.
2. Afinar el hardware y el software del sistema operativo.
3. Identificar cuellos de botella en el rendimiento de los recursos.
4. Determinar la causa de los problemas
5. Realizar acciones correctivas.

Cuando se está diseñando un sistema, se deben de trazar metas, por ejemplo: el tiempo de respuesta para la realización de un proceso debe ser de aproximadamente tres segundos. Cuando a una aplicación no se le conocen las metas, la identificación del cuello de botella provoca retraso, ya se debe determinar la causa, y luego realizar la acción correctiva. Durante el

desarrollo, se deben de realizar las pruebas para comparar si se lograron las metas establecidas en el diseño.

En cualquier evento, afinar es generalmente una serie de intercambios. Después que se ha determinado el cuello de botella, en ocasiones deberá sacrificar algunas otras áreas, para obtener los resultados deseados. Por ejemplo, si se tiene problemas de accesos de E/S (disco), posiblemente tendrá que comprar más memoria o más disco. Si una compra no es posible, entonces se tendrá que limitar la concurrencia de su sistema en la cantidad de usuarios. Sin embargo, si se han definido claramente metas para el rendimiento, se tendrán ya las decisiones por las cuales intercambiar, ya que tiene definidas además sus principales prioridades.

1.6. ¿Cuáles son las causas principales de los problemas de rendimiento?

Existen muchas razones por las cuales se obtiene un rendimiento pobre en un sistema. Pueden haber problemas de diseño y de desarrollo, problemas con los recursos del sistema, problemas de memoria, problemas de disco, de CPU, y problemas de la red.

1.6.1. Problemas con diseño y desarrollo

A nivel de diseño:

Los problemas de diseño son causados por analistas y diseñadores quienes no:

- Han considerado rendimiento cuando definieron el modelo.
- Diseñaron programas que son apropiados para una base de datos relacional.
- Diseñaron programas que se apropian de la configuración de hardware que están usando.

A nivel de programas:

- Los problemas son causados generalmente por programadores, quienes no hacen uso eficiente de las herramientas de optimización del manejador de base de datos.

A nivel de la base de datos:

- Los problemas son causados generalmente por los DBA's que no tienen:
- En uso efectivo los recursos de la máquina.
- Una dispersión estructurada de la base de datos en el acceso a disco.
- Colocados los valores adecuados que deben ir en los parámetros que configuran el manejador de base de datos.

Como una consecuencia de lo anterior, puede ocurrir lo siguiente:

Excesivo acceso de E/S a disco.

- Los accesos de E/S del disco no son balanceadas a través de los discos.
- La base de datos esta fragmentada (dependiendo del manejador de base de datos).
- La base de datos no esta efectivamente indexada.

A nivel del sistema:

Los problemas del sistema ocurren como resultado de:

- Otros sistemas ajenos al manejador de la base de datos, afectan el rendimiento de este.
- El sistema operativo no está afinado adecuadamente.
- El tamaño de la máquina o su configuración no es adecuada para soportar el manejador de base de datos.

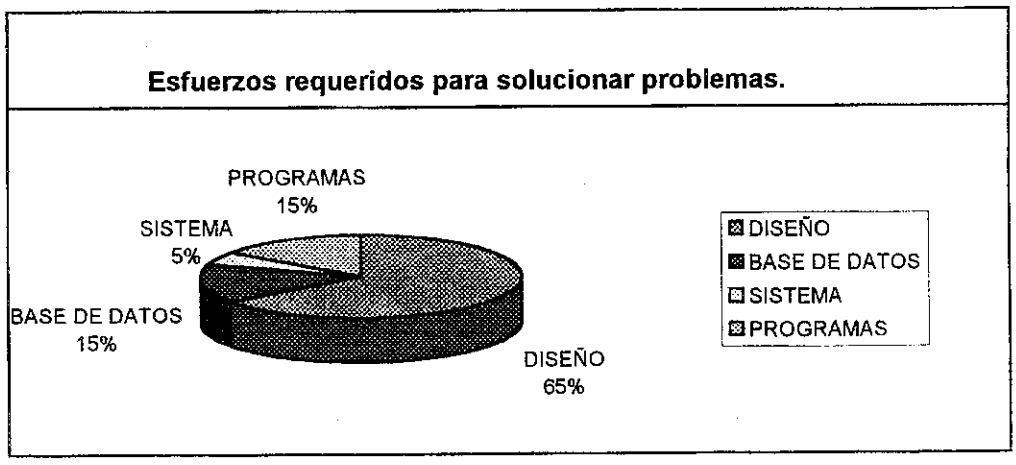


Figura 3. Esfuerzos para solucionar problemas en distintas etapas de desarrollo de software

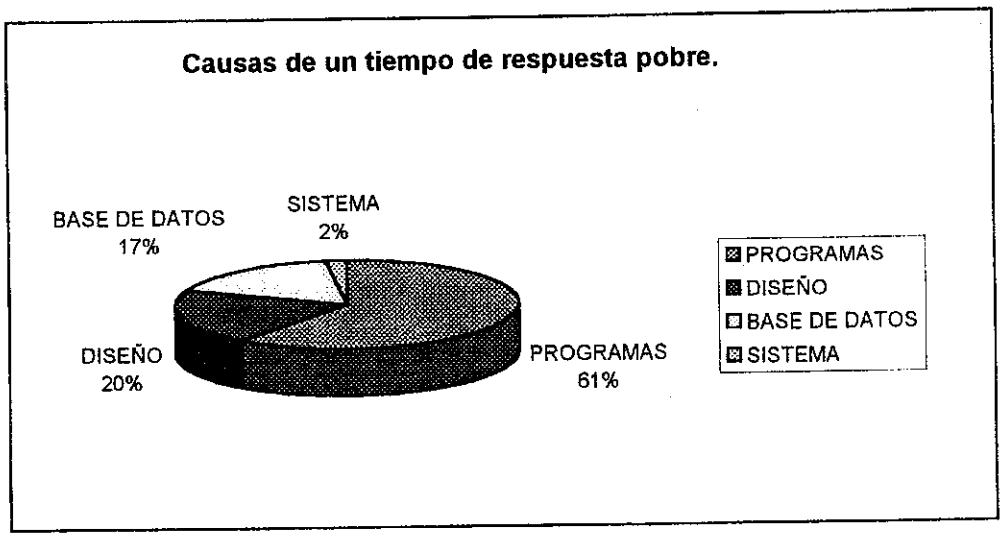


Figura 4. Posibles causas de un tiempo de respuesta pobre a nivel de las etapas de desarrollo de software.

La mejor manera de solucionar problemas de rendimiento en un sistema, es mantener los problemas de rendimiento con una alta prioridad. Como se muestra en la figura, el tratar de solucionar problemas de rendimiento resultantes de un diseño pobre de su modelo de datos y sus programas no es tan simple como inicializar un parámetro o ejecutar un diagnóstico. La única manera de solventar dicho problema es reorganizar y recodificar. De la misma manera que un programador realizará un programa pobre en tiempo de respuesta si el modelo de la base de datos ha sido mal diseñada, un DBA podrá hacer poco examinando y afinando la base de datos si los programadores realizan una mala programación, con tiempos de respuesta inadecuados.

1.6.2. Problemas con recursos del sistema

Para obtener un mejor rendimiento de su sistema, se deben cuidar cuatro componentes básicos de la máquina, los cuales interactuan y afectan el rendimiento de su sistema, los cuales son:

1. **Memoria** Se debe tener cuidado con los posibles cuellos de botella que pueden ocurrir, si no existe suficiente memoria para acomodar las necesidades del sistema. Cuando esto sucede, lo más probable es que exista "pagineo" (movimiento de porciones de procesos a disco) y "swapeo" (transferencia de información procesada de memoria a disco para obtener un poco de memoria libre).

2. **Accesos de E/S de disco** Pueden existir cuellos de botella en discos, cuando en uno o más discos exceden el promedio (razón) de Entradas/Salidas recomendado.

3. **CPU** Cuellos de botella de CPU pueden ocurrir cuando el sistema operativo o los programas del usuario están demandando demasiado del CPU. Esto es, a menudo, causado por "pagineo" o "swapeo" excesivo.

4. **RED** Los cuellos de botella en la red puede ocurrir cuando la cantidad de tráfico en al red es demasiado grande, o cuando existen colisiones en la red.

1.7. Principios del afinamiento

Podría afirmarse que el afinamiento tiene dos ángulos, uno fácil y otro difícil.

El afinamiento es fácil, porque no necesita de fórmulas o teoremas complejos, y difíciles de aplicar. Muchos investigadores académicos e industriales han tratado de representar el afinamiento en bases matemáticas. Generalmente, estos esfuerzos han fracasado, debido a que descansan en suposiciones irrealizables.

El afinamiento es difícil porque los principios y el conocimiento de sentido común que requiere, implica un profundo conocimiento de la aplicación, del software de la base de datos, del sistema operativo y del hardware. Generalmente en la mayoría de documentación ofrecen reglas prácticas para el afinamiento, pero no mencionan sus limitantes.

Por ejemplo: siempre se menciona que en una aplicación NUNCA se debe de usar funciones de agrupación (tal como el AVG) cuando existen transacciones con un tiempo de respuesta crítico. La principal razón es que tales funciones deben de buscar substanciales cantidades de datos y de esta manera pueden bloquear a otros queries. Así las reglas son generalmente verdaderas, pero esto podría no aplicar en el caso de que el promedio (AVG) use pocas tuplas que estén siendo seleccionadas por un índice. Esto nos deja de moraleja de que el afinador del sistema debe de entender las reglas antes de aplicarlas. Un afinador bien informado deberá tomar las reglas como: un buen consejo en la mayoría de situaciones.

1.7.1. Cuatro reglas principales

Las cuatro reglas principales que deben de prevalecer como consideraciones de rendimiento son:

1. Pensar globalmente: arreglar localmente.
2. Romper cuellos de botella.
3. Costos de inicio y de ejecución.
4. Hacer que el servidor realice lo que este debe de hacer, y no mas.

1.7.1.1.Pensar globalmente: arreglar localmente

Un afinamiento efectivo requiere una apropiada identificación del problema y un mínimo de intervención. Esto implica usar las cantidades correctas y las conclusiones correctas. Un ejemplo puede ser: un aprovechamiento común para afinar globalmente, es observar primero las estadísticas de hardware para determinar la utilización del procesador, actividad de entradas/salidas (E/S), el "pagineo", Etc.

Un afinador ingenuo podría reaccionar al primer valor alto en estas medidas (Un valor alto en la actividad de disco por ejemplo) comprando hardware para bajar esto (comprando otro disco por ejemplo). Pueden haber varias razones, por consiguiente el hecho de comprar nuevo hardware podría ser inapropiado. Por ejemplo: podría haber una alta actividad de disco, debido a que algunas consultas frecuentes en vez de usar el índice, están usando solamente la tabla, o porque los logs comparten un disco con datos que son

frecuentemente accesados. El hecho de crear un índice, o el de mover archivos de datos podría resultar mucho mas barato que comprar hardware extra.

1.7.1.2. Romper cuellos de botella

Un sistema lento rara vez es debido a que todos sus componentes están saturados. Generalmente, una de las partes del sistema, limita el rendimiento de los demás. A esto se le llama cuello de botella.

Una manera de pensar en un cuello de botella, es como una fotografía de un congestionamiento de carros en una carretera. El congestionamiento resulta de que una gran cantidad de carros deben pasar por un segmento de la carretera que es estrecho. Otra posible razón de un cuello de botella podría ser cuando un conjunto de calles van a desembocar a una sola, la cual no es del mismo ancho que las otras dos juntas.
Un cuello de botella se resuelve detectando este, y tomando una de las siguientes estrategias:

- Hacer que los conductores manejen mas rápido a través de la sección que contiene rutas mas pequeñas,

- Crear más rutas para reducir el paso de los carros por ruta, o animar a los conductores para que eviten pasar a horas pico.

La primera estrategia corresponde a una solución local (Pe.: la decisión de añadir un índice o re-escribir la consulta para hacer mejor uso de los índices existentes) esta podría ser la opción inicial a probar.

La segunda estrategia corresponde al particionamiento.

Particionamiento en un sistema de base de datos es una técnica para reducir la carga en ciertos componentes del sistema, dividiendo la carga por la utilización de mas recursos, o esparciendo dicha carga en el tiempo. El particionamiento puede romper cuellos de botella en muchas situaciones.
Veamos un ejemplo:

- En un banco, los clientes tienen varias posibilidades de estaciones de acceso para realizar sus transacciones. La mayoría de clientes accesan sus saldos por una estación que tienen en su casa. Si un sistema centralizado se sobrecarga, una solución natural es colocar la cuenta del cliente en un sub-sistema *i* que pueda ser accesado por la estación *i* del cliente. Esta es una forma de particionar en el espacio (recursos físicos).

- Los bloqueos de contención generalmente envuelven muy pocos recursos. A menudo una lista libre (la lista de páginas de buffer no usadas en la base de datos) sufren contención antes que los archivos de datos. Una solución es dividir tales recursos en piezas, en orden de reducir el número de accesos concurrentes por cada bloqueo. En este caso de las listas libres, esto podría significar crear varias listas libres, cada una apuntando a una porción de páginas libres. Esta es una forma de particionamiento lógico (de recursos bloqueados).

- Un sistema que incluya un pequeño conjunto de transacciones largas, además contengan un conjunto grande de transacciones cortas, y que dichas transacciones compartan el uso de las mismas tablas podría repercutir en un rendimiento bajo, ya que puede incurrir en bloqueos, y en contención de recursos. "Deadlock" pueden forzar a las transacciones largas a ser eliminadas de manera abrupta (abortadas), y éstas al mismo tiempo pueden bloquear a las transacciones cortas. También las transacciones largas, pueden estar ocupando grandes cantidades de memoria principal (buffer pool), y esto puede disminuir la velocidad de ejecución de dichas transacciones aun mas. Una posible solución es ejecutar las transacciones largas cuando hay poca actividad de transacciones cortas, de manera que no exista interferencia unas con otras, y además serializar las transacciones largas (ejecutar una después de otra, esto es particionamiento en el tiempo).

Matemáticamente, particionar significa dividir en conjuntos mutuamente separadas (no intersectadas). Desafortunadamente el particionamiento no siempre mejora el rendimiento, o puede salir demasiado caro. Por ejemplo, el hecho de añadir otras rutas para particionar un cuello de botella (en el caso del banco) puede incurrir en caminos de comunicación adicional para las transacciones, que podrían ser demasiado caro. Por esta razón el praticionamiento debe de hacerse con sumo cuidado.

Podríamos decir en conclusión que cuando se halla un cuello de botella, se debe primero de tratar de aumentar la velocidad del componente, y si ésto no funciona, entonces particionar.

1.7.1.3. Costos de inicio y de ejecución

Una gran cantidad de objetos construidos por la humanidad tienen la característica de dedicar recursos substanciales para iniciar. Por ejemplo en los carros (la ignición del sistema), ciertas clases de bulbos de luz e incluso los sistemas de base de datos.

El inicio de una operación de lectura a disco, implica un costo, sin embargo una vez leído el inicio, el disco puede extraer datos a altas velocidades. Así, el inicio de lectura de segmentos de 64 Kb del track de un disco probablemente será menos que dos veces tan caro que leer 512 bytes de un "track". Esto sugiere en planear búsquedas de tabla que podrían ser usadas frecuentemente.

El costo de parsear, llevar acabo análisis semántico, y seleccionar caminos de acceso por mas simples que parezcan las selecciones, implica un costo. Esto sugiere que a menudo se ejecuten selecciones que podrían se compiladas.

Obtener el efecto que se desea, con el mínimo de inicios.

1.7.1.4. Hacer que el servidor realice lo que éste debe de hacer y no más

Obtener el mejor rendimiento de un sistema, implica más que realizar afinamiento en el manejador de base de datos, el cual es una porción del sistema. Una importante pregunta de diseño es el hecho de que porción de trabajo se le debe de asignar al sistema de base de datos (en el servidor), y que porción a los programas de aplicación (en el cliente). Donde una particular tarea podría ser asignada dependiendo de los siguientes tres factores:

Los recursos computacionales son relativos del cliente y del servidor: si el servidor esta sobrecargado, entonces, todo lo demás esta igual, las tareas podrían estar fuera de carga hacia los clientes. Por ejemplo, muchos sistemas de base de datos orientadas a objetos permitirán que el programador de aplicaciones muevan los buffers de la base de datos al cliente. Esto es una buena idea para bajar contención, en aplicaciones de calculo intensivo.

Donde esta localizada la información relevante: suponga que alguna respuesta pueda ocurrir (pe.: escritura a pantalla) cuando algún cambio a la base de datos ocurre (pe.: la inserción a alguna tabla de la base de datos). Entonces un buen diseñador de sistemas, debería utilizar un trigger dentro de la base de datos en vez de aplicar un poll en la aplicación. Una solución polling realiza consultas periódicas a tablas para ver si estas han cambiado. Un trigger se dispara (ejecuta) solo cuando el cambio actual toma lugar.

Si las tareas de la base de datos interactuan con la pantalla: si esto sucede, entonces la parte que accesa la pantalla podría ser hecha fuera de la

transacción. La razón es que la interacción en la pantalla puede tomar largo tiempo (algunos segundos al menos). Si una transacción T incluye un intervalo, entonces T podría prevenir otra transacción desde accesar data que T esta cargando. Así, la transacción podría dividirse en tres pasos:

1. Una pequeña transacción que retorna los datos.
2. Una sesión interactiva que ocurre en el lado del cliente fuera de un contexto transaccional.
3. Una segunda pequeña sesión que instala los cambios archivados durante la sesión interactiva.

2. CONSIDERACIONES DEL AFINAMIENTO

2.1 Componentes

Este capítulo tiene como objetivo el estudio de las consideraciones de afinamiento, que se deben de hacer con los principales componentes (comunes) en la mayoría de sistemas manejadores de base de datos. Cada componente acarrea sus propias consideraciones de afinamiento:

Control de concurrencia: como minimizar la contención de candados (locks).

Recuperación y bitácora (logging): Como minimizar las bitácoras y "dumping overhead".

Sistema operativo como optimizar el tamaño de buffer, el procesamiento del planificador, etc.

Hardware como localizar discos, memoria de acceso aleatorio, y procesador.

La siguiente figura muestra los componentes principales para todo sistema de base de datos:

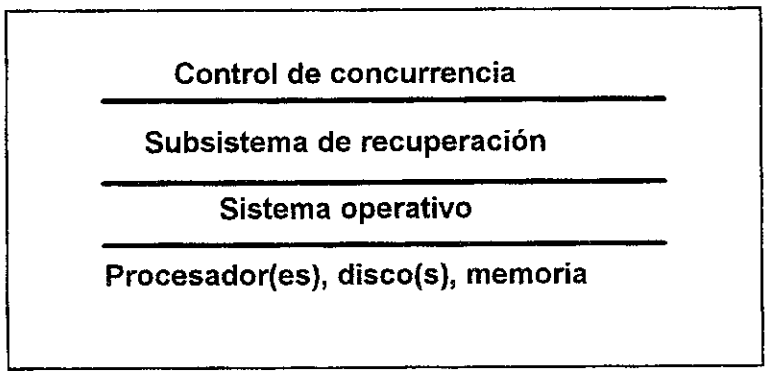


Figura 5. Componentes principales de un sistema de base de datos. FUENTE: Database Tuning, Dennis E. Shasha.

De esta manera la herencia de prioridad previene el problema de inversión de prioridad.

- Si su sistema no esta protegido para inversión de prioridad, entonces de la misma prioridad a todas las transacciones, para evitar el conflicto.

- Si su sistema esta protegido para inversión de prioridad, entonces seleccione la prioridad mas alta para las transacciones en línea, en vez de dársela a las transacciones en "batch". Esto sin embargo debe de manejarse cuidadosamente, ya que podría caer en el error de generar muchos cambios ("switchs") de apuntadores, y esto sería perjudicial al "throughput".

2.1.1 "Database buffers"

Debido a que los accesos a disco toman un tiempo mucho mas largo que los accesos a memoria aleatoria (RAM), el afinador de la base de datos debe de tratar de minimizar el número de accesos a disco. Una manera e hacer esto es almacenar la base de datos completa en RAM. Esto es factible solamente para ciertas aplicaciones. Sin embargo, la meta de afinar memoria para todas las demás aplicaciones, es asegurarse que las lecturas de páginas (páginas se refieren tanto a datos como a "buffers" para procedimientos almacenados o a otros "buffers" utilizados por el manejador de base de datos que se tenga), en su mayoría se realicen a memoria, y no sean accesadas a disco.

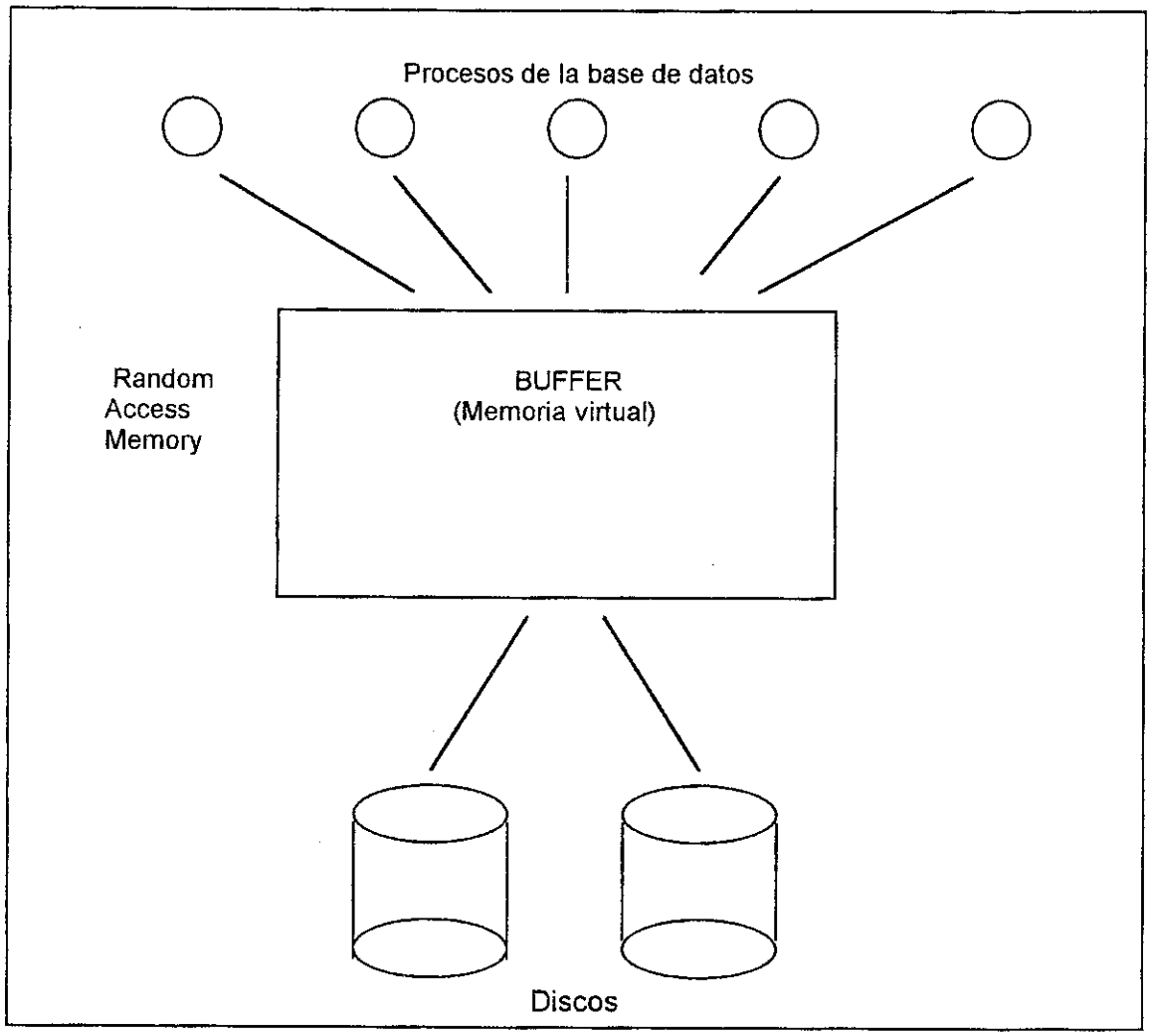


Figura 6. Posicionamiento de los buffers en memoria virtual o memoria RAM.
FUENTE: Database Tuning, Dennis E. Shasha.

2.2 Candados ("Lockings") y control de concurrencia

Las aplicaciones dividen su trabajo en *transacciones*. Cuando una transacción es ejecutada, esta accesa la base de datos y lleva a cabo ciertos cálculos computacionales localmente. La mayor suposición en que los programadores de aplicación caen, es que cada transacción parece ejecutarse aisladamente, sin actividad transaccional. Debido a que esta noción de aislabilidad, sugiera indivisibilidad, las transacciones garantizadas son también llamadas atomicidad garantizada. Por ese motivo, los investigadores de bases de datos, nunca se molestan en estudiar esto.

La secuencia de transacciones dentro de un programa de aplicación, es tomado como un todo. Se dice que entre las primeras dos transacciones ejecutadas por un programa de aplicación, y otros programas de aplicaciones, puede ser ejecutadas transacciones que pueden modificar campos de datos tanto accesadas por uno o ambos de las primeras dos transacciones. Por este motivo, el largo de la transacción debe tiene implicaciones correctivas muy importantes.

Veamos por ejemplo: se supone que un programa de aplicación procesa una compra, añadiendo el valor de un artículo al inventario, y substraendo el dinero pagado al contado. Los requerimientos especificados en la aplicación requieren que los pagos al contado nunca estén en negativo, así que la transacción se le da una operación de deshacer (rollback), si se substrae dinero de caja, que cause que el balance se vuelva negativo.
Para reducir el tiempo de recursos tomados por candados ("locks"), los diseñadores de la aplicación dividen esto en dos transacciones:

- La primera transacción chequea si hay suficiente dinero en caja para pagar el artículo. Si es así, la primera transacción añade el valor del artículo al inventario. De otra manera se aborta la aplicación de compra.

- La segunda transacción substrae el valor del artículo de caja.

Se supone ahora el siguiente caso:

Existen actualmente cien quetzales disponibles en caja cuando el primer programa de aplicación empieza a ejecutarse. Un artículo es comprado al costo de setenta y cinto quetzales. Así, esta primera transacción le da la operación de grabación ("commit"). Entonces sucede que otra ejecución de este mismo programa es efectuada, causando la substracción de cincuenta quetzales del efectivo. Cuando la primera ejecución del programa le da la operación de gravar (commit) a la segunda transacción, la caja tendrá un déficit de veinticinco quetzales.

De esta manera, dividiendo la aplicación en dos transacciones, puede resultar en un estado de base de datos inconsistente. Este problema posiblemente ocurra en pocas ocasiones.

Este es un problema en que se busco mejorar el rendimiento, pero se olvido las consideraciones de consistencia. ¿Qué se puede hacer en este punto?
Existen estas opciones a la hora de tratar de afinar un proceso que incluye candados:
El número de candados ("locks") que obtiene cada transacción (mientras menos utilice mejor será el rendimiento).

El tipo de candados ("lock") que se use (los candados de lectura son mejores para optimizar el rendimiento del sistema).

La cantidad de tiempo en que el recursos esta tomado (cuanto menos tiempo este bloqueado mejor será el rendimiento).

Para afinar candados, se deben de tomar los siguientes elementos:

- Eliminar el bloqueo ("locking") cuando este es innecesario.
- Hacer sus transacciones cortas.
- Utilizar las facilidades de lectura que permite el RDBMS (Pe.: la opción de ejecución de selección en modo de solo lectura que utiliza candados compartidos)
- Controlar la granularidad del bloqueo (bloque - a nivel de tabla, de página o de tupla).
- La sentencias DDL (Data Description language - Lenguaje de descripción de datos) son consideradas perjudiciales ya que bajan el rendimiento de un sistema, ya que tienen que modificar el catalogo del sistema en varias partes.
- Evitar los "hot spots" ("hot spot" son piezas de datos que son accesadas por muchas transacciones y que es actualizada por una (transacción), esto crea cuellos de botella, ya que las demás transacciones están esperando a que esta pieza sea actualizada.

Nota: solamente se da una pequeña descripción de este tema, ya que este pertenece a lo que es afinamiento de aplicaciones, la cual no es el objetivo principal de este trabajo.

2.3 Bitácora (Logging) y el subsistema de recuperación

Muchos de los sistemas manejadores de bases de datos afirman lo siguiente: *nuestro sistema está lógica y físicamente integrado un mecanismo de recuperación que protege la integridad de la base de datos en caso de una falla de hardware y de software.*

Esta afirmación es exagerada, debido a que una falla arbitraria de software podría transformar un estado correcto de la base de datos a un estado erróneo de la misma. Similarmente, existen suficientes fallas de hardware para causar que los datos de pierdan o se corrompan. Para el mejor manejador de base de datos, la aseveración es mas parecida a esta:

Nuestro sistema puede proteger la integridad de la base de datos contra fallas simples (de procesador, de red, o de cualquier drive de disco) y unas pocas fallas de software (fallas de cliente, unos pocos errores de direccionamiento dentro del servidor, y "fail-stop crashes" del sistema operativo).

En la práctica, existen dos clases de fallas de hardware que pueden ser toleradas:

Una falla del tipo *"fail-stop"* de un procesador, la cual podría borrar la información de la RAM. (Fail-Stop significa que cuando el procesador falla, este para. Un caso de excepción son las máquinas del tipo "fault tolerant" - tolerante a fallas, que incluyen una especie de mecanismo en espejo (mirroring) de todo incluyendo de procesador).

La falla del tipo "fail-stop" de disco, en el cual se provee suficiente redundancia de disco para poder levantar el sistema.

2.3.1 Principios de recuperación

Las transacciones son también la unidad de recuperación en los siguientes sentidos:

1. El efecto de transacciones grabadas (commited transactions) debe ser permanente. Estos es, cambios que deben de persistir aún después de la finalización de las transacciones que realizan estos cambios.

2. Las transacciones deben ser atómicas: Esto es, siguiendo la recuperación de una falla de "hardware", es posible reconstruir la base de datos reflejando las actualizaciones de todas las transacciones grabadas (exitosamente completadas). Además, debe ser posible el eliminar los efectos de cualquier actualización llevada a cabo por una transacción no terminada o abortada.

Realizar el primer objetivo significa el poner los datos de transacciones grabadas en un *almacenamiento estable* (o sea un almacenamiento inmune a fallas). La completa inmunidad es imposible, pero si es posible una buena aproximación. Como primer paso, el estado estable de almacenamiento, debe ser construido en alguna media (discos, tapes, o una RAM con batería para respaldo) la cual sobreviva a fallas. Tal media es llamada durable. En el segundo paso, para sobrevivir a fallas de media durable, tal como el hecho de que un disco se arruine (también llamado "crashes" de disco), los datos deben ser replicados en varias unidades de media durable, tal como la redundancia de discos.

2.3.1.1 Archivación de la atomicidad de transacciones

Los algoritmos para archivar transacciones atómicas, están basados en dos principios simples:

Antes de que una transacción sea grabada, debe de ser posible deshacer el efecto de esta transacción, aún si la memoria principal (RAM) falla. Esto implica que las imágenes anteriores de las actualizaciones de la transacción deben ser guardadas en almacenamiento estable hasta la hora de grabar la transacción. En una falla, estas pueden ser escritas a los discos de la base de datos si aún no están allí.

Una vez que la transacción ha sido grabada, debe ser posible colocar las actualizaciones de las transacciones hechas hacia la base de datos, aún si la memoria principal (RAM) falla. Por lo tanto las imágenes después de las actualizaciones de la transacción deben ser escritas en un almacenamiento estable algún tiempo antes de que las transacciones grabadas tomen lugar. De esta manera, si hay una falla a cualquier hora después de que la grabación de la transacción tome lugar, será posible deshacer los efectos de estas actualizaciones.

Sin embargo, también es cierto que la base de datos solamente puede almacenar una de las dos clases de imágenes. De esta manera, el almacenamiento estable debe de contener mas que solo la base de datos. La otra área de almacenamiento estable es la bitácora (*log*). Esta puede contener imágenes posteriores, imágenes anteriores o ambas.

Los algoritmos comerciales de las bitácoras (logs) trabajan de la siguiente manera:

- Los subsistemas de recuperación escriben las imágenes posteriores de cada transacción de actualización a la bitácora antes de que la transacción sea grabada, en orden de satisfacer el principio 2 descrito arriba. Una decisión importante de afinamiento, es determinar si las transacciones de imágenes posteriores podrían ser escritas a la base de datos inmediatamente a disco después de grabar o si se decide retardar las actualizaciones. Retardar es mejor para liberar de fallas de rendimiento, pero puede hacer lento el tiempo de recuperación.

- En la mayoría de sistemas, se escribe imágenes posteriores que no están grabadas en los discos de la base de datos, cuando ya no existe espacio en los buffers de la base de datos. Para no poner en peligro el principio 1, el subsistema de recuperación debe asegurar que las bitácoras (logs), contengan la imagen anterior de cada dato. Esto puede hacerse, con la escritura explícita de la imagen anterior o usando el valor escrito en la última escritura grabada de este dato. Esto es conocido como el algoritmo de "redo-undo logging".

Cada algoritmo de bitácora (logging) establece la siguiente ecuación:

el estado actual de la base de datos =

estado actual de la base de datos en disco + *bitácora (log)*

El estado actual de la base de datos es el estado que refleja a todas las transacciones grabadas. En contraste, la base de datos en disco refleja solo las transacciones grabadas físicamente en los discos de la base de datos. Durante una operación normal, algunas transacciones deben de haber sido grabadas en la bitácora, pero algunas de estas actualizaciones, aún no han sido escritas a los discos de la base de datos.

2.3.1.2 Variantes del manejo de bitácora ("logging")

La mayoría de sistemas comerciales guardan una página completa cuando una porción de la página es modificada. Esto es llamado manejo de bitácora o "logging" a nivel de página. También es posible guardar solamente una porción de la página, por ejemplo, dividir la página en un número de porciones y luego solamente guardar las porciones modificadas. Esto es llamado manejo de bitácora a nivel de "byte". Esto salva espacio de disco, especialmente si existen páginas largas. Una tercera posibilidad es guardar cada registro cambiado. Esto es llamado manejo de bitácora a nivel de registro. Y algunos sistemas orientados a objetos, usan de manejo de bitácora a nivel de objeto.

2.3.1.3 Punto de verificación ("Checkpoints")

Para prevenir el crecimiento de las bitácoras, los subsistemas de recuperación periódicamente copian las últimas actualizaciones grabadas desde la bitácora hacia los discos de la base de datos. A esto se le denomina un punto de verificación ("checkpoint"). La configuración del intervalo entre los

puntos de verificación es un parámetro de afinamiento. El punto de verificación causa sobrecarga, pero salva espacio en algunos casos, y reduce el tiempo de recuperación.

2.3.1.4 "Database dumps"

Un "database dump" es un estado de transacción consistente de la base de datos completa, en un tiempo dado. Un estado de transacción consistente es uno que refleja las actualizaciones de grabadas solamente. Se debe de notar que, en contraste de un punto de verificación, el "database dump" consiste en la base de datos completa (en vez de incluir solamente las últimas actualizaciones).

Si una falla corrompe los discos de la base de datos, entonces es posible reconstruir esta a un estado correcto usando el database dump previo, combinado con la bitácora. (En ausencia de un "dump", una falla de un disco de base de datos incluye perdida de datos, a menos que los discos que contienen los datos, estén duplicados en al menos otro disco). Esto se puede representar en la siguiente ecuación:

El estado actual de la base de datos = bitácora (log) + "database dump".

2.3.1.5 Grabar en grupos

Si varias transacciones son grabadas a causan una escritura a una bitácora, entonces la bitácora en disco podrían tener un cuello de botella. Por lo tanto, mucho sistemas usan la estrategia conocida de "Grabar en grupos"

(group commit). De acuerdo a esta estrategia, las actualizaciones de muchas transacciones son escritas a la bitácora en un solo acceso a disco. Si la aplicación, debe soportar muchas transacciones de actualización concurrentes (que sean pequeñas) entonces se debe de asegurar de que el sistema manejador de base de datos contenga la opción de grabación en grupos (group commit).

La única desventaja del la grabación en grupos, es que los candados de transacciones no pueden ser liberados hasta que las actualizaciones sean escritas a la bitácora.

2.3.2 Afinamiento de subsistemas de recuperación

Existen cuatro formas principales para afinar subsistemas de recuperación. Estos se pueden aplicar de manera individual o en combinación:

1. Colocar las bitácoras en discos dedicados, esto es para evitar el movimiento excesivo del brazo del disco (seeks)

2. Retardar la escritura de actualizaciones a los discos de la base de datos tanto como sea posible.

3. Estudiar el tiempo de recuperación contra el tiempo y espacio de sobrecarga cuando se configura el intervalo de los puntos de chequeo y el de database dump.

4. Reducir el tamaño de las transacciones de actualizaciones largas.

2.3.2.1 Colocar la bitácora en discos separados

Debido a que los archivos de bitácora están en almacenamiento estable, y porque en muchos sistemas, el almacenamiento estable se realiza en discos, las transacciones que se actualizan en disco, deben ser escritas en la bitácora. Esto en cierto sentido no es bueno, debido a que los discos pueden fallar. Sin embargo, en ausencia de fallas, los archivos de bitácora pueden ser escritos secuencialmente y en pedazos largos y contiguos. Por lo tanto, si un disco solamente tiene la respectiva bitácora, entonces el disco raramente tendrá contención y así puede mantener un buen promedio de accesos de E/S. Se recomienda por lo tanto que la bitácora se coloquen en discos separados.

Un disco es una colección de platos colocados uno sobre otro, y que rotan al rededor de un eje común (llamado "spindle"). Cada plato, a excepción del los platos de los extremos, tienen dos superficies de lectura - escritura. (La superficie de los platos externos, no son utilizados) Los datos de cada superficie esta colocada en los "tracks", y cada uno es un circulo. Cada plato esta asociado con una cabeza de disco. Para accesar datos en el "track" *i* de un plato dado, la cabeza del disco debe estar sobre al "track" *i* . En cualquier momento, todas las cabezas de disco asociadas al disco están sobre el mismo "track" en sus respectivos platos. El conjunto de "tracks" *i's* es llamado el *i-ésimo* cilindro. Así, el conjunto de cabezas del disco se mueven de cilindro en cilindro.

El tiempo que toma le toma accesar un disco, está compuesto por tres componentes significativos:

- *"Seek time"* El tiempo que le toma mover la cabeza del disco hacia el "track" apropiado (cerca de 10 mili segundos).
- *"Rotational delay"* (Retardo Rotacional). El tiempo de espera hasta que la porción apropiada del "track" esta bajo la cabeza del disco.
- *"Read/Write time"* (Tiempo de lectura/escritura). El tiempo para leer o escribir los datos del respectivo "track" (entre 1 a 10 kilobytes por milisegundo).

El retardo rotacional puede ser minimizado a la hora de leer o escribir pedazos grandes de datos a la vez. De hecho, el tiempo para leer o escribir un "track", no es mucho mas grande que el tiempo para leer o escribir solamente una porción del "track". Esto explica del porqué los grupos de grabaciones tienen importancia en la optimización, ya que una aplicación en su principio tiene un costo de inicialización alto, pero cuando ya esta corriendo, el costo es reducido.

El "Seek time" o tiempo de búsqueda puede ser minimizado de dos formas:

- La mejor manera, es asegurarse que accesos subsecuentes, continúen desde donde el último acceso se quedó. Esto explica el porque es bueno mantener los datos de las bitácoras en un disco propio. Cuando se escriben tales datos, el sistema operativo se asegurará que después de que el track es llenado, las siguientes escrituras se realicen en el mismo cilindro (menor movimiento para las cabezas, mayor rapidez para las búsquedas).

- La segunda mejor manera, es colocar los datos accesados frecuentemente en la mitad del track del disco, si este es magnético.

Los efectos de la localización ("seek"), y del retardo rotacional pueden ser substanciales. Una buena calidad de disco puede llevar a cabo 50 accesos aleatorios de disco por segundo. (Un acceso aleatorio de disco es cuando se accesa un cilindro y un "track" sin tener en cuenta los accesos previos). Así, la mayoría de estos accesos envolverán tiempo de localización, y retardo rotacional) Si cada acceso retorna una página de 4 kilobytes, entonces el total de tareas ejecutadas por el sistema por unidad de tiempo ("throughput") será de 200 kilobytes por segundo. Si se lee (o escribe) secuencialmente, el mismo disco puede ofrecer un throughput entre 1 y 10 megabytes por segundo. Así, un disco bien organizado puede tener un factor de 10 o mas veces mas rápido que uno pobremente organizado.

2.3.2.2 "Buffered Commits" (Grabado de Buffers)

Antes de grabar una transacción de la base de datos, esta escribe la imagen posterior de dicha transacción, actualizando así la bitácora. Pero ¿qué sucede exactamente después de la grabación?

Existen dos posibilidades:

- La estrategia de no grabar con buffer ("**Unbuffered commit strategy**"): Escribe las imágenes posteriores hacia la base de datos después de hacer la grabación (commit).

- La estrategia de buffers grabados (“**buffered commit strategy**”): Conserva cada una de las imágenes posteriores *x*, en la memoria principal (RAM) hasta que sea conveniente escribir estas. Normalmente, significa que espera hasta que las cabezas del disco de la base de datos están sobre el cilindro que contiene a *x*.

Al usar una estrategia de buffers grabados, la recuperación de fallas en la memoria principal (RAM) es mucho mas rápida. La razón es que la mayoría de las actualizaciones ya son transacciones grabadas, y por lo tanto ya están en la base de datos.

La estrategia de buffer grabados da un alto rendimiento durante las operaciones normales. Existen dos razones para esto:

1. La escritura de las actualizaciones de transacciones grabadas hacia los discos donde están las bitácoras, constituyen una escritura secuencial (no hay localizaciones - “seeks”), si estas mismas actualizaciones pueden ser distribuidas en el disco de la base de datos. La escritura de estas actualizaciones inmediatamente hacia los discos de la base de datos, implica una estrategia de buffers no grabados, esto causará que los discos de la base de datos hagan muchas escrituras aleatorias (por lo tanto habrán muchas localizaciones - “seeks”). De otra manera, con la estrategia de buffers grabados, se puede planear las escrituras evitando localizaciones, así se reducirá la carga en los discos de la base de datos.

- Los buffers de la base de datos pueden no llenarse con las páginas grabadas que no han sido bajadas a disco. (esto sugiere escribir las páginas grabadas recientemente a disco en vez de esperar).

- Podrían haber pocas escrituras aleatorias, a los discos (esto sugiere escribir tales páginas después, en vez de hacerlo pronto).

Cada manejador de sistemas de base de datos, ofrecen sus propias herramientas para esta facilidad. Por ejemplo INGRES permite al administrador de la base de datos, especificar el número de 'daemons' demonios multitarea que escriben páginas grabadas a los discos de la base de datos. En contraste, ORACLE (que usa los parámetros del proceso llamado DBWR) permite al administrador especificar cuando el sistema debe de escribir las páginas grabadas a los discos de la base de datos, basados en el número de páginas que no han sido escritas.

2. La estrategia de buffers grabados puede algunas veces evitar escribir actualizaciones a la base de datos, en conjunto. Por ejemplo, si muchas transacciones escriben el mismo dato, entonces dicho dato necesita ser escrito a la base de datos solo cuando se realice el punto de verificación ,y no cada vez que exista una actualización. De nuevo se reduce la carga a los discos de la base de datos.

2.3.2.3 Configurando intervalos para "dumps" de base de datos y puntos de verificación

La configuración de intervalos de database dumps, es una especie de negociación entre el tiempo de recuperación, seguida de una falla de uno o mas discos de la base de datos y un rendimiento en línea. Mientras más frecuente sea un 'dump', menos datos serán leídos desde un "database dump"

a la hora de una falla de disco. Muy pocas aplicaciones requieren "dumps" que ocurran con mayor frecuencia de una a dos veces diarias.

Algunos administradores del sistema dudan si vale la pena llevar a cabo "dumps" de todo. Los "dumps" ofrecen los siguientes beneficios:

- Si los discos de la base de datos fallan, y no existe un "dump", entonces el sistema perderá datos irremediablemente. Así, el "dump" sirve para asegurarse contra fallas de los discos de la base de datos. Se debe de notar, sin embargo, que una falla de bitácora puede causar perdida de datos si existe o no "dumps".

- Un "dump" puede ser usado para extraer datos con selecciones que no requieren información que esté completamente al día.

Un "dump" tiene dos costos:

- Este incrementa el tiempo de respuesta mientras ocurre.
- Este requiere espacio para almacenamiento.

Se debe de recordar que un punto de verificación forza a los datos que estén solo en la bitácora y en los "buffers" de la base de datos para ser llevados a disco. Así, un punto de verificación es solamente necesario cuando usa una estrategia de "buffers" salvados. Un punto de verificación tiene los siguientes dos beneficios:

- Reduce el tiempo y el espacio de las bitácoras, necesario para recuperarse cuando hay una falla en memoria principal (RAM), porque las actualizaciones grabadas ya están almacenadas en los discos de la base de datos.

- Reduce el espacio de las bitácoras necesario para recuperar de fallas de disco. Para recuperarse de tales fallas, la bitácora debe de cargar todas las actualizaciones desde el último "database dump".

La principal desventaja es que un punto de verificación es que degrada el rendimiento en línea (sin embargo mucho menos de lo que hace un "database dump"). Las aplicaciones que demandan una alta disponibilidad podrían hacer puntos de verificación cada 20 minutos o algo así. Si la aplicación demanda menos, entonces los punto de verificación deben ser menos frecuentes.

2.3.2.4 Reduciendo el tamaño de las transacciones de actualización

Una transacción que lleva a cabo muchas actualizaciones, sin un riguroso tiempo restringido de respuesta, es llamada una transacción en "batch". Si tal transacción es excesivamente larga, el "buffer" se puede llenar, y esto resultará en una operación fallada, y esto implica una operación de deshecha que será muy costosa.

Para poder resolver este problema, es recomendable dividir la transacción en pequeñas transacciones ("minibatch"). Cada mini-transacción actualiza una variable persistente que indica cuanto se a realizado. Por ejemplo, se supone que una tarea de actualización para una cuenta de clientes, es ordenada basada en un identificador de cliente "Cliente_id". Cada transacción "minibatch" puede actualizar un conjunto de registro de cuentas, y entonces poner el "cliente_id" de la última cuenta modificada hacia una variable especial de la base de datos llamada "ultimoclienteactualizado". Esta transacción "minibatch" se ejecutara serialmente. En caso de falla, el programa conocerá donde se quedó para continuar, modificando la cuenta del "ultimoclienteactualizado".

2.4 Consideraciones a nivel del sistema operativo

El sistema operativo lleva a cabo varias funciones que pueden ser de impacto significativo en el rendimiento de aplicaciones de bases de datos.

El plan de apuntadores (threads) de control (llamado procesos en algunos sistemas operativos) que son las unidades fundamentales de ejecución en un sistema de computación. El resultado aquí, incluye la cantidad de tiempo necesario para que el plan de un nuevo apuntador (el menor es el mejor); el espacio de tiempo de un apuntador de base de datos (cuanto podría durar); y si diferentes apuntadores podrían tener diferentes prioridades (los apuntadores de base de datos podrían todos correr con la misma prioridad para la mayoría de aplicaciones).

- El sistema operativo maneja mapeos de memoria física y virtual. El resultado aquí es que tan grande se escogerá la porción de memoria virtual que es compartida por todos los apuntadores de la base de datos, llamados "buffers" de base de datos y que tanta memoria principal RAM se deben dedicar para estos buffers.

- El sistema operativo configura el número de apuntadores de usuarios de control que pueden accesar la base de datos concurrentemente. La meta es tener suficientes para poder contener a la mayoría de usuarios tratando también de evitar desperdiciarlos.

- El sistema operativo maneja archivos. El resultado es incluir el distribuir archivos en diferentes dispositivos (necesarios para bases de datos largas); es bueno que los archivos puedan ser construidos con porciones contiguas de disco (esto ayuda al rendimiento en búsquedas);

- Si los archivos son leídos, pueden llevar a cabo una búsqueda anticipada (o "lookahead", el cual ayuda al rendimiento en búsquedas). Si se accesa una página de un archivo largo, toma en promedio, mas tiempo que accesar una página de un archivo pequeño (obviamente podría ser evitado); y si un proceso puede escribir páginas asíncronamente .

- El sistema operativo da información de tiempo. Esto puede ayudar a determinar si una aplicación está ocupando mucho procesador, o mucho disco E/S.

- El sistema operativo controla la comunicación entre los espacios de direcciones en un mismo lugar (por ejemplo, el mismo procesador u otro

procesador dentro de la misma memoria compartida entre multiprocesadores) y el espacio de direccionamiento en diferentes lugares. La principal característica en este punto es el rendimiento de mensajes. Si este es rápido, entonces el rendimiento de la base de datos será mucho mejor. De otra manera, el rendimiento de la base de datos no será adecuado. En esto existe poco que se pueda hacer como afinador, a excepción de tomar la decisión de mejorar el rendimiento con una estrategia de distribución.

2.4.1 El Planificador (Scheduler)

Cada sistema operativo planifica a los diferentes apuntadores de control, y de acuerdo a ciertos calculo que este realiza, también le da servicio a las aplicaciones de la base de datos. De manera que el afinador de la base de datos debe de tratar de minimizar la cantidad de tiempo desperdiciado en cambiar de contexto (esto es cambiar de un proceso a otro). Existen dos maneras de hacer esto:

1. Escoger un sistema operativo que tenga la característica de realizar el cambio de apuntadores fácilmente (que esta acción no sea una carga fuerte).

2. Minimizar el número de cambios (switches). Un cambio o "switch" es inevitable cuando cierta aplicación debe de realizar un requerimiento de E/S, pero existen también razones opcionales para cambios (switches) que pueden ser evitados.

Un segundo aspecto de la planificación son las prioridades de los apuntadores. A continuación, se expondrán dos decisiones de prioridad que pueden dañar el rendimiento de la base de datos.

Los procesos del sistema de base de datos corren con una prioridad baja, con respecto a las demás aplicaciones. Cuando estas aplicaciones consumen una gran cantidad de recursos, las aplicaciones de la base de datos tiene un rendimiento bajo.

Transacciones no son ejecutadas a la misma prioridad. Si se intenta dar a un apuntador una prioridad mayor a otras, podría resultar al contrario de los objetivos que se persigue, ya que puede haber conflicto con los mismos datos que se comparten con otras aplicaciones. Veamos el siguiente ejemplo:

2.4.1.1 Inversión de prioridad

Se supone que la transacción T_1 tiene la más alta prioridad, seguida de la transacción T_2 (la cual tiene la misma prioridad que otras transacciones), y por último la transacción T_3.

1. La transacción T_3 se ejecuta y obtiene un bloqueo (lock) en algún dato x.

2. La transacción T_1 inicia y requiere un lock en x, pero esta bloqueado por la transacción T_3.

3. La transacción T_2 inicia (sin accesar x). Otras transacciones de la misma prioridad continúan ejecutándose por un tiempo largo, evitando que se le de el respectivo quantum a T_3.

Indirectamente, T_2 y las otras transacciones de la misma prioridad evitan que T_1 continúe ejecutándose. A esto se le llama inversión de prioridad.

Algunos sistemas operativos manejan este problema, por el método llamado herencia de prioridad. La idea es que una vez que un apuntador actúa con un candado, su prioridad de planificación se incrementa al máximo nivel para evitar que cualquier otro apuntador espere por dicho candado.

Transacciones concurrentes en una aplicación de la base de datos, comparten datos en ciertas porciones, de la memoria virtual, conocida como los buffers de base de datos, buffers o algunas veces el 'cache' de base de datos. El propósito de los buffers es reducir el número de accesos físicos a almacenamiento secundario (generalmente discos).

El impacto de los buffers en el número de accesos físicos depende de tres diferentes parámetros:

Lecturas y escrituras lógicas: éstas son las páginas que el manejador de base de datos revisa vía comandos del sistema para lectura y escritura. Algunas de estas páginas son halladas en los buffers. Otros serán traducidas en lecturas y escrituras físicas.

Reemplazo de páginas del sistema manejador de base de datos (DBMS) Estos son los accesos físicos a almacenamiento secundario que ocurren cuando una página debe ser llevada hacia los buffers y no existen páginas libres. El afinador de la base de datos debe de asegurarse que este reemplazo ocurra con poca frecuencia.

Pagineo a nivel del sistema operativo: éstos son los accesos físicos a almacenamiento secundario (en algunos sistemas, un "swap" a disco) que ocurre cuando parte del espacio de buffer tiende a quedarse afuera de la RAM. El afinador debe asegurarse que tal pagineo ocurra en raras ocasiones.

Asumiendo que el "pagineo" y el reemplazo de páginas ocurren raramente, una pregunta importante es ¿cuántos accesos lógicos se convierten en accesos físicos?.

El "hit ratio" es definido en la siguiente ecuación:

$$\textit{hit ratio} = \frac{\textit{número de accesos lógicos - número de accesos físicos}}{\textit{número de accesos lógicos}}$$

Entonces el "hit ratio" es el número de páginas accesadas lógicamente y que fueron halladas en los buffers, dividido por el número total de páginas accesadas lógicamente.

El afinamiento de los parámetros que determinan el "hit ratio", es el tamaño de "buffers" de memoria. Para afinar el tamaño de los "buffers", es necesario correr una carga típica (carga normal) durante algunas horas. A continuación, se debe de revisar si el "hit ratio" esta muy bajo. Sistemas con suficiente RAM podrían tener un hit ratio del 90%. Sin embargo, para sistemas con bases de datos extremadamente largas, y con una gran cantidad de accesos de E/S aleatorias, esto es imposible. Por ejemplo, si el 30% de los accesos pueden tocar cualquier página en la base de datos, con igual demanda, y la base de datos contiene miles de billones de bytes, entonces se necesitaría un sistema con un gigabyte de tamaño de buffers para lograr tener un hit ratio del 70% o menos. Así, la mejor estrategia es incrementar el tamaño de los buffers hasta que el hit ratio tenga búsquedas rápidas, mientras se asegura que el reemplazo de páginas del DBMS, y el pagineo a nivel del sistema operativo sea lento. Algunos sistemas como ORACLE, ofrecen utilitarios que indicarán un estimado de lo que sucedería si el tamaño del buffer es bastante extenso.

Un segundo parámetro a afinar es el comprar RAM (random access memory) adicional. Existen dos factores por lo cual esto podría se tomado en cuenta:

- Incrementar el tamaño del "buffer" en nuestra RAM disponible podría causar un págineo significativo.

- Incrementar el tamaño del "buffer" podría mejorar el accesos a la base de datos significativamente. Esto es que más accesos E/S podrían encontrar páginas en el "buffer", y no en disco (de acuerdo al "hit ratio").

En ocasiones puede darse el caso en que aplicaciones *X* tengan mucha mas demanda con tiempos de repuesta altos, mientras que el resto de aplicaciones *Y* no sea así, y accesen diferentes datos; Esto podría dar la pauta para considerar dar cierta cantidad de buffer a las aplicaciones *X* y otra cantidad para las aplicaciones *Y*. (Esta es una facilidad del DBMS DB2 por ejemplo). Sin embargo, si todas las aplicaciones tiene el mismo nivel de requerimiento se debe usar un solo buffer.

2.4.2 Nivel de multiprogramación

Muchos administradores del sistema creen que si su sistema soporta mayor cantidad de usuarios concurrentes, es mejor. Es verdad que, incrementando el número de apuntadores de control (generalmente, en este caso, dentro del sistema servidor en donde esta la base de datos) ayuda a consumir ciclos desperdiciados del procesador en el que no ha sido utilizado. Sin embargo, los niveles de multiprogramación pueden bajar el rendimiento si sucede lo siguiente:

- La cantidad de RAM ocupada por los usuarios excede la cantidad real de memoria RAM del sistema, causando pagineo.

- Conflictos de candados pueden aparecer debido a alto número de transacciones concurrentes.

Realmente no existe un algoritmo, que indique que tan concurrente debe de ser un sistema, ya que en la mayoría de ocasiones, alguna regla es valida solo para ciertos sistemas, sin embargo muchas personas utilizan el método de pasos incrementáles:

- Inicie con un límite bajo en el número máximo de transacciones concurrentes permitidas.

- Incremente el límite en uno, y mida el rendimiento.

- Si el rendimiento mejora, entonces incremente el límite de nuevo. De otra manera, se ha alcanzado el rendimiento máximo local. (Estudios teóricos y prácticos, indican que el primer límite máximo local es probablemente el máximo absoluto).

Si la aplicación es estable, pero tiene diferentes perfiles (descripciones - profiles) de transacciones en diferentes horas del día, entonces su límite podría cambiar sus perfiles (profiles) descripciones de aplicación.

2.4.3 Archivos: acceso a discos y su utilización

Los sistemas de archivos o comúnmente llamados "filesystems" permiten a los usuarios crear, borrar, leer y escribir archivos (los cuales son secuencias de bytes o registros). Los principales parámetros de afinamiento para los sistemas de archivos son los siguientes:

- El tamaño de los "pedazos" de disco que serán localizados en un tiempo: Algunos sistemas de archivos los llaman "*extents" o extensiones* . Si múltiples selecciones tienden a buscar porciones en un archivo, entonces es bueno especificar el tamaño del "track" en extenciones para mejorar el rendimiento. Generalmente, el rendimiento de las escrituras a disco pueden ser mejoradas usando extensiones. Por ejemplo, los archivos históricos y

las bitácoras, serán beneficiados significativamente con la utilización de extents u otra técnica similar. Si los accesos a archivos son completamente aleatorios, definir pequeñas extensiones resulta mejor, ya que esto da una mejor utilización del espacio.

- El factor de uso en páginas de disco: algunos sistemas, como ORACLE, ofrecen a los usuarios el control de como llenar completamente una página, para aún permitir inserciones. Un factor de uso alto, permite que la mayoría de la página sea utilizada para inserciones. Si existen muchas actualizaciones de tuplas, entonces es recomendable mantener este factor bajo (70% o menos). De otra manera, es recomendable tener un factor de uso alto (90% o mayor) para mejorar el rendimiento de búsquedas en tablas.

- El número de páginas que pueden ser pre - buscadas: generalmente, el hecho de pre- buscar, es usado para selecciones que revisan archivos. A menos que las búsquedas en memoria principal RAM sean escasas, el numero de páginas a ser pre- buscadas debe de corresponder a largas porciones de un "track".

- El número de niveles que accesan una página de manera indirecta: Esto es una importante característica basada en sistemas UNIX. Debido a que las estructura de índices de los archivos UNIX, interponen más niveles de acceso de página indirecta, hacia el final de un archivo en vez de colocar páginas cerca del inicio del archivo, esto puede prolongar el acceso a dichas páginas. Para evitar esto, algunos manejadores de bases de datos utilizan lo que son "*raw slices* - "pedazos crudos" semejante al concepto de "raw devices".

2.5 Afinamiento de "hardware"

Todo sería mucho más fácil si el afinamiento de "hardware" pudiera hacerse independientemente del afinamiento de "software". Desafortunadamente, los dos están íntimamente relacionados. Sin embargo, afinadores expertos, inician con el análisis de utilización del procesador, actividad de disco E/S, y págineo, raramente paran allí.

La razón es simple: el hecho de que los recursos estén sobrecargados, no implica que se tenga que comprar más. Es mejora añadir índices para mejorar el rendimiento de "queries", por ejemplo.

Esta parte del capítulo, esta dirigida a determinar que se debe de hacer después de haber encontrado que se necesita mas "hardware". Generalmente, este tipo de afinamiento tiene un orden, primero memoria, luego discos, después procesador y por último el ancho de banda de la red.

2.5.1 Añadir memoria

En la mayoría de sistemas, la mejor manera de subir el rendimiento, es añadir memoria y luego incrementar el tamaño del espacio de "buffer". Esto reduce la carga de los discos, si se incrementa el "hit ratio", evitando que se incremente el "pagineo".

2.5.2 Añadir discos si están saturados

Si se añade una cantidad razonable de memoria, y esta no ayuda, entonces se recomienda comprar discos y controladores siempre y cuando se detecte que existe sobrecarga en los discos. Las siguientes, son algunas recomendaciones con respecto de este tema:

- Colocar las bitácoras en discos separados, asegurándose que las escrituras a estas bitácoras sean secuenciales.

- Usar espejo (mirroring) de discos como mecanismo para mejorar del "throughput". La mayoría de sistemas que incluyen "espejo" (mirroring) dividen la carga de lecturas a través de los discos espejos.

- Algunos manejadores de bases de datos como SYBASE, permiten escrituras en discos espejo, haciéndolo en paralelo, si los discos tienen controladores independientes, pero forzan que las escrituras se realicen serialmente, si son por medio del mismo controlador. Las escrituras paralelas son mucho más rápidas.

- Particionar tablas a través de diferentes discos: De esta manera, se puede asegurar que los propios discos manejen la carga de acceso. Dependiendo de la aplicación que se tenga, se puede escoger entre dos tipos diferentes de estrategias:

1. Aplicaciones con escrituras intensivas Pueden mover índices del tipo no agrupado (no-cluster) hacia otros discos, en vez que residan en uno solo. La razón es que cada modificación, tendrá que actualizar la mayoría de índices y tablas, de manera que se realice un balance entre los dos.

2. Aplicaciones con lecturas intensivas Esto permite balancear la lectura de los discos.

2.5.3 Añadir procesador

Si sistema con un solo procesador, contiene una razón del 85% de utilización, índica que el procesador esta sobrecargado (La razón para multiprocesadores es del 80%). Si se añade más memoria, y su software está correctamente afinado, entonces su sistema necesita más poder de procesamiento.

Es mas barato comprar mas procesadores, que un solo procesador mas grande. Dos procesadores de X MIPS (millones de instrucciones por segundo) puede ser significativamente menos efectivo para una aplicación que un procesador de 2X MIPS, si estos deben de comunicarse entre sí. Esta comunicación produce sobrecarga, y a dicha sobrecarga se le denomina *penalización de MIPS.*

Cuando se usa mas de un procesador, se deben observar las siguientes reglas de particionamiento:

Si es posible, descargar las aplicaciones que no son de la base de datos hacia un procesador.

Razón: Estas aplicaciones no tendrán que comunicarse con las de la base de datos.

Considerar hacer "dumps" de la base de datos

Si su sistema tiene muchas lecturas prolongadas por selecciones, entonces se debe de considerar dos sistemas redundantes. Uno estará dedicado a transacciones prolongadas de solo lectura, específicamente de los datos recolectados en el último dump de base de datos. La otra, llevará a cabo transacciones en línea. Esto eliminará comunicación en red entre los dos sistemas, excepto en el momento de realizar el "dump" de base de datos.

Permitir que las transacciones de solo - lectura se dividan en servidores locales, antes de permitir transacciones de actualización se dividan en varias localidades.

Razón Algunos sistemas, no garantizarán la atomicidad de transacciones si ocurren actualizaciones en diferentes localidades. Además, las transacciones de solo - lectura incurren en menor sobrecarga cuando están divididos en localidades que las transacciones de actualización.

Examinar la forma en que los datos serán compartidos en la aplicación para seleccionar entre las siguientes tres arquitecturas:

- Tener varios procesadores, pero una sola memoria compartida y un conjunto de discos. Tal arquitectura es llamada "*tightly-coupled*" y es típica

para multiprocesadores basados en bus. Es una buena arquitectura para aplicaciones en las cuales toda la data se compartirá (por ejemplo, aplicaciones de contabilidad que estarán centralizadas).

- Tener varios procesadores, cada uno con su propia memoria y sus discos. Tal arquitectura es llamada "*shared-nothing*" y es usada en aplicaciones que requieren una distribución geográfica (por ejemplo, un banco multinacional) o una aplicaciones que puede ser dividida en sub-aplicaciones independientes.

- Tener varios procesadores, donde cada uno tiene su propia memoria, pero los procesadores comparten los discos. Esta arquitectura es llamada "*shared-disk*" y podría ser utilizada, cuando una arquitectura "taghtly-coupled" es ideal, pero las aplicaciones tienen mejor rendimiento al tener un solo lugar de acceso.

3. Aspectos generales de afinamiento en diferentes manejadores de bases de datos relacionales

3.1 Facilidades de afinamiento

Generalmente, existen tres áreas para ser "afinadas" las cuales permiten mejorar el rendimiento:

- El sistema operativo,
- el servidor de la base de datos (Kernel) y
- la base de datos por si misma (aplicación).

Obviamente, se limitarán las respectivas implicaciones a las últimas dos, haciendo énfasis a la segunda.

El afinamiento del servidor de base de datos implica ajustar el software de base de datos, a la hora de realizar la instalación y configuración del mismo, de manera que dicha interface con el sistema operativo sea mejor. La meta principal es: *la óptima utilización de los recursos de la máquina como del sistema operativo.* Afinamiento de la base de datos en sí, implica ajustar la implementación de los objetos en la base de datos para mejorar el acceso a las aplicaciones. A continuación, se presentaran las facilidades de afinamiento que proveen algunos de los mas populares manejadores de base de datos relacionales, tales como CA-Open-Ingres 1.2, Informix OnLine 7.2, 6.0 y 5.0, Borland InterBase 4.0, Microsoft SQL Server 6.5, Oracle7 release 7.3, Centura

SQLBase 6.0 y Sybase SQL Server 10 y 11. Estos serán aspectos genéricos, luego de esto, se han seleccionado de estos, tres, que son los que (a nivel de bases de datos relacionales) tienen mayor demanda en nuestro mercado y mayor base de empresas instaladas, de los cuales se tomaran aspectos mas profundos (donde aplique) como afinamiento de áreas temporales, áreas de deshecho (rollback segments), afinamiento de memoria, de CPU, de contención de disco, áreas de ordenamiento, Etc.

3.2 CA-OpenIngres

La mayoría de aspectos concebibles en el manejador de base de datos de CA-OpenIngres pueden ser afinables. Para el servidor de este manejador de base de datos en sí mismo, se puede especificar, cuanta memoria compartida debe seleccionada para sus respectivos caché de buffers (tamaño), cuanta memoria debe ser reservada para realizar optimización de selecciones (queries), cuanto tiempo de CPU será disponible para cada sesión, el número de escrituras de apuntadores (el cual escribe buffers de memoria a los archivos de bitácora), y si alguna sesión será activada, etc.
Hay muchos parámetros más que se pueden especificar y que son derivados de acuerdo a ciertos parámetros: por ejemplo, para conectarse a los respectivos subsistemas, se debe especificar si el doble de conexión esta habilitada, donde los archivos de bitácora están situados (en un raw device o en un sistema de archivos regular), que tan grande deben ser los archivos de bitácora, y que tan a menudo y de que tamaño los bloques del buffer de memoria serán escritos a los archivos de bitácora.

Para el subsistema de candados, se puede especificar cuantos candados se esperan por transacción, cuando los candados serán escalados de nivel de página a nivel de tabla, Etc. La configuración de los procesos servidores del manejador de base de datos y sus variados subsistemas pueden tener una gran influencia en el rendimiento de sistema en general; Por ejemplo, aplicaciones con consultas intensivas tales como WareHouses, las cuales se llevan a cabo extensas lecturas de buffers conviene tener buffers de bitácora mas grandes, mientras aplicaciones que son actualizadas en línea (OLTP - On Line Transaction Process) se pueden realizar mejor con mayor número de buffers de bitácora, pero mas pequeños, y mas escrituras de apuntadores.

Para cada base de datos, se puede especificar donde están los archivos de datos, los archivos de puntos de chequeo, los archivos de trabajo temporal. En general, una base de datos tiene mejor rendimiento si sus datos, bitácoras, y archivos de punto de verificación son distribuidos en diferentes discos, dando así un alto grado de paralelismo de accesos de E/S (disminución de la contención de disco).

Para cada tabla de la base de datos, se puede especificar sobre cual localización de la base de datos esta distribuida, el número de páginas de datos reservadas con anterioridad, y cuando se llena, el número de páginas en la cual, la tabla se extenderá. Se puede especificar la estructura de almacenamiento de cada tabla como B-tree, hash, heap o ISAM. Para cada estructura de almacenamiento, se puede especificar el tipo de compresión (si existe).

El CA-open Ingres es extremadamente afinable, desde el servidor ("kernel" del manejador de base de datos) hasta la implementación de la base de datos, incluso a nivel de implementación de tablas e índices. Las opciones de implementación por defecto usadas por CA-Open Ingres generalmente da un rendimiento aceptable para base de datos y tablas consideradas de tamaño mediano.

Desafortunadamente, debido a la gran cantidad de elementos que se pueden afinar, solamente para consultores expertos, es fácil obtener un sistema rápido. Para consultores de rendimiento y administradores de base de datos, quienes no son muy expertos, es fácil especificar una opción errónea y obtener como resultado un rendimiento pobre.

3.3 Informix

El servidor dinámico de Informix OnLine es un servidor de base de datos multithreaded que está basado en una arquitectura dinámica escalable (DSA - de Dinamic Scalability Architecture). DSA usa un espacio dinámico para los procesos del servidor de la base de datos llamados 'Procesadores virtuales' a lo largo de múltiples concurrentes apuntadores (MCT - de Multiple concurrent threads) para requerimientos de servicio de clientes en paralelo. Esta arquitectura es altamente escalable. Cada procesador virtual pertenece a una clase de procesadores virtuales dedicados a tareas especificas. La programación del servidor de la base de datos que apunta (threads) y determina la prioridad de cada apuntador. Informix contiene el llamado On line Extended Parallel Server, el cual extiende el DSA hacia la arquitectura no compartir (share - nothing), incluyendo agrupamientos (clusters) de sistemas con procesamiento simétrico (SMP) y de sistemas de procesamiento

masivamente paralelo (MPP)- El informix Online permite usar el paralelismo con su opción parallel data query (PDQ)
El PDQ implementa el paralelismo vertical, según el cual el servidor OnLine trabaja en mas de un aspecto en el tiempo. PDQ puede llevar a cabo búsquedas de tablas, uniones, ordenamientos, operaciones de agregación, y operaciones de agrupamiento en paralelo.

La configuración de memoria del servidor dinámico OnLine, permite configurarle la memoria, el CPU, y utilización de disco. Se puede especificar un tamaño de memoria compartida por defecto y el tamaño por el cual será incrementado cuando sea requerido; también se puede reservar o eliminar la reservación de memoria compartida. Se puede configurar cuando los "buffers" de datos deben ser usados en las páginas de la base de datos en memoria, así como también el tamaño de los tres buffers lógicos de la bitácora. Se pueden especificar el número de procesadores virtuales a la hora de que el servidor se levante, esto es el numero que puede ser incrementado, y en algunos casos decrementado, mientras el sistema esta corriendo.

También se puede especificar la afinidad de procesador, en el cual puede asignarle un proceso virtual a un procesador especifico. Si se usa la opción PDQ, se puede especificar el porcentaje de CPU y memoria que puede ocupar una selección. Esta característica puede ser usado para limitar el impacto de selecciones con intensivo uso de CPU en la velocidad, transacciones pequeñas OLTP. En adición, se puede configurar cuantas lecturas de cabezas a apuntadores que trabajen de manera asincrónica, apuntadores que deben llevar a cabo búsquedas desde una tabla secuencial o una búsqueda de índice. Se puede especificar el máximo número de candados

que pueden ser cargados en algún objeto en el tiempo, arriba del límite de 8,999,999.

Se puede dividir una tabla o un índice en uno o más discos o en porciones de un disco. Una tabla es asignada a un llamado "dbspace" compuesto de uno o mas "chuncks" que, habilitado, corresponde a toda o una parte de una partición de disco. Las tablas e indices pueden también estar fragmentadas horizontalmente a través de diferentes discos, o por medio de una expresión, o por medio de la política "round robin". El grado de fragmentacion que se escoge puede mejorar el rendimiento en el soporte de queries, y así se puede reducir la contención en disco. Se puede reservar texto largo o columnas de bytes llamadas "blockspace" en un dispositivo separado. Sin embargo, no se puede influenciar las estructuras de almacenamiento usadas para tablas e índices. Informix usa estructuras B-tree+ para sus índices, y estas estructuras pueden estar encerradas en grupos (clusters) o no. Informix puede almacenar sus datos en sistemas de archivos o en particiones de crudas (raw disk).

El Informix usa el optimizador basado en costo, el cual puede tomar requerimientos estimados de acceso de E/S (algo así como estadísticas), estimando la cantidad de procesamiento de CPU, los recursos requeridos para búsquedas, y las estadísticas de las tablas e índices hacia una cuenta. Este construye todos los posibles planes de ejecución (query plans) con todas las posibles uniones a nivel de selecciones (joins) usando una estrategia de búsqueda "bottom-up breadth first", el cual selecciona el plan de ejecución con el costo estimado mas bajo. Se puede influenciar el optimizador de Informix

como una extensión: a parte de especificar las estadísticas que son colectadas para cada tabla, se puede especificar el método preferido para llevar a cabo las operaciones de uniones (joins), los nombrados ciclos de uniones anidadas o "nested-join loops" o los uniones de búsquedas completas a tablas (table-scan joins).

El servidor dinámico de Informix OnLine es muy configurable; en particular, se pueden hacer cambios en la utilización de memoria.
La reservación de datos en los dispositivos de almacenamiento es también bastante configurable.

3.4 Borland InterBase

Existen pocos aspectos que pueden ser configurados para el servidor de este manejador de InterBase: El rango y páginas de memoria dedicada para el servidor, el nivel de prioridad de los procesos servidores con respecto a otros procesos que están corriendo en la misma máquina, el tamaño del cache de la base de datos, y el tamaño de mapeo de clientes. El tamaño del cache de la base de datos, define el número de páginas de memoria reservadas para cada una de las bases de datos activas. Un cache de base de datos que es largo, mejora el rendimiento si toda la actividad de la base de datos puede manejarse en memoria física en vez de que haga swap a disco. Muchos cache de base de datos que sean grandes, sin embargo pueden causar swapeo excesivo. El tamaño de mapeo de cliente define el tamaño del buffer de comunicación usado para cada cliente local de interbase. Un tamaño de mapeo de cliente largo puede mejorar el rendimiento si las aplicaciones usan varios bloques.

Una base de datos interbase consiste de un archivo simple por localización de disco (o directorio) en el cual todos los objetos de la base de datos están almacenados. Interbase inicia llenando la primera localización, después sigue con la segunda, etc. Para una base de datos InterBase, se puede especificar el tamaño de página para ser usado en la base de datos, la localización de disco donde los archivos de la base de datos debe ser colocada, y el tamaño (en paginas) de cada archivo. Tamaños de pagina largos pueden mejorar el rendimiento de base de datos grande, porque los índices tendrán poca profundidad (según el árbol de búsqueda o el plan de trabajo) y tuplas largas pueden ser almacenadas en páginas simples. Sin embargo, de una pequeña base de datos del tipo OLTP, tamaños de paginas pequeñas pueden ser más apropiadas porque menor cantidad de datos son retornados de la base de datos cada vez.

También se puede configurar un "Intervalo de barrido" (sweep interval) y forzar las escrituras de la base de datos por cada base de datos. El "sweep interval", si es habilitado, especifica que tan frecuentemente InterBase podría buscar en la base de datos para remover viejas versiones de tuplas no usadas en las tablas. (Nótese que interbase usa un control de esquema concurrente basada en una versión optimizada, en vez de un esquema de control concurrente basado en bloqueo (locks) que es utilizado en la mayoría de productos).

Forzar las escrituras de la base de datos, asegura que los cambios en los datos son escritos inmediatamente a disco, esto provee una alta confianza a fallas, sin embargo degrada en rendimiento.

Para una tabla InterBase, se pueden solamente definir si los datos son almacenados en una tabla interna de InterBase, o en un archivo externo en el sistema operativo. Una tabla en un archivo del sistema operativo no tiene todos los controles de concurrencia, y backup de una tabla interna de InterBase, pero ofrece una conveniente manera heredar datos o importar nuevos datos hacia la base de datos. La única opción de afinamiento para un índice es si este esta almacenado en orden ascendente o descendente.

InterBase es el lado opuesto en la escala de "Afinamiento" de CA-OpenIngres. Esta es como una caja negra.

3.5 Microsoft SQL server

Una instalación de SQL Server puede ser afinada en cierto número de niveles. Durante la instalación, se especifica un dispositivo de datos maestro, el cual es donde la base de datos maestra es almacenada. Antes de que se pueda crear las nuevas bases de datos por primera vez, se debe de añadir al menos un nuevo dispositivo para estas bases de datos. Se puede usar estos nuevos dispositivos para las bases de datos o para las bitácoras de transacciones (o para ambos, sin embargo esto no es recomendado). Un dispositivo de datos o de bitácora, es simplemente un archivo enorme en el sistema. De esta manera se puede controlar donde las bases de datos y la bitácora están colocados - preferiblemente en discos físicamente separados. Cada base de datos se puede dividir en varios dispositivos de datos.

Microsoft SQL Server no es muy flexible con respecto al número de servidores y como ellos están inter-relacionados, porque su programación esta controlada por el sistema operativo Windows NT. Sin embargo, Microsoft SQL Server aun provee un impresionante número de parámetros afinables del servidor. Algunos de estos parámetros pueden ser cambiados mientras el servidor esta corriendo, pero otros solo pueden tomar efecto cuando el servidor es re-iniciado. Los parámetros configurables del servidor incluyen:

- Número de conexiones permitidas
- Si las actualizaciones son permitidas
- El tamaño del buffers para la bitácora de transacciones
- El número de apuntadores usados para operaciones de carga.
- Tamaño por defecto de la base de datos.
- El factor de llenado por defecto para tablas e índices.
- El número máximo o porcentaje de candados por pagina antes de dar lugar a un escalamiento de candados.
- El intervalo entre la escritura de buffers hacia las bitácoras de transacciones.
- El número de accesos E/S asincronas que serán requeridas desde el sistema operativo.
- El número máximo de apuntadores en el servidor
- La memoria reservada para el servidor
- El número esperado de cache para los procedimientos almacenados
- Si se usa área temporal, debe ser colocado en memoria para búsqueda rápida.

Una tabla o índice puede ser localizado en un segmento particular. Un segmento puede dividirse en múltiples dispositivos de datos, definidos para la base de datos. Para reservar tablas o índices a segmentos, se pueden distribuir óptimamente los accesos de E/S a la base de datos sobre un número de discos. Se puede especificar el factor de llenado de cada índice para dejar los datos que llena las paginas del índice dispersa. No se puede especificar las estructuras de almacenamiento de una tabla o un índice, pero un índice puede ser agrupado (clustered) con su tabla (en este caso el índice se vuelve la principal estructura de almacenamiento para dicha tabla, y la tabla será removida hacia un segmento especificado para el índice). Una tabla solo puede tener un índice agrupado (clustered) simple.

3.6 Oracle

Oracle release 7.3 es también bastante afinable, desde "instancias individuales" de los servidores de Oracle, (base de datos) hasta sus tablas e índices individuales. Para sistemas paralelamente masivos o agrupados (clustered), que soportan discos compartidos, se puede usar "Oracle Parallel Server", el cual permite que múltiples máquinas tengan instancias separadas, todas accesando a la misma base de datos física. Alternativamente se puede activar el "Parallel Query Option" de Oracle server, el cual permite procesar múltiples trabajos simultáneamente para procesar una simple sentencia SQL. El servidor de Oracle puede procesar ordenamientos, uniones (sorts), barridos completos de tabla (table scans), operación de poblado de tablas, y operaciones de creación de índices en paralelo.

Afinar la instancia de Oracle server implica colocar la utilización de memoria, actividad de E/S, contención, y otros factores tales como áreas de ordenamiento, listas libres, puntos de chequeo, etc. La utilización de la memoria del servidor de Oracle, puede ser ajustada para especificar la cantidad de memoria reservada para el "parseo" de las áreas de SQL, el "shared pool", y el "buffer cache". La sobrecarga del acceso de E/S del servidor de Oracle, puede reducir o paralelizar colocando los archivos de datos o archivos de bitácora en diferentes discos, dividiendo las tablas sobre diferentes discos, y separando las tablas e índices en diferentes discos, etc. Estas acciones son hechas especificando la reservación los llamados tablespaces para determinar las áreas de disco y entonces reservar los objetos de la base de datos en los apropiados tablespaces. La contención ocurre cuando múltiples procesos requieren el mismo recurso simultáneamente.

La contención puede reducirse añadiendo procesos despachadores (dispatchers) y/o procesos servidores compartidos, reduciendo el número de servidores de selecciones paralelas, o ajustando el uso de los buffers de bitácora (redo log buffers).

Para cada tabla, se puede especificar el porcentaje de espacio libre dejado para futuras actualizaciones, el mínimo porcentaje de espacio libre que se utiliza en cada bloque de datos, el tablespace donde se puede reservar espacio para algún objeto, las características de almacenamiento (el tamaño inicial y su patrón de crecimiento), su grado de paralelismo, etc. Para cada índice, se puede especificar el porcentaje de espacio libre dejado para futuras actualizaciones, el "tablespace" donde puede ser colocado, sus características de almacenamiento y su grado de paralelismo. Tablas relacionadas puede ser almacenadas juntas como un cluster. Esta configuración puede resultar en un rendimiento excelente cuando las dos tablas son unidas (en un join) a menudo,

pero esto no es ventajoso cuando las dos tablas son accesadas frecuentemente de manera separada. Los "clusters" también pueden ser almacenados en estructuras de "hash", el cual da un buen rendimiento principalmente en tablas que son estables (no son muy volátiles en inserciones o actualizaciones).

En Oracle, se puede seleccionar el tipo de optimizador que se desea utilizar, las dos opciones son el de regla (que es antiguo) y el de costo (que es el mas reciente). El optimizador basado en regla escoge un plan de ejecución basado en caminos de accesos disponibles (paths) y el rango (ranking) de estos caminos de accesos, (que están colocados en una tabla en sus respectivos manuales) de esta manera uno puede analizar un query, y modificarlo para disminuir el costo de acuerdo al ranking menos caro (en tiempo). El optimizador basado costo, selecciona el plan e ejecución basado tanto en caminos de accesos disponibles, como en estadísticas que están localizadas en el diccionario de datos para tablas, grupos (clusters), e índices. También se pueden utilizar los llamados "hints" (o sugerencia de optimización) para sus queries. La meta del optimizador basado en costo es incrementar el throughput (ver glosario), esto es reducir los recursos de uso necesarios para procesar todas las tuplas que están siendo accesadas.

3.7 SQL Base

Varios aspectos del servidor de SQLBase de Centura pueden ser configurados, incluyendo el número de páginas en la base de datos y el cache de ordenamientos. Ambos aspectos pueden afectar el rendimiento del servidor. Generalmente, un "cache" grande, puede reducir el número de

accesos de E/S, e incrementar el throughput. También se puede configurar un período de descanso (time-out) para los requerimientos de candados, los directorios para la base de datos SQL Base, bitácora, y archivos temporales. Se puede configurar además el tipo de accesos de E/S, ya que éstas pueden ser directas o a base de buffers, y también el tipo de optimizador que se desea utilizar.

Para la base de datos SQL Base, se puede especificar los directorios de la base de datos o, alternativamente, el almacenamiento en grupo (clusters) en el cual la base de datos puede estar almacenada. Un grupo de almacenamiento, es una lista nombrada que contiene áreas de la base de datos, y un área de la base de datos es un área nombrada de disco (tal como un directorio). Además se puede especificar el número de páginas, si la base de datos necesita ser extendida cuando esta está llena, el tamaño de los archivos de bitácora, el tamaño por el cual los archivos de bitácora pueden ser extendidos cuando es necesario, el número de archivos de bitácora en el cual una transacción activa puede dispersarse, la frecuencia en que los puntos de verificación son realizados, Etc.

Para cada tabla, solo se puede especificar el porcentaje de espacio libre que puede ser dejado en cada página de tabla, cuando la primera es llenada. Para cada índice, se puede especificar si tiene una estructura B-tree (que es por defecto) o una estructura de hash (debe ser especificada). Nótese que no se puede actualizar un valor llave en un índice con estructura de hash. Se puede especificar si los valores de los índices son almacenados ascendente o descendentemente en orden, el porcentaje de espacio libre a ser dejado en un

índice cuando este es llenado inicialmente, y el tamaño esperado (lógico o físico) de los índices. Si se coloca el tamaño del índice muy pequeño, esto puede afectar el rendimiento de manera adversa debido a que puede existir numerosos desbordamientos a nivel de páginas.

SQL Base da varias opciones de configuración para el servidor de su manejador de base de datos, especialmente para bases de datos individuales, sin embargo, esta selección tiende a ser limitada. La selección entre índices del tipo B-tree y hash podría se suficiente, pero la estructura de B-tree por defecto también parece ser restringida. Una restricción bastante fuerte es que no se pueden colocar tablas e índices en diferentes áreas de almacenamiento físico, esto implica que no se puede eliminar la contención de disco.

3.8 Sybase

El servidor SQL de Sybase es extremadamente afinable, desde los servidores individuales, hasta índices y tablas a nivel individual. De hecho, de las mas nuevas y significativas características de Sybase SQL Server 11 son que mejoran el rendimiento sobre Sybase 10.

Sybase es completamente multithreades, tiene una buena escalabilidad de arquitecturas de un solo procesador hacia sistemas SMP. Sybase SQL Server puede ser configurado para ejecutar múltiples máquinas (procesos que realizan rutinas elementales y altamente repetitivas o engines) en un sistema SMP, y específicamente la máquina de red simétrica de sybase puede distribuir conexiones iguales de cliente a través de estas máquinas. Las características

avanzadas del manejador de Sybase, incluyen un manejo lógico de memoria que puede mejorar bastante el rendimiento del sistema. La característica de balanceo de carga de la arquitectura de servidores virtuales de Sybase puede ser usado para afinar y maximizar el rendimiento sobre múltiples CPU's. El kernel de Sybase puede también ser configurado para opciones MMP (Procesamiento Masivamente Paralelo).

Varios recursos pueden ser compartidos, por ejemplo: la memoria, candados, y CPU. A través del manejo de memoria lógica, se puede configurar un número de caches los cuales pueden ser nombrados, estos pueden ser usados para datos y para procedimientos almacenados. Cada cache puede ser configurado con varios grupos de "buffers" de diferentes tamaños, utilizando las estrategias de págineo 'el menos recientemente usado' o la 'búsqueda y descartación mas reciente'. Se pueden entonces atar tablas específicas e índices a caches específicos. Los índices pueden ser atados a diferentes cache, distintos al de su tabla.

Cada usuario puede tener su propio cache para bitácora privada, de manera que los apuntadores no tengan que esperar, debido a que otro buffer este accesando dicho archivo de bitácora.

Se puede especificar que el tipo de candado sea escalable, de manera se defina un candado a nivel de página, y sea escalado a nivel de tabla. Los candados a nivel de tabla son mas eficientes, pero resultan mucho mas costosos a nivel de rendimiento.

Cuando se utilizan estas características, se puede configurar el SQL Server en ambientes de trabajo mixto, por ejemplo, ejecución en batch, OLTP, aplicaciones DSS. Debido a que Sybase SQL Server lee su configuración desde un archivo, se puede cambiar o alternar las características de los servidores fácilmente, ya que se pueden usar diferentes archivos de configuración. La mayoría de opciones de configuración descritas aquí, pueden ser especificadas por sesión, por selección individual, o por objeto de la base de datos.

Una base de datos Sybase puede ser separada en múltiples dispositivos físicos de almacenamiento. Cada base de datos tiene su propia bitácora transaccional, que puede ser configurada con sus propias características. La bitácora puede ser colocada en dispositivos separados de los datos normales (tablas e índices), no solo para mejorar el rendimiento, sino que también para mejorar la fiabilidad.

Cuando una tabla de Sybase es creada sin índices, es almacenada con una estructura de montículo (heap). Se puede crear solamente un índice del tipo agrupamiento (cluster), y muchos índices normales (no cluster) para cada tabla. Cuando se crea un índice del tipo agrupamiento (cluster), los datos de la tabla están físicamente almacenados en el orden de las llaves del índice. Sybase solamente soporta la estructura B-tree para sus índices normales. Por cada índice, se puede especificar un factor de llenado (población) y el número de tuplas que deben ser almacenadas por página.

Se pueden separar los índices de las tablas en dispositivos físicos diferentes, de hecho, se puede colocar páginas de índices para una tabla sin

agrupamiento (cluster) en dispositivos físicos separados de su misma data. También se puede particionar una tabla, la cual puede crear múltiples cadenas de páginas (page chains). El particionamiento reduce el acceso a las últimas páginas de la tabla, y permite el paralelismo de acceso de E/S para operaciones de tablas voluminosas.

Sybase tiene un optimizador de queries basado en costo. Este parsea cada sentencia SQL y determina un plan de ejecución optimo para la selección, de acuerdo a sus estrategias de cache, tamaño del cache, tamaño de bloques de E/S, los índices disponibles, y las estadísticas conservadas en la tabla y sus índices. El plan de la selección, es una secuencia de pasos requerido para llevar a cabo dicha selección, tal como búsqueda completa en la tabla (table scan), acceso a índices, y acceso a tablas de trabajo temporales. Se puede influenciar el plan de la selección especificando el orden en el cual se coloca la unión (join) de las tablas, el número de tablas que será evaluado simultáneamente en la unión, los índices que deben ser usados para accesar la tabla, el tamaño del bloque de E/S, la estrategia de caché. Sin embargo, estos pasos solo deben ser llevados a cabo por un DBA experto, ya que el uso inapropiado de estas técnicas podrían causar un rendimiento pobre.

Sybase es afinable con respecto a los datos y la localización del índice, pero no con respecto a las estructuras de almacenamiento usadas para las tablas e índices.

Como se puede ver, existe una amplia gama de aprovechamiento en el afinamiento de las base de datos, desde el cerrado afinamiento de InterBase, hasta los manejadores de base de datos extensamente afinables como CA-OpenIngres, Informix, Sybase y Oracle. Se puede observar que, mientras mas

afinable es un RDBMS, mejores resultados se pueden obtener, pero también implica que se necesita a un DBA mas experimentado, ya que podría incurrir a aplicar mal algún principio de afinamiento, y provocar un bajo rendimiento. Sin embargo, como se comento en el capítulo uno, no se puede hacer mucho afinando el RDBMS si existe detrás de el, un mal diseño. El factor mas importante para obtener un rendimiento efectivo, es una base de datos eficiente, y un buen diseño de la aplicación.

4. SYBASE

4.1 Manejo de memoria

- Relación entre memoria y rendimiento.
- Estimación de como configurar el tamaño de la memoria
- Examinar y cambiar la configuración del servidor el cual afecta el uso de memoria.

4.1.1 Tamaño de la memoria (posibles fallas)

Si el tamaño de la memoria es muy pequeña, puede fallar en lo siguiente:

- No se conecta con un nuevo cliente.
- Falla al ejecutar un procedimiento almacenado
- Las selecciones simples pueden ser ejecutadas, pero las complejas fallan
- Actualizaciones/inserciones/borrados fallan por falta de candados.

Si es muy grande puede haber "swap", o tener problemas a la hora de levantar. Pueden fallar las páginas a nivel del sistema operativo.

Cuanta más memoria se le dé, mucho mejor. El límite es el punto en el que empiezan a fallar páginas o no puede reservar el tamaño de memoria compartida.

4.1.2 "Caché" para el SQL Server

Sobre el "caché" del manejador de base de datos de Sybase (El SQL Server) existen los siguientes elementos que pueden ser cargados:

Datos para tablas e índices en actualizaciones, borrados, e inserciones.
Procedimientos procedimientos son compilados cuando se cargan a cache. Son re - ejecutados en cache.

Aproximadamente, para este manejador de base de datos se estima un accesos de E/S físicas de disco de 18 milisegundos, y de 2 milisegundos si son lógicos.

4.1.3 ¿Qué tanta memoria se le puede dar a SQL server?

A la hora de considerar el tamaño de la memoria del "kernel" de Sybase, se debe de incluir lo siguiente:

- Del tamaño total de la memoria principal (RAM) en la máquina, se debe de restarle el utilizado por el sistema operativo, y por otros programas como el respaldo del servidor ("backup manager").
- El resto es para el SQL server.

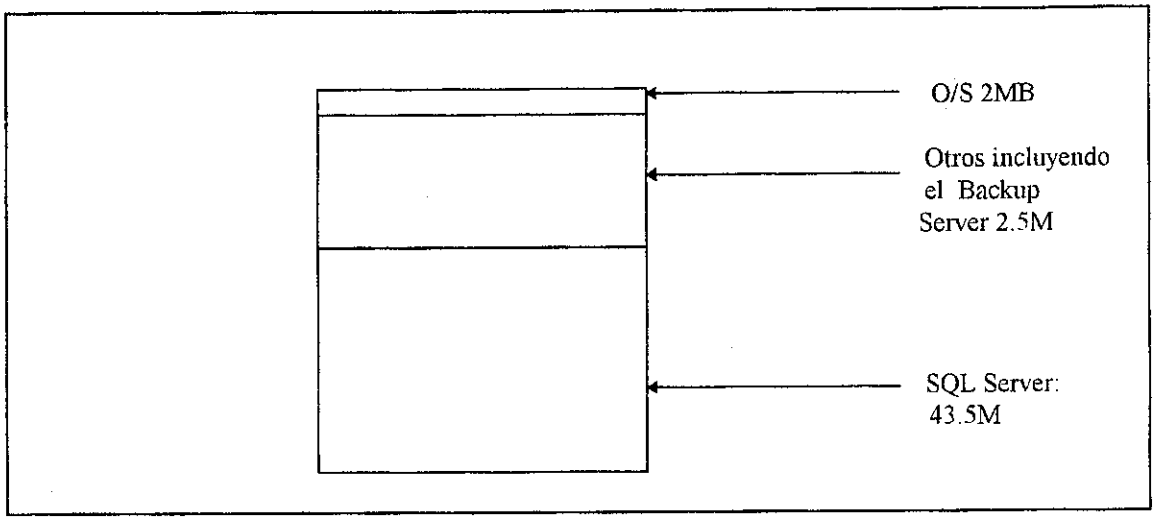


Figura 7. Distribución de la memoria en SQL server de Sybase

4.1.4 Composición de la memoria

El código ejecutable de SQL Server tiene la característica de ser fijo, de esta manera, por el utilitario llamado "dbcc memusage", se puede examinar el tamaño de los ejecutables de este servidor. Sin embargo, el total de memoria a configurar, debe ser suficiente para lo siguiente:

- Las necesidades de memoria estática de SQL server (Sobrecarga del "Kernel", espacio de la pila ("stack") de usuario).
- Caché para procedimientos y para datos.
- Buffers de red para cada conexión de usuario, reservada con el tamaño por defecto del manejador de la red ("default network manager size" que se explicara luego).
- Buffers usados para extender buffers de acceso a E/S se explicara luego).

4.1.5 ¿Cuánta memoria puede usar el SQL server?

Para maximizar la cantidad de memoria en su sistema:

- Reste la memoria requerida del sistema operativo del total de memoria física.
- Reste la memoria requerida del servidor de respaldo ("backup server").
- Reste los requerimientos de memoria para el uso de los otros sistemas tal como x-windows.
- Reste la memoria reservada de la memoria para conexiones de red ("aditional netmem").

Se debe hacer notar que la memoria adicional de red, esta realmente localizada fuera de la memoria normal del SQL server, pero aún esta conectada a dicho kernel.

4.1.6 Uso de memoria con el dbcc

Su salida se divide en tres partes:

- código
- caché de datos
- caché de procedimientos

Por ejemplo se presentan los siguientes comandos del "dbcc" y se indican las respectrivas salidas:

```
dbcc traceon (3604) (direcciona la salida a screen)
dbcc memusage
dbcc traceoff (3604)  (manda la salida de regreso al log).
```

Figura 8. Algunas instrucciones del comando DBCC de Sybase

4.1.7 Reconfigurando la memoria

Para poder reconfigurar la memoria del kernel de sybase, se debe usar la opción llamada "Buildmaster".

Si se usa esta opción, la configuración de bloques que contiene todo el sistema de parámetros de inicialización, se sobre - escribirá con los valores por defecto. Se debe de asegurar que se tenga una copia antes de re-inicializar, ya que se corre el riesgo de perder configuraciones anteriores.

Una recomendación es que, no use esta opción a menos que SQL server no levante.

Esto implica que el procedimiento es modificar el kernel de Sybase, y por el método de prueba y error obtener un total de memoria del kernel adecuado, pero por si algún error ocurriera y por lo tanto la base de datos no levanta, entonces dicha memoria será inicializada con los valores por defecto con el utilitario "Buildmaster"

Para obtener el tamaño Total de SQL server:

```
sp_configure "Memory",8193   (Pagina usual de 2k)
go
reconfigure
go
```

Figura 9. Instrucciones para obtener el tamaño total de memoria de SQL Server

y reinicia SQL server.

Si SQL server no levanta, use el "buildmaster" para re-inicializar a valores por defecto así:

```
buidmaster -r
```

Figura 10. Comando para reconfiguración del kernel de SQL Server

4.1.8 La mayoría de opciones afectan el tamaño de las estructuras del servidor

Algo que podría sucedes es el colocarle mas espacio para el kernel, y a las estructuras del servidor, y por lo tanto se le esta restando espacio al cache de datos y de procedimientos, lo cual incurriría en que la mayoría de accesos los trataría de hacer en disco, lo cual producirá efectos no deseados en el rendimiento, por ejemplo:

User connections----------50 bytes c/u Devices----------512 bytes c/u Open DB ----------7400 bytes c/u Open objects ----------315 bytes c/u Locks ----------72 bytes c/u

Tabla 1. Ejemplo de posible configuración de memoria en SQL Server de Sybase

Ejemplo de conexiones

El número total de conexiones esta limitado por el sistema operativo o por la red.

Cuando se esta configurando el número de conexiones, se esta determinando el pedazo mas grande de memoria: ya que, no solo esta contando por conexión de usuarios externos, sino que también hay usuarios internos, como por ejemplo procesos.

Estas conexiones pueden incluir lo siguiente:

- Conexiones Internas: Tareas de auditoria, manejadores, etc.
- Conexiones externas: Dumps y loads, etc.
- Usuarios reales: Aplicaciones de cliente.

Ejemplo:

```
Sp_configure 'User Connections',50
go
reconfigure
go
```

Figura 11. Ejemplo de configuración de usuarios en SQL Server de Sybase

En el ejemplo de arriba se esta colocando una memoria por conexión de 50 kilobytes mas, en cualquier incremento en el tamaño de la pila (la pila es una estructura de datos en la cual se mantiene dicha conexión).

Asumiendo que existe un total de 50 conexiones, entonces el total de memoria usada es de 50*50 = 2,500 kilobytes, y esto significa que se le está quitando 2.5 Mb de cache de datos y de procedimientos.

4.1.9 Opciones que también consumen memoria: bases de datos, candados, objetos y dispositivos (devices).

- Dispositivos (Device) El número de particiones que serán reservadas para el uso de SQL server (por ejemplo: máximo vdevno).
- Bases de datos abiertas (Open Databases) 1 por base de datos.

- Sitios Remotos (Remote sites) Cada uno toma 1 conexión; 1 por SQL server remoto.
- Conexión remota (Login Remoto) Total de usuarios remotos simultáneos.

- Objetos abiertos, candados Escala proporcional con la complejidad de la base de datos y el número de usuarios o espera de errores e incremento cuando se encuentran.

4.1.10 Las extenciones de buffers (Buffers Extents) de E/S incrementan el tamaño del servidor

Debido a estos, se debe de tomar en cuenta lo siguiente:

- Reservar páginas en el "caché" de datos, para uso exclusivo en la creación de índices.
- El rendimiento se incrementa dramáticamente para la sentencia 'Create Index'.
- Solamente un índice bajo creación obtiene el uso de caché especial en el tiempo.
- Cada buffer añade 16 Kb al tamaño de SQL server.

Y para poder añadirle extensiones a los buffers del kernel, se puede usar la siguiente instrucción:

Por ejemplo:

```
Sp-configure "Extent I/O buffers",20
```

Figura 12. Instrucción para añadir buffers a la memoria de SQL Server

4.1.11 Auditoría

Las capacidades de auditoría construidas por el servidor de Sybase, pueden ayudar para monitorear el uso de las aplicaciones. Es posible examinar información de auditoria con otros sistemas estadísticos para ayudar a identificar cuellos de botella a nivel de rendimiento. Algo interesante que se puede hacer es afinar el sistema, comparando con los registros generados por la auditoria, junto con la carga esperada.

Para realizar la auditoria, SQL Server necesita la base de datos llamada "Sybsecurity". Esta es creada como parte del proceso de instalación de la auditoria. Contiene tablas del sistema que son halladas en el modelo de la base de datos, y dos tablas del sistema adicionales que son:

- sysaudits - que es el rastro de la auditoria.
- sysauditoptions - La cual contiene las selecciones globales de la auditoria.

4.1.11.1 Las colas de auditoría consumen memoria

Sin embargo, la auditoria también consumen recursos:
Las colas de auditoría de Sybase, consumen memoria, y el tamaño por defecto para las colas de auditoria es de 100.

- En un registro simple arriba de 424 bytes:

- El registro puede ser tan pequeño como 22 bytes cuando se escribe a una página de datos.
- El número de registros auditables por página : de 4 a 80.

- El máximo número de registros auditables que pueden ser perdidos es igual al tamaño de la cola auditable (en registros) mas 20.

Después de dejar los registros en la cola de auditoria, estos son grabados en la páginas de buffer hasta que son escritos en el "sysaudits".

- Los requerimientos de memoria por defecto, para los tamaños de las colas de auditaje son de 42K.

4.1.11.2 Proceso de auditoría

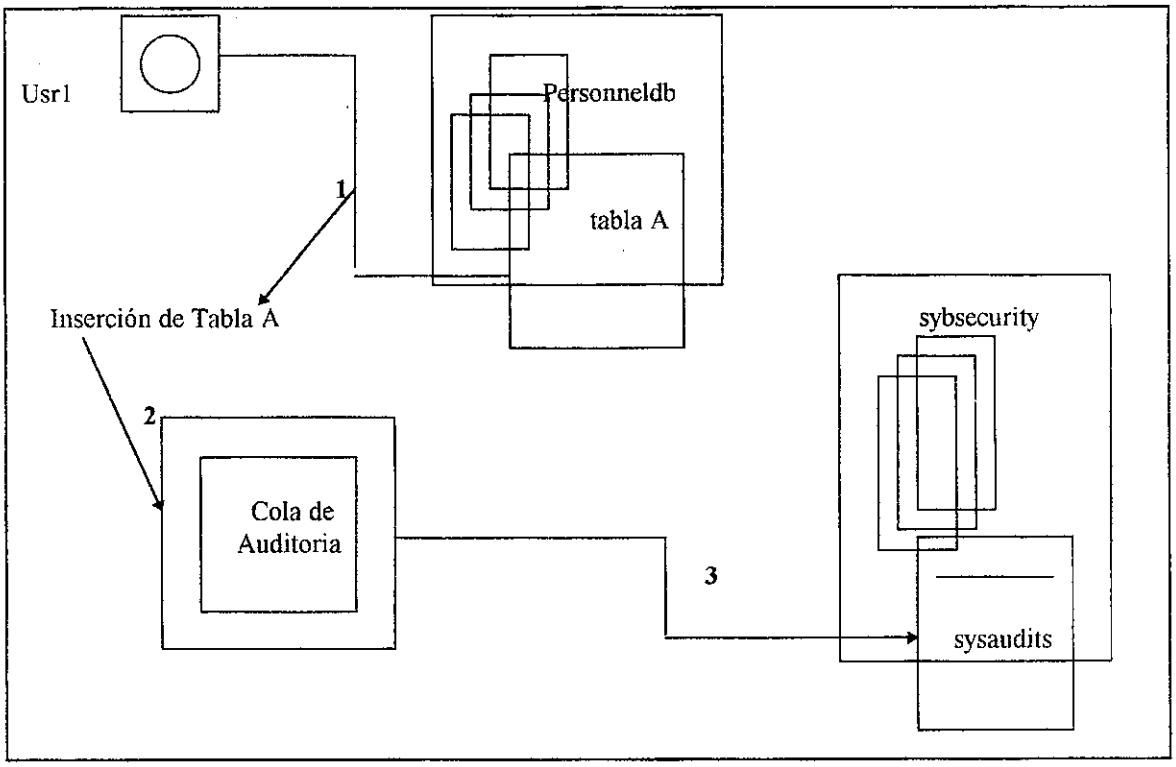


Figura 13. Grafica del proceso de auditoría de Sybase

Explicación

1. Los usuarios tratan de insertar a la tabla A, y la auditoria ha sido levantada para este evento.
2. Después que los permisos chequean información sobre este comando y el proceso es registrado en la cola de auditoria en la memoria compartida (separada del caché de datos y procedimientos)
3 Las tuplas de la cola de auditoria son copiadas a disco ("sysaudits" en la base de datos sybsecurity) cuando la tarea de auditoria es hecha.

4.1.11.3 Auditoria y rendimiento

Una gran cantidad de auditoria puede afectar el rendimiento.

Si la cola de auditoria (en memoria) está llena, los procesos de usuarios que generan dicha auditoria se quedan en un estado de espera (estado de "sleep").

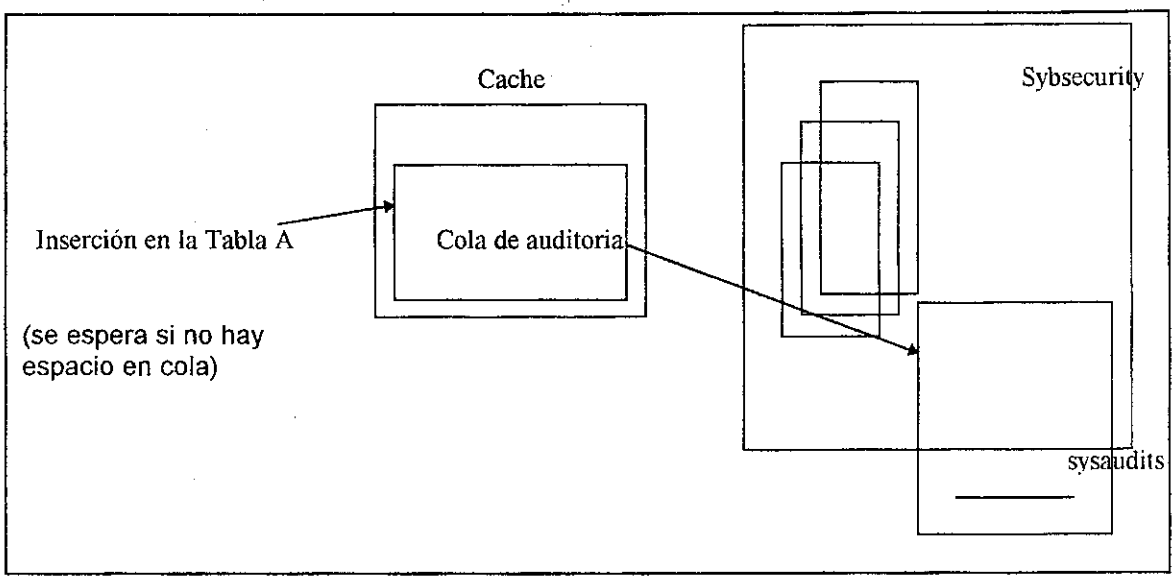


Figura 14. Inconveniente de no tener espacio en la cola de auditoria

El rendimiento puede sufrir si el "sybsecurity" esta siendo altamente usado (uso de disco). Se pueden afectar el rendimiento al configurar "los tamaños de colas de auditoria"

Colas Largas Mejor rendimiento.

Colar Cortas Pocos registros de auditoria son vulnerables en caso de fallas.

4.1.11.4 Tamaño de las colas de auditoría

Estos son los valores por defecto:

- Los registros por defecto en las colas de auditoria es de 100.
- Tamaño del registro: 424 bytes.
- Tamaño del ancho de la cola por defecto 42K

4.1.12 División del resto de la memoria

4.1.12.1 "Cache" de procedimientos

El tamaño del "cache" de procedimientos es expresado como un porcentaje. El valor por defecto es 20. Esto significa que el 20% de la memoria disponible que le deja al SQL server es reservado para el "cache" de procedimientos. El porcentaje del data cache es entonces 100 menos el porcentaje especificado para el cache de procedimientos. El valor óptimo para el "cache" de procedimientos variará de aplicación en aplicación, y podría también variar dependiendo de los patrones de uso durante el día. Se recomienda experimentar y mantener una bitácora de resultados.

4.1.12.2 "Cache" y E/S de disco

El uso del "cache" es en esencia, un trueque entre espacio - tiempo. Un "cache" es una estructura de almacenamiento que trata de reducir la cantidad de tiempo necesario para accesar otras estructuras de almacenamiento (por ejemplo, un disco). Sin embargo, el hecho de que exista un caché no implica que el acceso a disco será reducido. De hecho, si el algoritmo usado para decidir cuando una página será removida del "cache", no es afinada para el patrón normal de acceso, entonces es posible que el cache perjudique el rendimiento, porque ningún acceso de entrada/salida del disco es salvado, y manejo del caché tendrá sobrecarga.

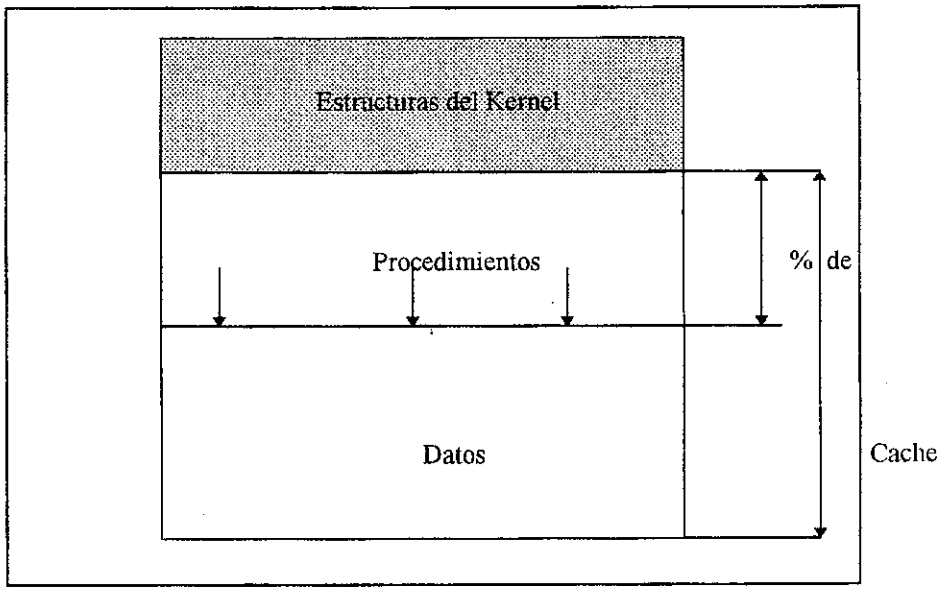


Figura 15. Distribución de "caché" de Sybase

4.1.12.3 ¿Para qué sirve el "cache" de procedimientos?

En el "cache" de procedimientos se cargan el plan de ejecución de las consultas optimizadas para procedimientos, "triggers" y vistas.

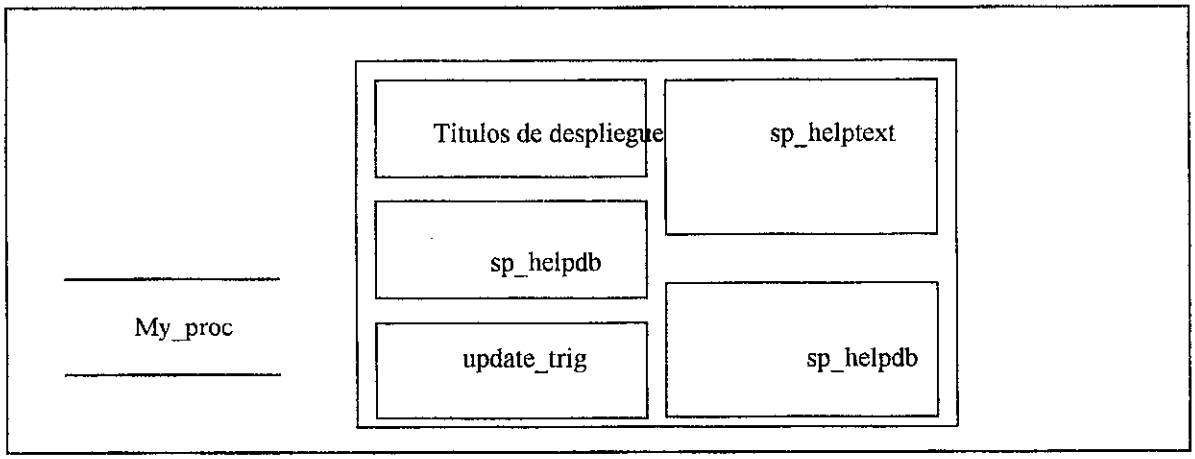


Figura 16. Ejemplo gráfico de la distribución del "cache" de procedimientos de Sybase

Los procedimientos son no re-entrantes, por lo que cada usuario simultáneamente tiene que obtener una copia. Si el "cache" de procedimientos es muy pequeño, los procesos de usuario deben de esperar.

Los procedimientos, "triggers" y vistas son optimizados, cuando están cargados en el cache de procedimientos. Si el procedimiento está en "cache", este no necesita ser re-compilados para ser re-utilizados.
Si no hay espacio en el cache de procedimientos, el procedimiento no puede ser ejecutado.

4.1.12.4 Tamaño del "caché" de procedimientos

4.1.12.4.1 Tamaño mínimo para el "caché" de procedimientos

El tamaño mínimo para el "cache" de procedimientos es la cantidad más pequeña de memoria, en la cual permite al menos una copia de cada objeto compilado y usado frecuentemente, el cual en el "cache".

4.1.12.4.2 Errores del "cache" de procedimientos

Si no hay suficiente memoria para cargar otro árbol de selección ("query") o plan, SQL Sever reportará un error 'Error 701'.

Si un árbol de selección (query) o plan de selección es demasiado largo (131072 bytes es el máximo permitido), SQL Server reportará un 'Error 703'.

Si se llega al máximo número de objetos compilados que están en uso, SQL server reportará un error 'Error 701'.

4.1.12.4.3 ¿Qué tan grande debería ser el "cache" de procedimientos?

La siguiente fórmula da un buen punto de inicio:

Tamaño del "cache" de procedimientos = *(máximo número de usuarios concurrentes) * (tamaño más largo de un plan de selección) * 1.25.*

Mínimo del "cache" de procedimientos necesarios = *(No. de procedimientos pricipales) * (promedio del tamaño de plan de selección)*

Se debe de recordar que cuando se incrementa el tamaño del cache de procedimientos, se esta decrementando el tamaño del "cache" de datos.

4.1.12.5 ¿Para qué es el "cache" de datos?

Sirve principalmente para:

- Cargan los datos usados mas recientemente y las páginas de índices.
- Los servidores siempre buscan primero en "cache".

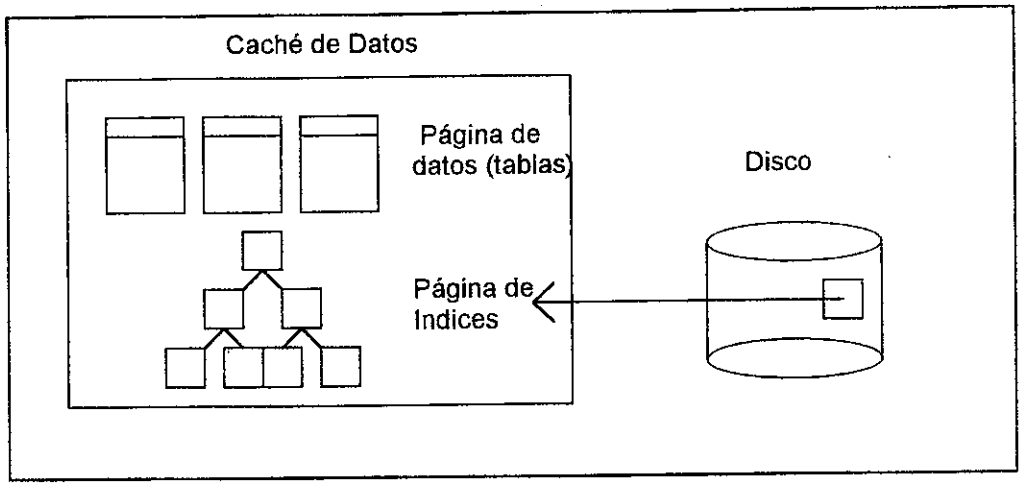


Figura 17. Ejemplificación gráfica del uso del "cache" de datos

4.1.12.5.1 Páginas viejas en el "data cache"

Observemos las siguientes consideraciones:

- Los índices más antiguos en el "cache" son mas lentos que las páginas de datos.

- La estrategia de - El menos recientemente usado ("LRU - Least - recently - used") - es utilizado para las páginas de datos. (Sin embargo no es así para las páginas de índices).

4.1.12.5.2 Efectos del "data cache" para recuperación

Como un ejemplo, se considera una tabla que contiene 1000 páginas. La primera transacción que usa esta tabla, tiene una gran cantidad de E/S a disco. Si esta se encuentra en cache, otro pueden tomar ventaja de esta. Esta es una muy buena ventaja porque con una vez que se suba, los demás usuarios pueden usarle, y es algo que vale considerar cuando se afina el tamaño del "cache" de datos. Se debe de notar que, bajar el "cache" toma una gran cantidad de tiempo.

4.1.12.5.3 Selecciones y tamaños del "data cache"

Considerar una serie de sentencias de selección, las cuales tiene la característica de ser aleatorias, y que son ejecutadas en un período de tiempo. Se asume que el caché esta completamente vacío, por lo que la primera sentencia de selección (query) se garantiza que necesita E/S de disco. Mientras que el "cache" es llenado, hay un incremento de probabilidad que una o más recuperaciones de disco puedan ser proporcionadas o satisfechas por el "cache", así se reduce el promedio de tiempo de respuesta del conjunto de recuperaciones (sentencias de selección que al ser ejecutadas recuperaran alguna información). Una vez el "caché" está lleno, habrá una muy buena probabilidad de hallar una página deseada en el "cache".

Asumiendo que el "caché" es más pequeño que el número total de páginas de disco, habrá siempre una oportunidad de que una transacción dada espere por E/S de disco. Esto es que, un "cache" no reduce el tiempo máximo de repuesta posible, simplemente decrementa la probabilidad que el máximo retardo en una transacción sea efectuado.

4.1.12.5.4 Efectos del "cache" de datos en recuperación

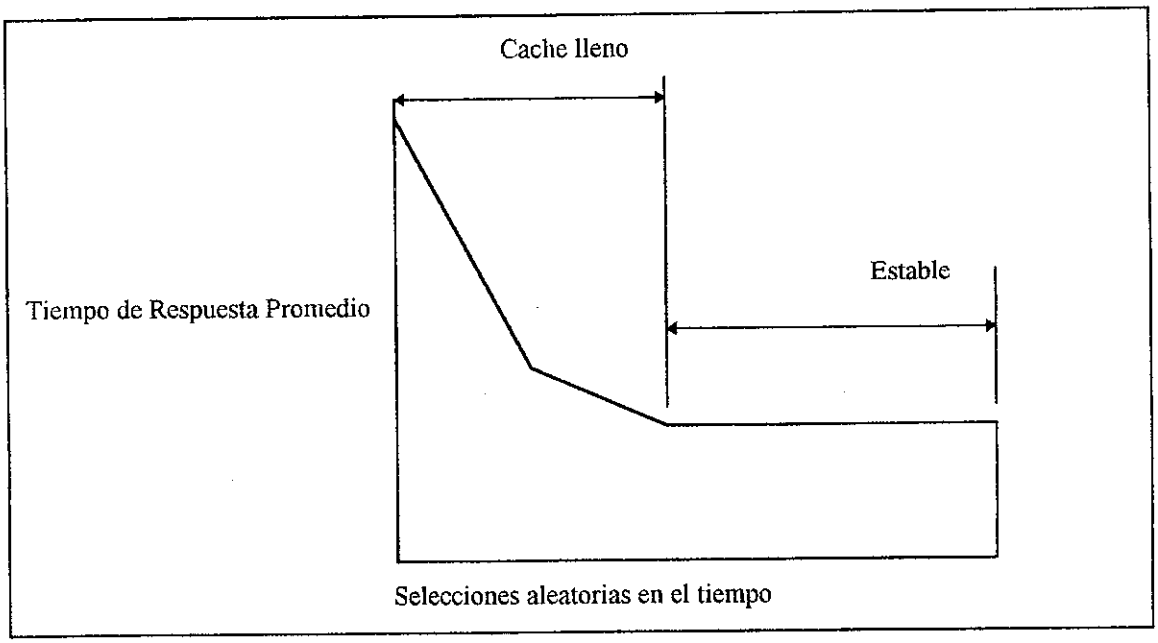


Figura 18. Ejemplo gráfico de selecciones aleatorias en el tiempo

Cuando el "caché" se estabiliza, cada nuevo requerimiento tiene una oportunidad de hallar páginas ya en memoria.

4.1.12.5.5 Efectos del "cache" de datos en actualizaciones (updates)

4.1.12.5.5.1 El tamaño del "cache" de datos y las actualizaciones (updates)

El comportamiento del "cache" en presencia de transacciones de actualización es más complicado que el de recuperaciones. Hay aún un período inicial durante el cual, el cache es llenado. Entonces, debido a que las páginas del "caché" son modificadas, (por ejemplo.: páginas sucias), hay un punto en el cual el "cache" debe iniciar a escribir estas páginas al disco, antes que este pueda cargar otras páginas. En el tiempo, la cantidad de escrituras y lecturas estables y todas las subsecuentes transacciones tienen una probabilidad de requerir una lectura a disco y otra probabilidad de causar una escritura a disco. El período de estado seguro es interrumpido por puntos de verificación, los cuales causan que el "cache" escriba todas la páginas sucias a disco.

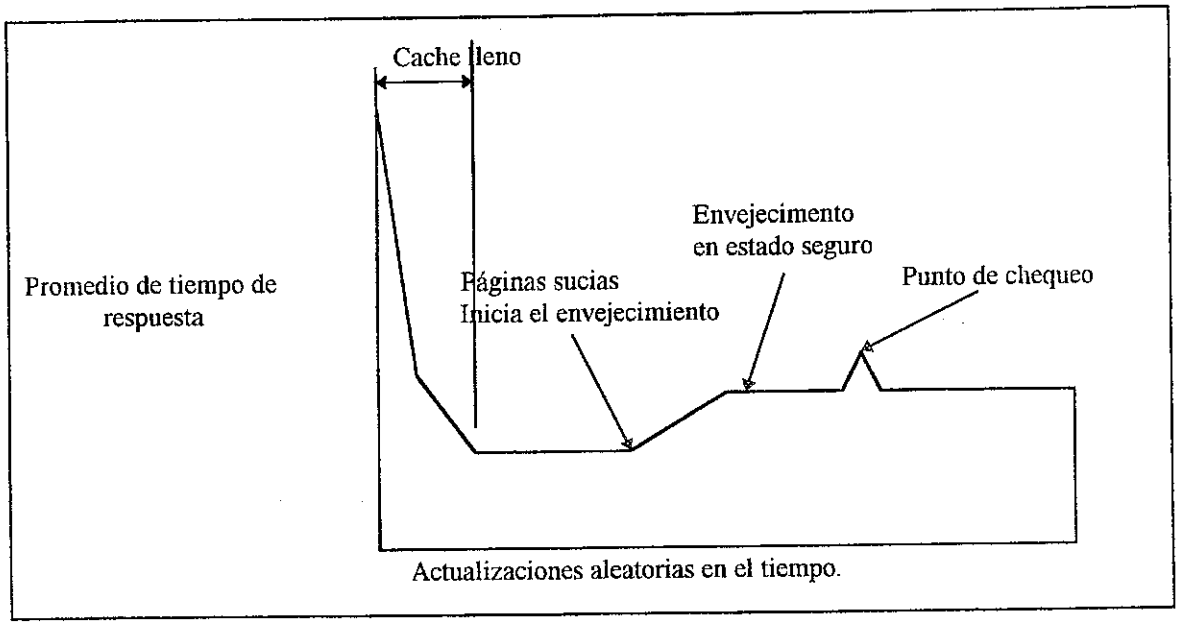


Figura 19. Ejemplo gráfico de actualizaciones aleatorias en el tiempo

Las actualizaciones pueden causar que algunas páginas sean cambiadas (marcadas como "sucias"). Cuando el "cache" esta lleno de páginas sucias, una página debe ser escrita antes de que una página nueva sea leída.

Los puntos de verificación escriben todas las páginas sucias a disco.

4.1.12.5.6 Medida del rendimiento y del "caché" de datos

Es importante tener en mente que el comportamiento del "caché" de datos y de procedimientos, cuando se mide el rendimiento de un sistema. Cuando una prueba inicia, el "cache" puede estar en alguno de los siguientes estados:

- vacío
- lleno aleatoriamente
- parcialmente lleno
- determinístico

Un "cache" vacío o un lleno aleatoriamente generará pruebas de resultados repetibles, porque el "caché" está en el mismo estado que cualquier otra prueba. Un parcialmente lleno o un determinístico tiene páginas en este ya que hay una transacción que fue ejecutada. Cuando hay pruebas, tales páginas podrían tener el resultado de una iteración previa al de dicha prueba. En tal caso, si los próximos resultados de la prueba requieren estas páginas, entonces ninguna operación de E/S a disco será necesaria.

Esta situación se inclina a resultados de una prueba puramente aleatoria y conduce a obtener estimados de rendimiento. La mejor estrategia de prueba es iniciar con un "cache" vacío o asegurarse que todos los pasos de la prueba tienen accesos aleatorios en la base de datos.

4.1.12.5.7 Rendimiento del data "cache"

Determinando el "caché hit ratio" al examinar lo siguiente:

- El porcentaje de páginas requeridas que son satisfechas en memoria.

- Si se tiene un 100% entonces significa que la base de datos reside en memoria.
- Si tiene un 5% indica que el cache es demasiado pequeño.

Desplegando el "cache hit ratio":

- dbcc tablealloc
- dbcc indexalloc
- configurar estadísticas de accesos de E/S.

"hit ratio" = Lecturas lógicas/(Lecturas lógicas + Lecturas físicas)
(Como se ha visto en capítulos anteriores)

4.1.12.6 Hallando los tamaños del "cache"

Se debe de notar que ambos, la bitácora de errores y el dbcc memusage reportan cuanta memoria de caché para datos y para procedimientos estar reservado. El reporte dbcc memusage es mas exacto, ya que incluye lo que es el gasto general de memoria; sin embargo, la bitácora de errores es mas utilizada, ya que esta reporta lo que actualmente esta disponible.

En términos de utilización, la bitácora de errores, el "caché de buffers" y las cabeceras de procedimientos no son intuitivos.

Cuando el SQL Server es levantado, el estado de la bitácora (log) de errores indica cuanta memoria cache de datos y de procedimientos está disponible.

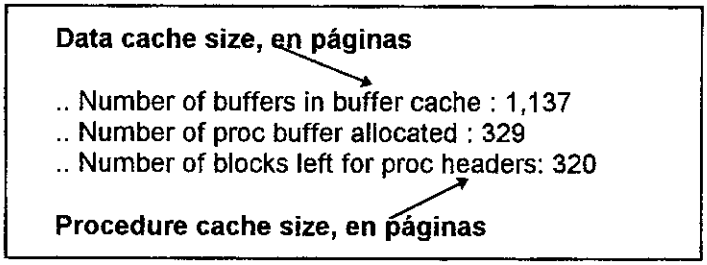


Figura 20. Ejemplo de la bitácora de errores.

dbcc memusage también provee esta información.

Su salida incluye (overhead) gasto general, el cual no está disponible para uso.

Buffer cache Páginas disponibles para lectura de disco.

Proc buffers (o buffers de procedimiento) son estructuras de datos usados para manejar los objetos compilados (cualquier procedimiento almacenado, "trigger", restricción de chequeo, o vista) en el caché de procedimiento. Un "proc buffer" es usado por varias copias de un objeto almacenado nombrado en el caché de procedimientos. Cuando el SQL server inicia, este determina el número de "buffers" de procedimiento requerido y multiplica este valor por el tamaño de un sencillo buffer de procedimiento (76 bytes) para obtener la cantidad total de memoria requerida. Entonces se reserva la cantidad de memoria, y entonces esta porción de memoria es tratada como un arreglo de "buffers" de procedimientos. A menos que se diga lo contrario sobre alguna estructura de datos, los "buffers" de procedimiento pueden dividir páginas. Uno por procedimiento.

Proc headers (o cabecera de procedimientos)

Es donde un objeto compilado (tal como un procedimiento almacenado) es guardado.

4.2 El "tempdb"

El "tempdb" es usado por todos los usuarios del SQL Server. Cualquiera puede crear objetos en el "tempdb", e incluso muchos de sus procesos utilizan este.

Por ejemplo existen las tablas de trabajo "WorkTables", las cuales son creadas, pobladas, ejecutadas y borradas durante la ejecución de una selección. Por ejemplo son usadas a la hora de hacer:

- Un ordenamiento (order by)
- un distinción (distinct),
- un agrupamiento (group by),
- listas
- y reformateo.

Las tablas de trabajo (WorkTables) son creadas en el "tempdb".

El "tempdb" puede ser usado por servidores de bases de datos inicialmente para:

- Procesamiento interno (ordenamientos, tablas de trabajo, reformateo, etc.)
- El almacenamiento temporal creado por tablas/índices para los usuarios.

Cualquier selección que este como requerimiento, potencialmente puede usar el "tempdb".

Las tablas de trabajo interno son creadas en el "tempdb" para:

- Distinct
- Not Exists
- Group by
- Order by
- reformateo
- o estategia (política internal del proceso)

4.2.1 Almacenamiento en tablas temporales

Temporales verdaderas

- Existen solo para la duración de la sesión de usuario o ejecución de procedimientos almacenados.
- Creadas usando "create table #temptable".
- No pueden ser compartidas.

Tablas temporales permanentes

- Puede durar entre sesiones
- puede ser compartidas dándoles permisos (via "grants")
- se crean usando se usa el comando "create table tempdb..temptable"
- Deben ser explícitamente borradas por el usuario.
- Se borran al re-inicializar SQL Server.

Las tablas de trabajo creadas por SQL Server para ordenamientos, etc. nunca son compartidas.

Para poder hacer permanentes tablas temporales que estén disponibles todo el tiempo en que SQL Server este arriba, es recomendable que estas estén creadas en el modelo de la base de datos.

En las tablas temporales permanentes, se debe de recordar que el "tempdb" es reconstruido cuando el SQL Server esta levantando.

4.2.2 ¿Por qué preocuparse por el "tempdb"?

Principalmente por las siguientes razones:

- Se puede llenar frecuentemente
- Las búsquedas se hacen lentas
- Los usuarios pueden quedarse bloqueados por la creación de tablas temporales.

4.2.3 Usando tablas temporales para dividir uniones (Joins) de tablas múltiples

Es usado en la siguientes circunstancias:

- Si la selección esta reuniendo más de cuatro tablas y no se selecciona el mejor plan de ejecución.
- Selecciones (queries) que exceden uniones (joins) con diez y seis tablas.

- Selecciones (queries) muy complejas.
- Si se desea filtrar datos como un paso intermedio.

El uso de tablas temporales en estas circunstancias es muy útil y tiende a reducir los accesos de E/S (pocas pasadas a través de tablas base).

Desventaja: incrementa la carga y los requerimientos de espacio en el tempdb.

Otros contextos en los cuales las tablas temporales pueden ser útiles:

Denormalización de varias tablas hacia unas pocas tablas temporales.
Normalizar a denormalizar tablas en orden para agregar un mejor procesamiento.

4.2.4 Índices en tablas temporales

Se puede definir índices en tablas temporales.

Restricciones:

- El optimizador utilizará el índice solo si las estadísticas han sido actualizadas.
- En un procedimiento almacenado, no puede crear una tabla temporal, y entonces se debe crear el índice, por lo tanto tiene que esperar a que el optimizador use este índice. Esto es que si se crea una tabla temporal, y luego un índice en la tabla en un procedimiento

almacenado, el optimizador no podrá usar este índice en la tabla. Sin embargo, se puede usar el índice para forzar unicidad (único).

Guías:

- Actualizar las estadísticas en transacciones separadas o crear el índice después de que la tabla es cargada.
- Crear una tabla permanente en el "tempdb" en vez de crear una temporal.
- Crear los índices necesarios en tablas permanentes del "tempdb".
- Para índices en tablas temporales, asegurese de que el costo de esta creación no excede al del beneficio (esto normalmente se hace en los manejadores por medio de el plan de ejecución).

Tres áreas principales pueden ser trabajadas fácilmente: El tamaño, el lugar y el uso óptimo del "tempdb" (otras estructuras extras además de las tablas internas de trabajo).

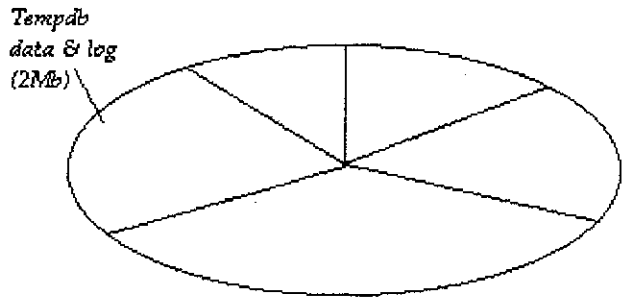
4.2.5 Tamaño del "tempdb"

Colocando el tamaño adecuado del "tempdb" ayudará a minimizar la mayoría de errores ocasionados en el "tempdb", y a mantener un rendimiento consistente para todos quienes usan dicha memoria (tempdb). Este depende

del tamaño y cantidad de datos y de conexiones que se utilice (mas adelante se darán algunas recomendaciones).

4.2.5.1 Reservación inicial de "tempdb"

Cuando se instala SQL Server, 2 Mb son reservados en el dispositivo maestro para el "tempdb". Se puede lograr incrementar esta reservación, o mover esta hacia otro dispositivo separado (device); En la siguiente figura se muestra un ejemplo:



d_master

```
1>  sp_helpdb tempdb
2>  go
```

name	db_size	owner	dbid	created	status
pdb	2.0 Mb	sa	. . .	select into/bulkcopy	tem

device_frag	size	usage	free kbytes
master	2.0Mb	data and log	....

Figura 21. Ejemplo del estado actual del tempdb

4.2.5.2 Tamaño del "tempdb"

Este necesita ser lo suficientemente grande para manejar: (para todos los usuarios concurrentes)

- Ordenamientos internos
- Otras tablas de trabajo internas
- Tablas temporales
- Indices en tablas temporales
- Tablas permanentes en el "tempdb"

Si se tienen datos aproximados de un uso tentativo, entonces es recomendado a la hora de instalar el SQL Server dejar un espacio suficiente en el dispositivo maestro (master device). Sin embargo esto no es lo normal, pero algo se si se puede hacer es determinar si el "tempdb" tiene un tamaño apropiado; Esto se hace determinando como esta siendo utilizado.

4.2.5.3 Ingresos para el tamaño del tempdb

Máximo número de usuarios concurrentes U

Esto es proc ids (identificadores de procesos a nivel del sistema operativo).

Tamaño de los ordenamientos S

Tamaños requeridos en páginas, de ordenamientos internos (selecciones con ORDER BY que no son soportados por un índice). Una

manera de medir el tamaño de las tablas indirectamente puede ser por estadísticas de E/S para inserciones.

Tamaño de las tablas de trabajo **W**

Tamaño de tablas de trabajo necesarias para re-formatear, agrupamientos (Group by) , Etc. pero no para ordenamientos. (Se puede examinar esto, observando el número de páginas listadas en estadísticas de E/S).

Número de pasos en la selecció de planes de trabajo (query plans) **Q**

Número típico de pasos que usa una selección (query) para re-formatear, agrupar (Group by), Etc. Indica el número de tablas de trabajo creadas.

Número de procedimientos almacenados y/o sesiones que crean tablas e índices temporales. **P**

Número de procedimientos almacenados (remotos o locales) y/o sesiones de usuario que crean tablas temporales e índices temporales.

Tamaño de las tablas temporales e índices **T,I**

Requerimientos de tamaño, en páginas, de esta tablas temporales (T) e índices (I). Pueden ser desplegados, usando estadísticas de E/S.

Número de tablas temporales e índices **C,D**

Número de tablas temporales (C) e índices (D) creadas por procedimiento almacenado o por sesión.

4.2.6 Localización del "tempdb"

Se recomienda colocar el "tempdb" lejos de la actividad de otras aplicaciones, es esencial para el rendimiento.
También se recomienda el mantener las páginas del "tempdb" contiguas.

Un hecho deseable es que se altere el "tempdb" en un dispositivo separado, esto es, en un disco físico separado, o alterar el "tempdb" hacia otro dispositivo, en el mismo disco físico del maestro. En cualquiera de las dos maneras se debe de evitar colocarlo cerca de las bases de datos de aplicación.

Algo que no es una muy buena idea es el dividir el "tempdb" en distintos discos. Esto hará que las tablas temporales o las tablas de trabajo de dividan en discos, y definitivamente disminuirá el rendimiento. Lo mejor es que el "tempdb" tenga una localización única y contigua.

4.2.7 Algorítmo para prevenir que las tablas temporales se dividan en múltiples dispositivos

- Alterar el "tempdb" a un dispositivo que no sea altamente usado por aplicaciones.
- Borra el segmento por defecto en el dispositivo maestro.
- Borrar el segmento sistema (system) en el dispositivo maestro.

Resultado: Todas las tablas temporales y de trabajo estarán localizadas en el nuevo dispositivo.

4.2.8 Bloqueos en el "tempdb" (Locks)

Bloqueos en el tempdb pueden ocurrir si grandes cantidades de usuarios concurrentes están:

- Creando y borrando tablas temporales y sus índices.
- Actualizando tablas permanentes en el "tempdb".

Razones:

La información de tablas temporales puede caer en las mismas páginas dentro del "tempdb" como las tablas del sistema del "tempdb" (sysobjects, syscolumns, sysindexes).

Tablas permanentes en páginas de datos del "tempdb" están siendo reservadas para un proceso con duración de la transacción.

5. INFORMIX On-Line (Versión 6)

5.1 Manejo de memoria

Los tópicos que se abarcarán en este tema son:

- Tipos de memoria a examinar
- Calculo de los requerimientos de memoria
- Como examinar la memoria
- Algunos tópicos para decrementar el uso de la memoria
- Monitoreo y afinamiento de la razón de lecturas de cache.

La memoria esta comprendida en uno de dos formas en el sistema "On-Line": memoria de procesos y memoria compartida. La memoria de procesos es aquella necesaria para los procesos servidores de la base de datos. La memoria compartida es en la cual se carga todo lo utilizado por los procesos servidores, tales como el buffer cache, las estructuras de candados, y los buffers lógicos de bitácora (de logs).

En versión 6, el trabajo de todos los usuarios es completado por unos pocos procesos servidores de la base de datos (entre 7 y 20 procesos) ya que contiene la característica de multi-enlazado (multitheaded). Los procesos servidores de la base de datos, o *procesadores virtuales* (vps), se inician cuando la base de datos se levanta y están activos hasta que dicha base de datos se baja.

El tamaño del proceso del vps se mantiene estático mientras este está ejecutándose.

Los requerimientos de memoria compartida para la versión 6.0 se incrementan dramáticamente de la versión 5.0, esto es debido a que la memoria compartida carga muchos de los datos que previamente eran cargados en el proceso "sqlturbo" (en la versión 5.0, el "sqlturbo" era un proceso servidor que corría para cada usuario. Las porciones de datos para la memoria en cada proceso eran dinámicas, y crecían de manera independiente, de acuerdo a la utilización de cada usuario). La memoria compartida de esta versión esta dividida en tres partes:

Residente la memoria compartida es muy similar a la memoria compartida de la versión 5.0. Esta es estática, y contiene estructuras tales como caché de buffers, candados y bitácoras lógicas y físicas.

Virtual esta memoria compartida contiene las estructuras necesarias principalmente para los subsistemas multi-enlazados ("multithreaded"), tales como la pila ("stack") enlazado de los usuarios. La memoria virtual esta organizada en "pools", donde cada "pool" es usado para un propósito específico. Cada "pool" es reservado cuando es necesario, y es liberado cuando ya no es utilizado. En versión 5.0, mucha de esta memoria era mantenida en cada usuario como proceso servidor de la base de datos. Debido a que la versión 6.0 utiliza la arquitectura de multi-enlazada ("multithreaded") los servidores de la base de datos permite que un solo apuntador se ejecute para múltiples procesos, y es por esto que los pools deben de ser localizados en memoria compartida para que puedan ser accesados por todos los procesadores virtuales. La memoria compartida

virtual puede crecer de acuerdo a la necesidad que tenga un usuario conectado y que ejecute sentencias SQL.

Mensaje esta memoria compartida es usada como un "bulletin board" (ver glosario) para mensajes entre el cliente y los procesos servidores de la base de datos. La memoria compartida es estática y es configurable, ya que se puede colocar el máximo número de conexiones de memoria compartida en los archivos de configuración del "OnLine".

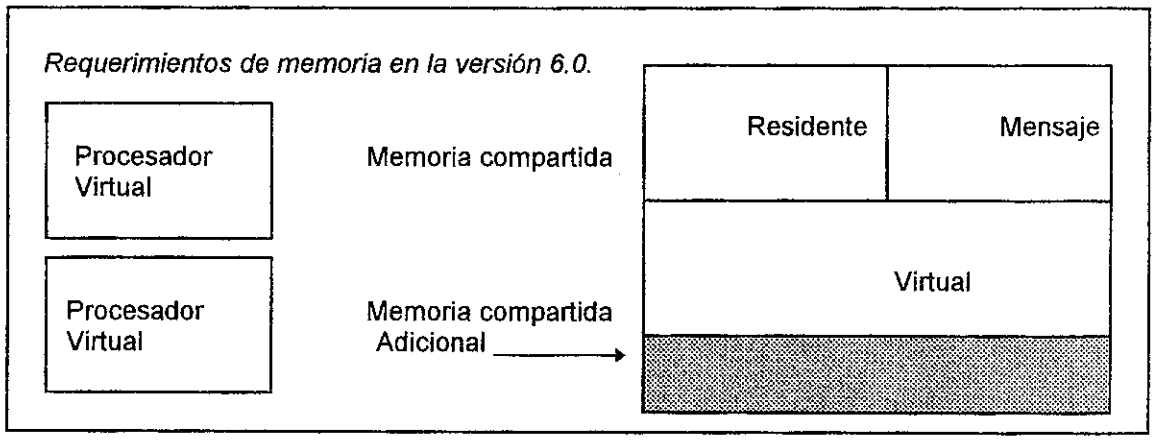


Figura 22 Requerimientos de memoria compartida de Informix versión 6.0. FUENTE: Informix On-Line Performance Tuning - Elizabeth Suto

5.1.1 Calculo de los requerimientos de memoria

5.1.1.1 Memoria de procesos

Cada proceso esta compuesto de tres partes: El texto (o tamaño del código), la pila ("stack"), y los datos. Los procesos servidores de la base de datos de Informix, comparten el mismo código, o espacio de texto, así que esta

cantidad necesaria de memoria, es requerida para una sola vez, de todas las ocurrencias del mismo proceso.

Se puede usar el comando de unix "size", para determinar el tamaño fijo de un proceso (esto no incluye memoria extra añadida cuando el proceso está ejecutándose). Por ejemplo, para correr el comando "size" para el proceso sqlturbo (versión 5.0):

```
$ size sqlturbo

text       data       bss        dec        hex
1171456    122880     12540           1306976
     13f160
```

Figura 23 Ejemplo de la salida de la sentencia "size" de Informix.

En este ejemplo, el tamaño del texto es de 1,171,456 bytes, o sea 1.12 megabytes. Los datos y la pila (stack) pueden ser calculados añadiendo la columna de datos y la bss. El ejemplo muestra que la sumatoria de estos dos da un total de 148,160 bytes, o aproximadamente 144 Kbytes.

5.1.1.2 Estimación de la memoria compartida (versión 6.0)

En la versión 6.0, los segmentos de memoria compartida residente, pueden ser calculados, usando el monitor llamado **"omonitor"** a través de los parámetros de memoria compartida, que se encuentran en una opción del menú. Un ejemplo de esta pantalla podría ser como la siguiente:

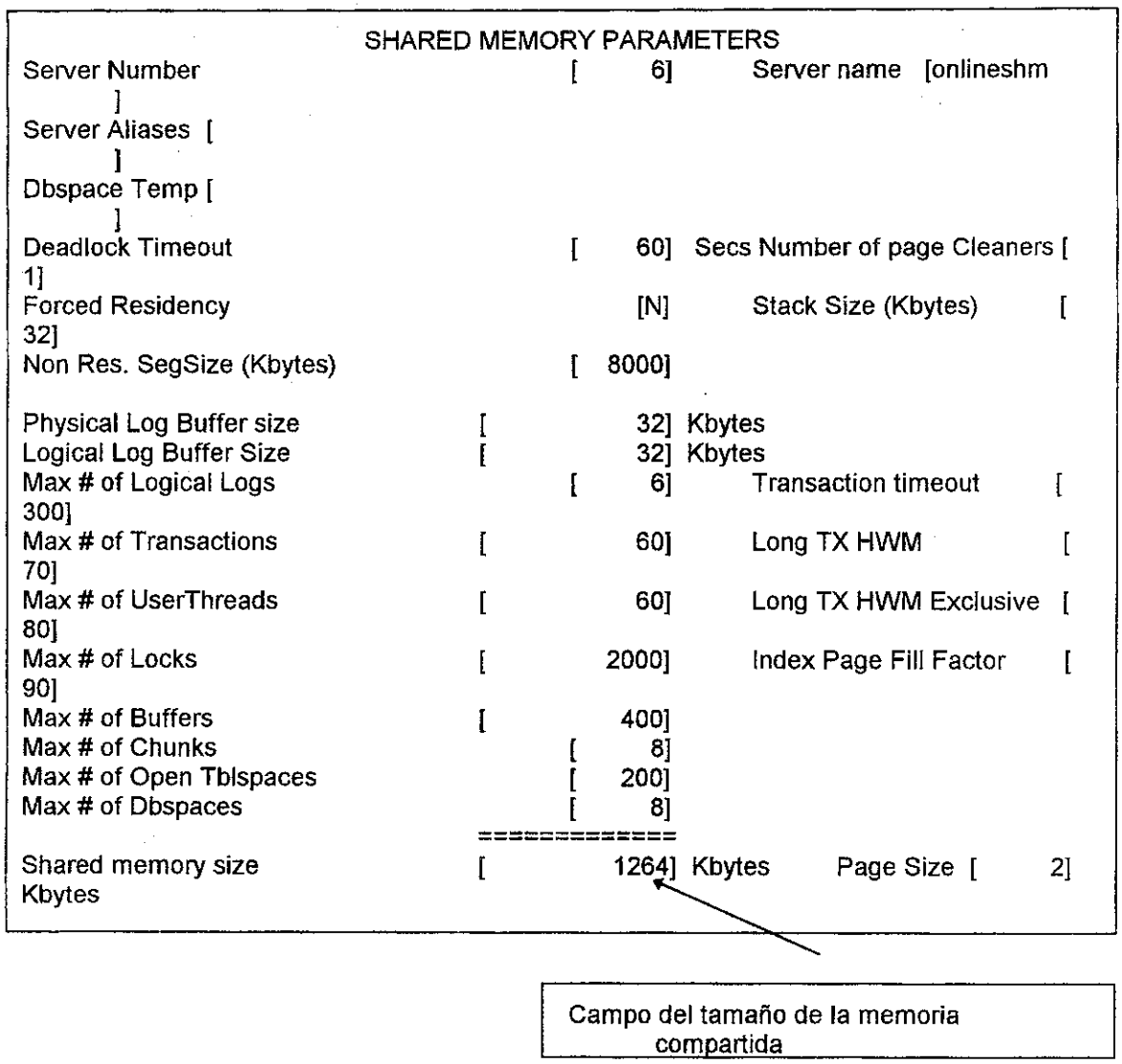


Figura 24 Ejemplo del despliegue de los parámetros de memoria compartida con el "omonitor"

Debido a que el administrador configura los parámetros que afectan la memoria residente, Se puede ver el campo del tamaño de la memoria compartida. Los parámetros de configuración que pueden afectar el tamaño de la memoria residente, son el tamaño de buffers de bitácora físicos y lógicos ("Physical Log Buffer Size" y "Logical Log Buffer Size"), el máximo número de

apuntadores (threads) para usuarios (Max # of Userthreads), el máximo número de candados, el máximo número de buffers, el máximo número de pedazos de disco (chunks), el máximo número de "tablespaces" abiertos, y el máximo número de "dbspaces". El parámetro que cambia la porción residente de la memoria compartida, de manera significativa es el número máximo de buffers.

Los mensajes en los segmentos de memoria compartida tienden a ser relativamente pequeños, y son calculados por medio del número de conexiones a la memoria compartida, especificado en el parámetro llamado NETTYPE. Este parámetro contiene varios campos, de entre los cuales, el tercero especifica el número de conexiones. Por ejemplo:

```
NETTYPE  ipcshm,1,50,CPU
```

El tercer parámetro, indica que 50 conexiones son permitidas vía la memoria compartida.

La fórmula para calcular el tamaño aproximado del segmento de mensajes, en bytes, es la siguiente:

(10,531 * #conexiones) + 50,000

Usando el parámetro NETTYPE, mostrado arriba, con 50 conexiones, el tamaño del segmento de mensajes podría ser (10,531*50)+50,00 es aproximadamente de 563 Kbytes.

El tamaño de los segmentos de memoria virtual que se necesita, es muy difícil de estimar, y esto es debido a que la memoria es reservada como es

necesitada (o sea por demanda), para una sesión individual. Si los segmentos virtuales, no pueden manejar la memoria necesaria para una sesión, algún otro segmento de memoria compartida, automáticamente reservará esto para el Informix "OnLine". Los segmentos de memoria compartida, no son liberados, hasta que Informix "OnLine", sea bajado.

Como una regla general, cada sesión que este ejecutando sentencias básicas de SQL, necesitaran al rededor de 100k de memoria compartida virtual. Sin embargo, los siguientes elementos podrían incrementar la cantidad de memoria compartida necesaria para una sesión:

- Ordenamientos
- Procedimientos Almacenados bastante largos
- La ejecución de un "oncheck"

Por ejemplo, un ordenamiento, puede ir de 100k hasta 1.5 Megabytes, dependiendo del tamaño de las columnas que están siendo ordenadas. Si la aplicación tiene cualquier sentencia SELECT con una cláusula ORDER BY, o si se está creando un índice, se debería estimar aproximadamente 500k por sesión durante horas pico.

5.1.2 Cómo monitorear la memoria

Unix reserva memoria, en unidades de un segmento. Que tantos segmentos Informix "OnLine" usa? Esto depende de varias cosas: El tamaño de segmentos del "kernel" de Unix, la cantidad de memoria que inicialmente es

requerida por Informix "OnLine", y la cantidad de memoria requerida, una vez que el sistema "OnLine" esta arriba (solo para versión 6.0).

El tamaño de los segmentos del "kernel" de Unix, son especificados principalmente por los parámetros del "kernel" SHMMAX o SHMSIZE (El nombre de los parámetros varían de sistema en sistema). Suponiendo que el parámetro SHMMAX esta configurado a un megabyte, e Informix "OnLine" necesita diez megabytes de memoria compartida, entonces Unix creará diez segmentos de memoria compartida.

Se debe monitorear el número de segmentos de memoria compartida creada, así como la cantidad de memoria usada. Algunos sistemas operativos, tienen un límite al número de segmentos que pueden ser creados. La mayoría de sistemas operativos, contienen el número de segmentos por medio del parámetro del kernel (SHMMNI).

El comando de unix **ipcs** presenta cada segmento de memoria compartida y su tamaño apropiado:

```
$ ipcs -m

IPC status from prod as of Sun Jan 23 13:32:17 1994

T     ID    KEY        MODE            OWNER           GROUP
      SEGSZ
Shared Memory
m     0     0x52574801      --rw-rw----          informix
      informix  1048576
m     1     0x52574802      --rw-rw----          informix
      informix  1048576
m     2     0x52574803      --rw-rw----          informix
      informix  1048576
m     3     0x52574804      --rw-rw----          informix
      informix  1048576
m     4     0x52574805      --rw-rw----          informix
      informix  819200
```

Figura 25 Ejemplo de la ejecución y salida del comando "IPCS" de unix para memoria compartida

En este ejemplo, existen cinco segmentos de memoria compartida, que ha sido reservada con un total de (1,048,567 * 4 + 819,200) = 5,013,504 bytes. En la versión 6.0 de informix, se pueden identificar los tipos de cada segmento reservado, con el comando llamado **onstat -g seg.** El siguiente es un ejemplo de su respectiva salida:

```
Segment Summary:
(resident segments are not locked)

id    key         addr      size       ovhd class      blkused
      blkfree
100   1381779457      800000    2613248    780  R      315
      4
103   1381779460      a7e000    8192000    696  V      288
      712
11    1381779468      262144    262144     567  M      29
      3
```

Figura 26 Ejemplo de la salida de la sentencia onstat -g seg para ver segmentos reservados de informix.

En el ejemplo anterior, se nota que existen tres segmentos de memoria compartida, uno para la memoria compartida residente, el otro para la memoria virtual, y el último, para la comunicación de memoria compartida por medio de mensajes. Cada uno es identificado por la columna de clase.

5.1.3 Sugerencias para mejorar el rendimiento del uso de la memoria

No hay mucho que hacer, para decrementar el uso de la memoria del servidor de la base de datos. En el caso de Informix OnLine, sea más eficiente comprar más memoria. Sin embargo, debido a que no siempre se puede hacer, entonces se presentan las siguientes opciones:

En la versión 6.0, permite colocar un tope a la cantidad de memoria compartida que es usada por el sistema OnLine. El parámetro SHMTOTAL es la cantidad total de memoria compartida que puede ser reservada en el sistema OnLine. Si Informix OnLine trata de reservar otro segmento de memoria, porque la sesión necesita más memoria, y el límite SHMTOTAL es rechazado, la tarea para la cual esta sesión necesita, fallará.

En adición al tope de la memoria compartida usada, se puede controlar cuanta memoria es reservada, cuando la memoria es requerida. El parámetro SHMVIRTSIZE, es la cantidad de memoria virtual compartida, que reservará el sistema OnLine cuando este se levante. El valor por defecto para este parámetro es de 8,000 Kbytes, y necesariamente se tendrá que incrementar. El parámetro SHMADD, controla la cantidad de memoria compartida que es añadida cuando el sistema se levanta, y mas memoria virtual compartida, es

requerida. El tamaño por defecto es de 8,000 Kbytes, lo cual significa que en alguna ocasión, una sesión necesitará memoria y no podrá extraer del segmento ya existente, por lo que otro segmento de 8,000 Kbytes será añadido.

Como la configuración de parámetros afecta la memoria compartida:

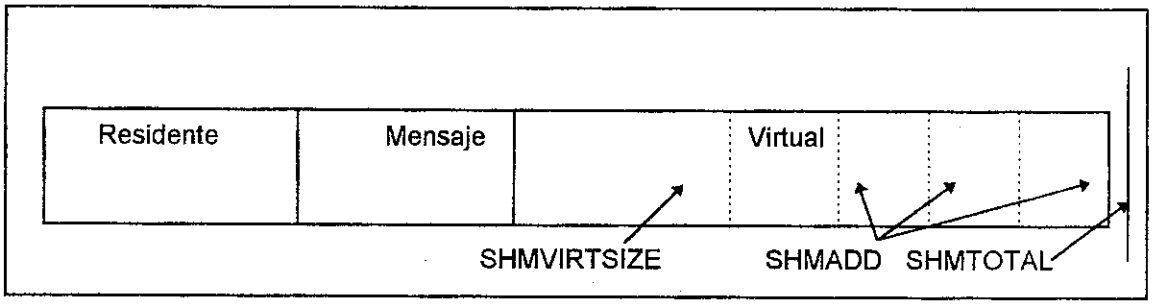


Figura 27 Visualización de como puede afectar los parámetros en la memoria compartida de Informix. FUENTE: Informix On-Line Performance Tuning - Elizabeth Suto

Uno de estos segmentos ha sido reservado, pero no puede ser eliminado, a menos que se baje el servidor del Informix OnLine. Se debe de notar que en general estas páginas de memoria, son tratadas como cualquier otra memoria. Si existe "pagineo", o "swapeo", entonces como cualquier área de memoria, tendrá problemas en el rendimiento.

Otros límites que se pueden agregar en el sistema "OnLine", son los siguientes:

Se puede decrementar la memoria usada, limitando el número de ordenamientos (sorts) que son ejecutados simultáneamente.

Se puede cambiar el tamaño de los buffers cache, con el parámetro de configuración "BUFFERS". Sin embargo con este parámetro se debe de tener cuidado, porque al disminuir su configuración, podría afectar el rendimiento del sistema, ya que se pueden producir una gran cantidad de lecturas y escrituras a disco.

5.1.4 Determinación de el uso de memoria en un sistema Informix OnLine v. 6.0

Suponiendo que se estima un crecimiento de 50 usuarios en los próximos seis meses. Por lo que se requerirá mas memoria. Se debe de iniciar, examinando el uso actual del sistema "OnLine". La instrucción **onstat -g seg** muestra la siguiente información de la memoria compartida, la cual ha sido reservada y representa la máxima cantidad requerida desde que el sistema "OnLine" se inicio:

```
$ onstat -g seg

id   key          addr      size      ovhd class      blkused
     blkfree
769  1381451777   800000    42000384  1432  R         5123
     4
770  1381451778   300e000   61440000  1520  V         7442
     58
771  1381451779   6aa6000   311296    558   M         35
     3
516  1381451780   6af2000   16777216  836   V         1950
     98
517  1381451781   7af2000   16777216  836   V         753
     1296
```

Figura 28 Ejecución y salida del comando "onstat -g seg"

Si se suman todas las entradas del campo de tamaño, entonces dará 133,814 Kbytes, esto es como 130 Megabyes. Este número representa la cantidad de memoria compartida que ha sido reservada para el servidor de la base de datos. Se debe de notar sin embargo, que 130 Megabytes han sido reservados, y que muy probablemente esta cantidad este provocando pagineo a disco.

Para encontrar, que tanta memoria compartida está siendo reservada para sesiones, y para tareas de carga general (overhead), se debe de sumar la columna "blkused" y multiplicar por ocho, para obtener el número aproximado de "Kbytes" que actualmente se están usando, en este caso es 122,424. Este valor representa, la cantidad total de memoria compartida, que esta siendo necesitada en promedio, para las sesiones que actualmente están corriendo.

Se puede ejecutar la sentencia **onstat -g seg** y llevar a cabo este calculo varias veces al día, para obtener un ejemplo mas representativo, de la cantidad de memoria compartida que se esta usando a diferente hora.

Luego de esto, se debe de determinar el tamaño del proceso **oninit,** con el comando ps de unix.

```
USER PID  #CPU #MEM SZ    RSS    TT  STAT START      TIME
COMMAND
root 22848     82.1 34.2 320   126948 p2 R    May 9
1385:24 oninit
root 22851     59.1 33.2 308   123112 p2 R    May 9
1009:57 oninit
root 22850     56.3 33.0 308   122704 p2 R    May 9
935:12 oninit
root 22849     0.0  0.3  296   1272   p2 S    May 9
3:43 oninit
root 22853     0.0  0.1  372   352    p2 S    May 9
6:36 oninit
root 22855     0.0  0.4  308   1432   p2 S    May 9
2:50 oninit
root 22852     0.0  0.4  308   1444   p2 S    May 9
2:53 oninit
root 22854     0.0  0.6  320   2396   p2 S    May 9
3:06 oninit
```

Figura 29 Salida de la sentencia "ps" de Unix.

El tamaño de los datos y de la pila (stack) de cada proceso oninit, es aproximadamente 320 Kbytes. Multiplicado por ocho procesos oninit, da un total de 2,560 Kbytes.

Para ejecutar el comando size en el proceso de oninit, se debe de medir el tamaño de la porción de texto del proceso.

```
$size $INFORMIXDIR/bin/oninit

text      data      bss       dec       hex
2482176   237568    27552           2747296   29eba0
```

Figura 30 Ejecución y salida del comando "size" de Informix.

El tamaño del texto entonces es de 2,482,176 bytes.

A continuación se llenara la siguiente tabla:

	Texto compartido	(Pila/Datos) (Por proceso)	Memoria Compartida
Proceso oninit	2424 Kbytes	320 Kbytes	
memoria compartida			122424 Kbytes
Total de Memoria	2424 Kbytes	2560 Kbytes	122424 Kbytes

Tabla 2 Estimación del total de memoria reservada para Informix OnLine.

La cantidad total de memoria que se usa actualmente es 2,424 + 2,560 + 122,424 = 127,408 Kbytes, aproximadamente 124 Mbytes de memoria.

En este momento, se debe de estimar que tanta memoria en promedio, están reservando los usuarios. El comando onstat -g ses, lista todas las sesiones activas. Para la salida de este comando, se pueden obtener unos cuantos identificadores de sesiones, y ejecutar el comando para cada sesión:

```
onstat -g ses session_id
```

La respectiva salida, lista entre otras cosas, la memoria compartida que esta actualmente reservando para la sesión. Un ejemplo de dicha salida es la siguiente:

```
Memory pools count 2
name class       addr     totalsize freesize  #allocfrag
     #freefrag

2117 V       35fa010 262144    48908         295
     53
2117 V       3694010 352256    42044         7
     6_SORT_0
```

Figura 31 Ejemplo de salida de la instrucción "onstat -g ses session_id"

Esta sesión esta actualmente ejecutando un sort, por lo que se debe de notar que el SORT Pool está siendo reservado. La cantidad total de memoria para el área de esta sesión es: 262144 + 352,256 = 614,400 bytes.

```
Memory pools count 2
name class       addr     totalsize freesize  #allocfrag
     #freefrag

2810 V       574e010 253952    47044         295
     53
2810 V       3f40010 303104    28748         7
     _SORT_0
```

Figura 32 Ejemplo de "onstat" en diferente periodo de tiempo

Esta sesión ha reservado 253,952 + 303,104 = 544 Kbytes.

Después de examinar la salida de esta sesión, diferentes veces al día, se puede concluir que 500 Kbytes por usuario, no es una cantidad razonable

durante horas pico del día. Para 50 nuevas sesiones, ejecutando aplicaciones similares, se necesitarian 50 * 500 Kbytes, que es aproximadamente 25 Megabytes de memoria para horas pico.

5.1.5 El proceso de Lecturas a memoria y a disco

En el sistema OnLine, las lecturas de información a la base de datos, se hacen en unidades de página (aproximadamente de 2 a 4 Kbytes dependiendo del sistema operativo). Las lecturas pueden ocurrir, no solo con sentencias SQL Select a la base de datos, sino que también por sentencias DML como actualizaciones (Updates) y borrados (Deletes), ya que el sistema OnLine, debe de leer las páginas con datos, antes de poder ser borradas o actualizadas.

Si la página necesitada esta en memoria (el los buffer cache), esta será leída desde la memoria, de lo contrario será leída de disco, y puesta en los buffers cache de memoria compartida. Debido a que los buffers están e memoria compartida, varios usuarios pueden accesar la misma página. Mientras mas a menudo, los datos sean leídos de memoria, y no de disco, la razón de lecturas a cache será mas alta, que es lo que se persigue.

5.1.6 Monitoreo de la razón de lecturas a "cache"

Se puede monitorear, la razón de lecturas a cache, ejecutando el comando **tbstat -p** (**onstat -p** para la versión 6.0). El primer campo llamado **%cached** muestra la razón de lecturas de cache, desde el sistema OnLine fue

levantado, o desde que las estadísticas fueron limpiadas con el comando **tbstat -z** (**Onstat -z**). En el ejemplo de abajo, se nota que la razón de lecturas a cache es del 99.48%, el cual es un buen porcentaje.

```
Profile
dskreads  pagereads bufreads  %cached   dskwrits  pagwrits
     bufwrits  %cached

2383 2389 453962     99.48     14776     20943     77792
     81.01
```

Figura 33 Monitoreo de la razón de lecturas a cache de Informix

La razón de cache es calculado por el reporte, pero también se puede calcular usando la fórmula:

"100 * (bufereads - dskreads) / bufreads"

La razón de lecturas de buffer cache, puede varias de manera dramática, depentiendo de las aplicaciones y del tipo y tamaño de dato que esté siendo operado. Si los usuarios están accesando los mismos datos constantemente, la razón de lecturas cache pueden potencialmente ser muy altos. Si los datos accesados, son bastante aleatorios, y la base de datos muy grande, la razón de lecturas cache puede ser muy baja, o sea entre el 70 y el 80 por ciento.

Ya que se ha monitoreado, entonces, se pueden realizar una serie de pasos que aseguren que la razón de lecturas a cache es optima para el sistema OnLine.

5.1.7 Afinamiento de lecturas a "cache"

Se puede afinar inicialmente la razón del "cache", incrementando o decrementando el número de "buffers" de memoria compartida reservada para los buffers cache, esto se hace con el parámetro de configuración "BUFFERS". Al incrementar el "cache buffers", se esta permitiendo que mas páginas estén en memoria. Sin embargo, no es recomendable añadir buffers en gran cantidad, ya que se corre el riesgo de eliminar memoria para otros procesos, como los de usuarios (versión 6.0), o algunos del sistema operativo. Lo mas recomendable, es tomar una decisión de cuantos buffers incrementar hasta que se note que la razón de lecturas a cache pare de incrementarse dramáticamente, o llegue al límite de la memoria por sistema operativo. Para afinar "cache", se deben seguir los siguientes pasos:

1. Re-Iniciar las estadísticas nuevamente con el comando **tbstat -z (onstat -z para V. 6.0)**
2. Correr varias horas, o días, que sean típicos de producción. Afinar la razón de cache durante esos días.
3. Registrar la razón de "cache" en porcentaje por medio del **tbstat -p (onstat -p)**
4. Incrementar el parámetro de configuración de "buffers" y bajar el sistema OnLine, luego hacer "backup".
5. Repetir los pasos del 2 al 4 hasta que no halla incremento de la razón de lecturas de "cache".

5.2 Manejo de disco

Las lecturas y escrituras a disco, son operaciones catalogadas de costo alto, en cualquier sistema computacional. Debido a esto, los esfuerzos del DBA para manejo de disco deben ir orientadas a:

- Tratar de evitar esta operación tanto como sea posible.

- Si las operaciones de Entrada/Salida son necesarias, entonces tratar de distribuirlas de manera equitativa a través de los discos.

Algunos asuntos que serán tratados en esta sección son:

- Monitoreo, y afinamiento de escrituras a disco.
- Consideraciones de arreglos de disco.
- Afinamiento de disco en la versión 6.0.

5.2.1 Monitoreo de escrituras a memoria y a disco

Existen un conjunto de procesos, los cuales son llamados "Limpiadores de páginas" (page cleaners), que son responsables de escribir páginas desde los buffers a disco. En la versión 6.0, los limpiadores de páginas son apuntadores (threads), y ellos pasan los requerimientos de escritura de buffers a los procesadores virtuales Asincronos de Entrada y Salida (AIO vp) los cuales limpian páginas, y hacen este trabajo durante un punto de verificación (checkpoint) o entre puntos de chequeo si los buffers cache están demasiado

llenos de páginas 'sucias' (páginas que han sido cambiadas desde que fueron leídas de disco).

Los buffers de páginas están organizadas en un conjunto de listas encadenadas, llamadas listas LRU (Menos recientemente usadas). Cada cola esta sub-dividida en una cola limpia y en una sucia. Las páginas en la cola de páginas limpias, están disponibles para ser sobre-escritas por otros usuarios. Las páginas en la cola de páginas sucias son escritas a disco, antes de que puedan ser re-utilizadas. Cuando un proceso servidor de la base de datos, necesita leer una página de disco, este halla un buffer en el final de la LRU (menos recientemente usada) de la cola de las páginas limpias, y reemplaza su contenido con el contenido de las páginas leídas desde disco. Los "buffers" son colocados como las mas recientemente usadas, al final de la cola LRU. Este mecanismo es usado para mantener las páginas que son muy usadas, alejadas de las que serán sobre-escritas por nuevas páginas leídas desde disco.

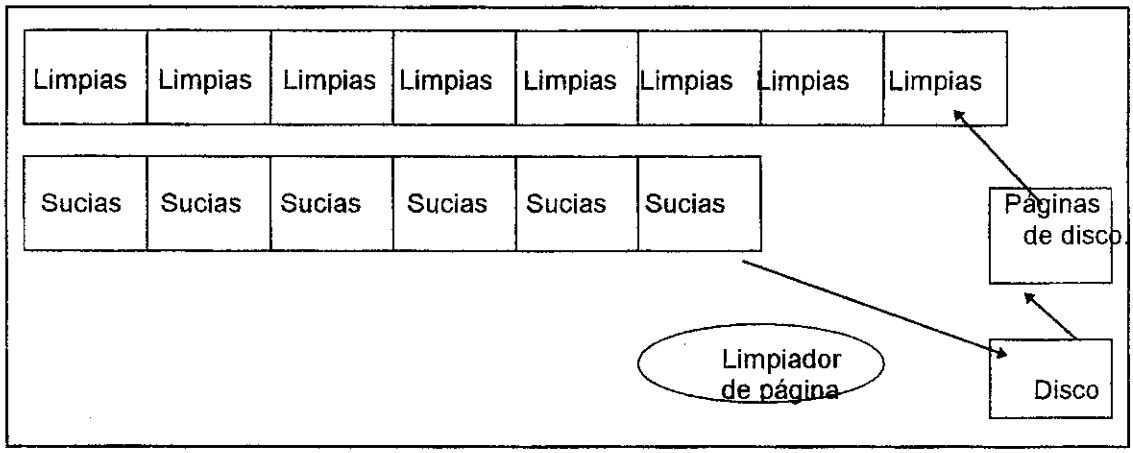


Figura 34 Diagrama del manejo de páginas de memoria en Informix OnLine.

Cuando las páginas son escritas en memoria, estas son marcadas como sucias, esto significa que su contenido ha sido cambiado y eventualmente serán escritas a disco. Durante un punto de chequeo, el cual no es mas que una sincronización entre lo que hay en memoria y lo que hay en disco, todas las páginas sucias serán leídas de memoria hacia sus respectivas localizaciones a disco.

Entre puntos de chequeo, el sistema OnLine, necesita proteger contra todo, a los buffers que se han vuelto sucios, ya que una vez estos buffer se han convertido en sucios, estos no pueden ser escritos hasta que el contenido sea bajado a disco. Ocasionalmente, las colas LRU son examinados por páginas sucias. Si un cierto porcentaje de páginas dentro de la cola LRU están sucias, el limpiador de páginas escribirá las páginas sucias a disco.

Según lo que se ha visto aquí, existen dos momentos en los cuales la páginas son escritas a disco: durante un punto de verificación, y entre puntos de verificación. La situación ideal, es que nunca se escriba a disco. Si se pudiera llevar operaciones de la base de datos en memoria, se podría tener la razón promedio del 100% para el mejor rendimiento. Existen dos elementos que no permiten realizar esto:

- la base de datos es probablemente mas grande que los "buffers cache",
- que la recuperación tomaría un largo período de tiempo si el sistema se cae.

Se podría afinar el sistema OnLine, para que la mayoría de páginas sucias sean escritas a disco durante los puntos de verificación. Esto podría incrementar la razón de "cache" en algo, además que se estará dando tiempo

potencial para que el proceso servidor de la base de datos escriba una página en memoria más de una vez. Si una página es modificada en memoria más de una vez, el total de la razón de caché mejorará, y potencialmente, todo el rendimiento del sistema OnLine también lo hará. El único problema en esto, es la manera en que se hacen los puntos de verificación. Cuando un punto de verificación inicia, todos los procesos servidores de la base de datos están prevenidos para ingresar a una "sección crítica" del código, esto significa que deberán esperar a realizar cualquier cosa, mientras se escriben las páginas. Si el punto de verificación es largo, esto tendrá serias implicaciones en el rendimiento.

En algún momento se podría querer escribir a disco entre puntos de verificación, de manera que hay algunas situaciones que son validas, y deben tomarse en cuenta:

Si todo el "caché buffers" no está en memoria (o sea cargada en todas las páginas) entonces se podrá alterar entre puntos de verificación, de manera que escrituras podrían ser hechas. También, desde que un punto de verificación finaliza, todas las escrituras desde que estas inician hasta que el punto de verificación es completado, deben de verse como un retardo en el procesamiento, si hay un gran número de páginas sucias que deben ser escritas.

En la siguiente figura podemos observa (caso a) como la actividad de los discos es mostrada cuando la mayoría de escrituras suceden. Se puede notar que los dispositivos de discos muestran un conjunto de picos que demuestran una actividad errática entre puntos de chequeo. En la otra gráfica (caso b) se

puede ver que el sistema se comporta mas establemente, debido a que las escrituras son realizadas entre puntos de chequeo.

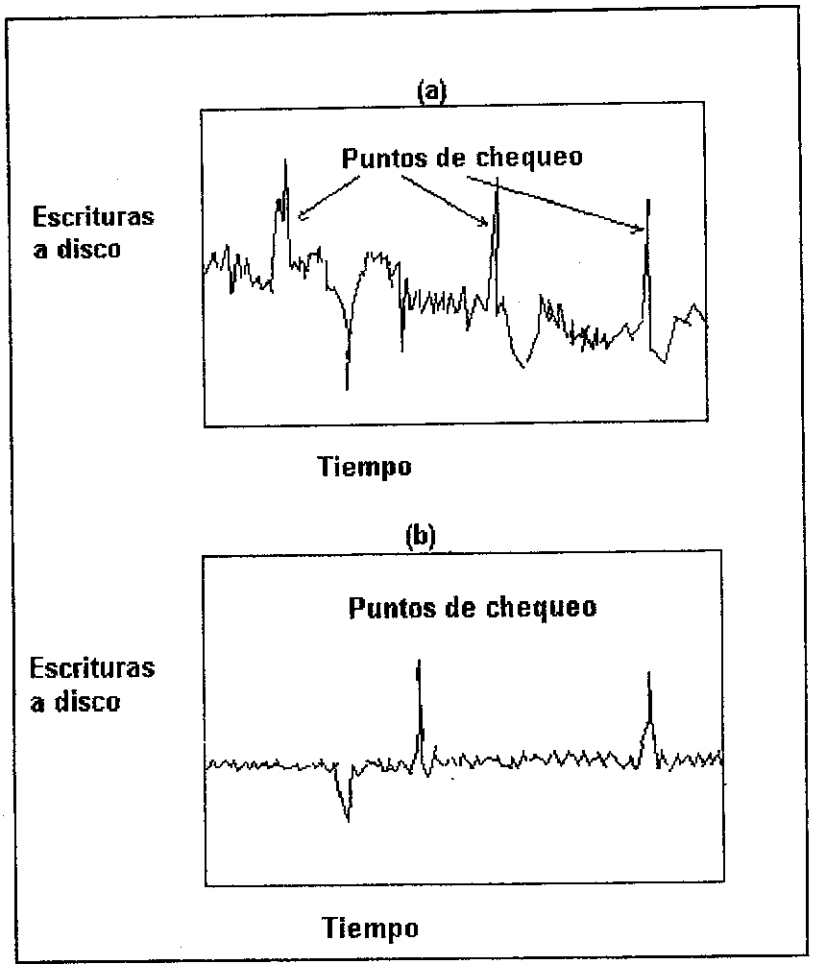


Figura 35 Escrituras a disco en distintos puntos de chequeo.

Si se distribuye la actividad de disco entre puntos de verificación, entonces se podrá observar un mejor tiempo de respuesta para los usuarios. También se notará de que el rendimiento mejora debido a que los puntos de verificación son más frecuentes y por lo tanto no habrá retardo en la actividad de transacciones. Sin embargo, la razón de la escritura a "caché" también puede decrementarse, porque las escrituras de páginas serán mas frecuentes a disco.

5.2.1.1 Afinando escrituras a disco

Para afinar esto, es necesario el conocimiento de una serie de parámetros de configuración, y además tener el conocimiento de como y cuando se deben de escribir las páginas a disco. Esta configuración se explica a continuación.

5.2.1.1.1 Intervalos de puntos de verificación

Un intervalo de puntos de verificación, es el número de segundos que existen entre los puntos de verificación (checkpoints). Existen varias formas de afinar los puntos de verificación.

1. El parámetro de configuración llamado CKPTINTVL es colocado al valor por "default". De acuerdo a la experiencia, se recomienda colocar su valor al máximo número de segundos entre los puntos de verificación.
2. Un punto de verificación puede ocurrir también cuando una bitácora física esta llena al 75%.

Se puede determinar la frecuencia de un punto de verificación, ejecutando el comando llamado **tbstat -m**. La opción de -m muestra las últimas líneas de los mensajes de la bitácora. Para ver el total, entonces se recomienda editar los mensajes de la bitácora.

Cada vez que ocurre un punto de verificación, una línea es escrita en la bitácora. Se puede calcular entonces el intervalo aproximado entre los puntos de verificación realizando una substracción. Por ejemplo:

```
Message Log File:  /home/informix/5.00/online.log
          15:49:36  Checkpoint Completed
          15:54:48  Checkpoint Completed
```

Figura 36 Ejemplo de la sentencia "tbstat -m" para monitoreo de distintos puntos de chequeo en el tiempo.

En este ejemplo, la frecuencia es de cinco minutos.

En lugar de configurar el parámetro del intervalo de puntos de chequeo CKPTINTVL, se considerará manejarlo por medio del tamaño de archivo de bitácora. Por ejemplo, si se configura el parámetro CKPTINTVL a un número algo largo como 15 minutos, y el archivo de bitácora físico también se configura para que un punto de chequeo ocurra cada 10 minutos con una actividad promedio. Esto causaría que los puntos de chequeo ocurran cuando no existe suficiente actividad en la base de datos, y esto implicaría que no es el intervalo adecuado.

Se puede realizar la verificación de la bitácora actual, observando el parámetro de inicialización llamado PHYSFILE. Se puede incrementar el tamaño de la bitácora ejecutando el utilitario **tbparams**, así:

```
tbparams -p -s tamaño-bitácora
```

Si el intervalo del punto de verificación es grande, esto puede hacer que el proceso de recuperación se alargue. Si por alguna razón el sistema de Informix "On-Line" se cae, todas las páginas de la bitácora tendrían que ser ejecutadas (aplicadas). Con un archivo de bitácora largo, este proceso entonces podría tomar desde varios segundos hasta 20 minutos, dependiendo de que tan largo sea este proceso de recuperación. Se recomienda que el intervalo de puntos de verificación sea el suficiente para que el proceso de recuperación este entre los 5 y los 15 minutos.

5.2.1.1.2 Colas de política menos recientemente utilizado (LRU Queues)

Estas colas también pueden ser configuradas, ya que generalmente los usuarios accesan estas colas para leer páginas, esto implica que deben de haber suficientes colas para evitar en la medida de lo posible la contención de estas colas.

Normalmente, estas colas son accesadas por apundadores (threads) que están siendo ejecutados por los procesadores virtuales de CPU. Esto significa que la posibilidad de que exista contención, depende de la cantidad de procesadores virtuales que estén configurados. Inicialmente se recomienda que se realice una configuración de por lo menos cuatro colas.

5.2.1.1.3 Procesos limpiadores de páginas

Estos procesos son los responsables de recolectar las páginas modificadas, y salvar los respectivos requerimientos de escritura al sub-sistema

de acceso de E/S asíncrono. Durante un punto de chequeo, cada limpiador de página es asignado a uno o mas "chunks". Las páginas son inicialmente ordenadas en el "chunk" y luego se localiza el lugar adecuado en el "chunk"; por último son escritas. Entre puntos de chequeo, los limpiadores de página son asignados a las colas de política menos recientemente usadas, y entonces los limpiadores de páginas escriben las páginas sucias en orden aleatorio desde las colas de políticas menos recientemente utilizadas.

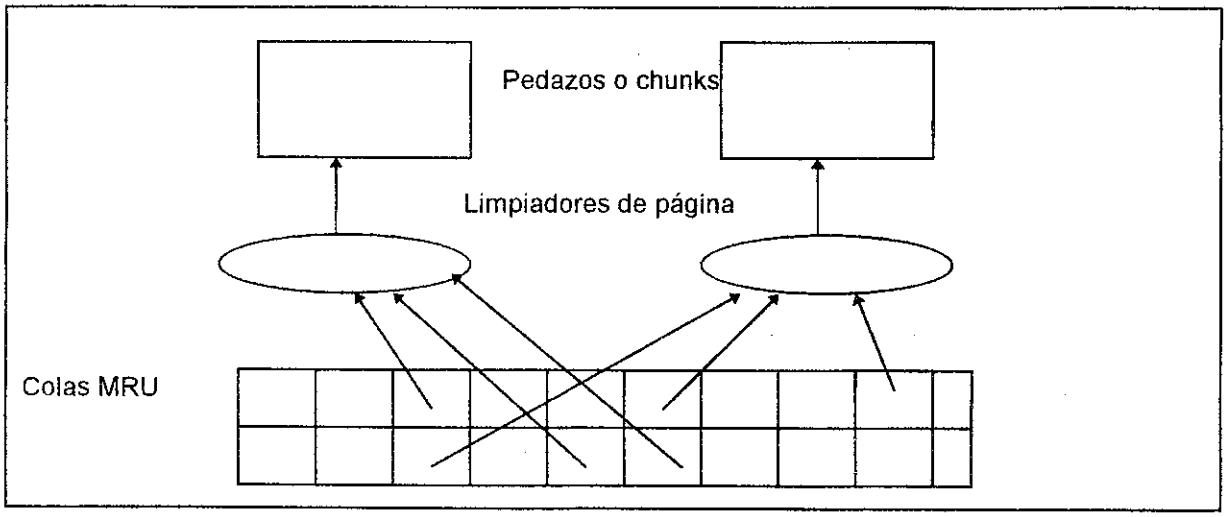


Figura 37 Diagrama de la actividad de los procesos limpiadores de páginas

El número necesario de limpiadores de página para un sistema en Informix, probablemente disminuya, debido a que existe poca contención para las colas menos recientemente utilizadas debido a la cantidad de procesadores virtuales. Debido a estos, se puede configurar el parámetro "CLEANERS" al número de discos que serán actualizados durante un punto de chequeo. Sin embargo, si las colas de política menos recientemente utilizadas están

configuradas a un valor mayor que el número de discos, entonces el parámetro de "CLEANERS" se debe de configurar con el mismo valor. El hecho de tener el parámetro CLEANERS mayor al de las colas, no afecta el rendimiento.

5.2.1.1.4 Parámetros de configuración LRU

Existen otros parámetros que pueden ser usados para determinar cuando y como grandes cantidades de escrituras deben ser realizadas entre puntos de chequeo. El parámetro LRU_MAX_DIRTY es el número máximo de páginas que deben estar sucias (el termino de sucias implica que han sido actualizadas) antes que los limpiadores de páginas las limpien (limpiar implica bajarlas a disco, hacer efectivos los cambios). También esta el parámetro LRU_MIN_DIRTY, que es el porcentaje de la cola que los limpiadores de página dejará como sucias.

Por ejemplo, si el parámetro llamado LRU_MAX_DIRTY tiene un valor de 80 y el LRU_MIN_DIRTY tiene un valor de 60, entonces el limpiador de página iniciará su actividad cuando el 80% de la cola contenga páginas sucias. El proceso limpiador de página se detendrá hasta que el 60% de la cola este llena de páginas sucias.

5.3 Rendimiento y uso del CPU

Algunos tópicos a tocar en esta sección son:

- Informix y los sistemas de multiprocesadores.
- Como monitorear el uso del CPU
- Cuales son las espectativas de rendimiento si se añaden más procesadores.
- Como afinar el uso del procesador en versión 6.0.

5.3.1 Informix y los sistemas de multiprocesadores

Informix-OnLine y el servidor dinámico de Informix-OnLine, pueden correr en sistemas mono-procesadores, o en una clase especial de sistemas multiprocesadores llamado sistema de multiprocesamiento simétrico o SMP. En un sistema SMP, los procesadores ejecutan código independiente, pero deben comunicarse con otros procesadores cada cierto tiempo. Un bus de alta velocidad es usado para conectar cada procesador a otros, y hacia el conjunto común de la memoria. El número de procesadores que pueden efectivamente se usados en un sistema depende de la carga en la cantidad de trafico entre procesadores que el bus puede manejar.

Procesamiento paralelo significa que una tarea pueden ser dividida en piezas y procesada en paralelo. Informix cumple con el procesamiento paralelo en la versión 6.0, dividiendo tareas a través de los procesadores, la cual puede dividir en los diferentes procesos del sistema SMP. El sistema OnLine actuará como un 'planificador', para asignar los diferentes apuntadores o "threads" a los procesadores virtuales. Sin embargo, es responsabilidad del planificador

del sistema operativo, asignar el procesador virtual (el cual esta simplemente procesado) al procesador de hardware.

5.3.2 Como monitorear el uso de CPU

En la mayoría de UNIX (del tipo BSD), se pueden monitorear el uso del CPU básicamente con el comando **vmstat**. Un ejemplo de esto podría ser:

```
procs  memory                    page                            disk
faults        cpu
r b w   avm   free     re   at   pi   po   fr   de   sr   d0   d1   d2
d3   in   sy   cs   us   sy   id

1 0 0   0     28688    0    6    0    1    2    0    0    0    4    2
5    46   254   116  5    2    93
1 2 0   0     28636    0    0    0    0    0    0    0    0    31   18
31   282  1373  674  27   11   62
1 3 0   0     28544    0    0    0    0    0    0    0    0    28   16
35   321  1286  699  29   13   58
```

Figura 38 Salida de la instrucción "vmstat" de Unix para monitoreo de CPU

Las columnas que reportan la información de la actividad de CPU son las últimas tres:

La columna "us" muestra el porcentaje del tiempo en que el CPU estuvo ejecutando procesos en el estado de usuario.

La columna "sy" muestra el porcentaje de tiempo en que el CPU estuvo ejecutando procesos en el estado del sistema (actividad del kernel y otra sobrecarga del sistema).

La columna "id" muestra el porcentaje de tiempo en que el CPU estuvo ocioso.

En este ejemplo, el sistema esta bien, ya que el CPU tiene entre el 62% y el 58% de tiempo ocioso.

Otra pieza importante de la información, es la primera columna, la cual despliega la cola de ejecuciones. Las estadísticas de la cola de ejecuciones, es un contador del número de procesos que están listos para ser ejecutados, pero que están esperando, generalmente por disponibilidad del CPU.

Se puede observar la actividad de los procesadores en el System V de Unix con el comando
sar -u:

```
00:00:00    %usr    %sys    %wio    %idle
  01:00:01    20       9       2       69
  02:00:01    22       8       3       67
  03:00:01    25       9       2       64
```

Figura 39 Ejemplo de salida de la sentencia "sar -u" para observar la actividad del procesador

Si el valor %wio es significante para esta salida, su sistema actualmente esta ligado a E/S, aún si el %idle es bajo. Algo conveniente es investigar el rendimiento de su disco, en este caso.

En Unix system V, el comando para monitorear las colas de ejecución es el "**sar -q**".

Para sistemas multiprocesadores, es de gran ayuda, monitorear la actividad individual de cada procesador. La mayoría de sistemas multiprocesadores, tienen utilitarios que hacen esto. Por ejemplo, en máquinas multiprocesadoras de SUN Microsystems, tienen el utilitarios "**mpstat"**, el cual da la utilización del CPU por procesador.

5.3.3 Como ayuda al rendimiento si se añade otro procesador

En la versión 6.0 de Informix, esta diseñado para trabajar mas eficientemente, con máquinas multiprocesadoras, corriendo con muchos usuarios. Los procesadores virtuales, deberán hacer la mayoría de trabajo intensivo en el CPU para los múltiples usuarios o sesiones.

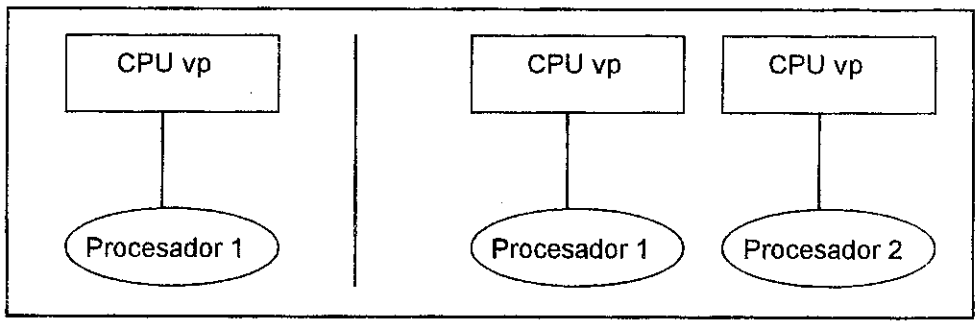


Figura 40 Existencia de CPU's virtuales para cada procesador físico.
FUENTE: Informix On-Line Performance Tuning - Elizabeth Suto

Con la arquitectura multienlazado ("multithreaded") de versión 6.0, cada sesión puede tener múltiples apuntadores (threads). Cada apuntador puede correr en diferentes procesos virtuales de CPU, efectivamente paralelizando una actividad de usuario individual. En realidad, la versión 6.0 divide múltiples apuntadores para las siguientes actividades:

- Ordenamientos ("Sortings")
- Construcción de Indices
- Restauración de la Base de Datos.

Las actividades paralelas podrían ser mas rápidas, si se añaden mas procesadores (y CPU vp). Cada apuntador puede correr en diferentes CPU procesadores virtuales (vps), y cada CPU vp, puede correr en diferentes procesadores. Esto significa que un ordenamiento de sesión o una sentencia "CREATE INDEX" puede correr en múltiples procesadores a la vez.

Sin embargo, para otras operaciones la actividad de la sesión solo será tan rápida como en un sistema mono-procesador, similar a la versión 5.0. Sin embargo, en las ultimas versiones del servidor dinámico OnLine, múltiples apuntadores serán divididos para selecciones de la base de datos, efectivamente paralelizando la sentencia SELECT para una sesión.

Para la versión 6.0, la sobrecarga de un procesador será evidente solo por el tiempo ocioso del CPU. La cola de ejecuciones de un sistema para la versión 6.0 del sistema OnLine, podría estar cerca de cero, porque los procesadores virtuales CPU están únicamente procesando dentro del sistema OnLine con un alto uso el procesador.

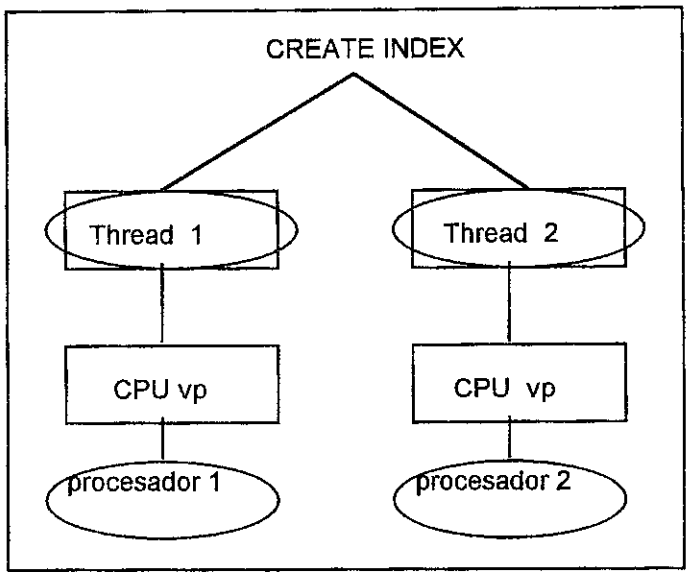


Figura 41 Ejemplo del manejo de una creación de índices con múltiples procesadores. FUENTE: Informix On-Line Performance Tuning - Elizabeth Suto

5.3.4 Como influenciar el uso del procesador en versión 6.0

En esta versión, se tiene mayor control del uso del procesador, ya que existen varios parámetros que pueden ser afectados, para determinar el número de procesadores a usar por el servidor de la base de datos.

El proceso OnLine que usa la mayoría de tiempo de procesador en 'estado - utilizado' .

El proceso virtual de acceso de entrada/salida asíncrono, usa algún tiempo de procesador, pero en su mayoría es tiempo del sistema.

La mayoría de parámetros afinables que afectar el uso del procesador es el "Número de CPU virtuales" (esto se hace colocando el parámetro de configuración NUMCPUVPS). Normalmente las máquinas virtuales de CPU, deben de estar en trabajo intensivo de CPU en los sistemas OnLine. Generalmente, no debería de existir mas de un proceso virtual de CPU para cada procesador.

La mejor manera de determinar si hay suficientes procesos virtuales de CPU, es chequear el tamaño de la 'cola de listos' del sistema OnLine. La cola de listos del sistema OnLine (no se confunda con la cola de listos del sistema) es una lista de todos los apuntadores que están listos para correr, pero por la carencia de un procesador que le de servicio, aun no a iniciado a correr. Lo ideal es que esta cola esté vacía, o casi vacía la mayoría del tiempo. Se puede monitorear esto por medio del comando "**onstat -g rea**" . Una salida puede ser como la siguiente:

```
Ready threads:
tid  tcb  rstcb     prty status    vp-class  name

13   b53120    0    2    ready     1cpu
     sm_discon
14   b5aed8    8067c4    2    ready     1cpu
     flush_sub(0)
44   ca09a4    809dc4    2    ready     1cpu
     sqlexec
45   ca946c    809a64    2    ready     1cpu
     sqlexec
46   cb69a4    809704    2    ready     1cpu
     sqlexec
```

Figura 42 Monitoreo de la "Cola de procesos listos" por el comando "onstat -g rea"

En el ejemplo anterior, existen tres apuntadores de usuario, y dos del sistema que están esperando para que un proceso virtual de CPU continúe trabajando.

Si se está utilizando una máquina con un simple procesador, realmente no hay mucho que se pueda hacer sobre una cola de procesos listos, ya que constantemente tiene entradas. En una máquina multiprocesadora, se puede incrementar el número de procesos virtuales de CPU por uno (hasta que se alcance el número de procesadores en su sistema) y continúe monitoreando la cola de listos, de manera de verificar si el procesador añadido, ayuda al sistema. Se debe, sin embargo de tener cuidado en este punto, ya que si se detecta que la utilización de CPU para todos los procesadores esta entre el 80% y el 100%, añadir otro procesos virtual de CPU, dañaría el rendimiento.

Para añadir un procesador virtual de CPU dinámicamente, se hace lo siguiente:

```
onmode -p +1 cpu
```

Para eliminar un procesador virtual de CPU dinámicamente, se hace lo siguiente:

```
onmode -p -1 cpu
```

Añadir o borrar procesadores virtuales de CPU dinámicamente, no cambia el número de estos, hasta que se inicie nuevamente el sistema OnLine. Una vez que se ha determinado el número óptimo de procesadores virtuales de CPU, se debe modificar el parámetro de configuración NUMCPUVPS, o cambiar el valor en la opción del menú del onmonitor.

5.3.5 Otras consideraciones

Además de alterar el número de procesadores virtuales de CPU, también existen algunas otras consideraciones, que podrían mejorar el rendimiento en su sistema OnLine:

Modificar el parámetro MULTIPROCESOR a 1 para máquinas multiprocesadoras. Este parámetro permite alterar el sistema OnLine para que use el mecanismo "Spin Lock para los apuntadores". Este mecanismo permite colocar los apuntadores desde el inicio en la cola de dormidos, o en la cola de espera, mientras esta esperando por tiempo del respectivo recurso. En un sistema mono-procesador, el valor de este parámetro debería ser 0.

Colocar el parámetro SINGLE_CPU_VP a uno para un sistema mono-procesador. Este pertenece al sistema OnLine, para utilizar un solo proceso virtual de CPU, y además de esto, también apaga algunos mecanismos de bloqueo, para que no halla contención de recursos, como con dos procesadores virtuales.

La carencia de estos candados internos, tienen un efecto positivo en el rendimiento, que aún sistemas con dos procesadores, en ocasiones se prefiere configurar este parámetro a 1. Sin embargo, esto significa que solo un procesador virtual de CPU puede ser iniciado. Para sistemas de mas de dos procesadores, se debe colocar el parámetro SINGLE_CPU_VP a 0.

Usar el ordenamiento paralelo. Los sistemas multiprocesadores, generalmente se benefician de características de ordenamiento paralelo. En la versión 6.0, esto significa que se manejan múltiples apuntadores para un único

requerimiento de usuario. El ordenamiento por apuntadores, puede ser ejecutado en paralelo, del cual se pueden tomar ventaja de múltiples procesadores. Sin embargo, en un sistema altamente cargado, el ordenamiento paralelo, puede hacer lentas otras operaciones, esto es debido a que los apuntadores son planificados con los mismos procesos virtuales de CPU, que están también realizando otras operaciones. Si la cola de listos en el sistema OnLine, esta constantemente cargada (onstat -g rea), se recomienda apagar el ordenamiento paralelo, para ayudar en el rendimiento. El parámetro que permite hacer esto por sesión es el PSORT_NPROCS.

Ajustar la afinidad del procesador. La afinidad de procesador permite a un proceso atarse a un solo procesador. Para mantener a un proceso en un procesador, se debe de tomar ventaja del cache del procesador y prescindir de la sobrecarga requerida para mover un proceso desde un procesador de hardware a otro. Generalmente, se deseará configurar la afinidad en procesadores virtuales de CPU, para que estos realicen una gran cantidad de trabajo. En los sistemas OnLine, existen dos parámetros de configuración que pueden cambiar la afinidad del sistema: AFF_NPROCS y el AFF_SPROC.

El AFF_NPROCS especifica el número de procesadores de hardware, los cuales serán asignados a procesos virtuales de CPU. Por ejemplo, si AFF_NPROCS es 2, entonces dos procesadores correrán todos los procesos virtuales de CPU. Generalmente se configurará el AFF_NPROCS al número de procesos virtuales de CPU en el sistema OnLine, y un número menor al total de procesadores existentes en hardware. El parámetro AFF_SPROC, especifica al inicio del procesador, el número de afinidad (iniciando en 0). Esto permite atar ciertos procesadores con ciertos procesos virtuales de CPU.

6. ORACLE

6.1 Afinamiento de memoria

6.1.1 Afinamiento del "shared pool"

Esta sección describe el afinamiento de las partes del "shared pool".

- Caché de librerías (Library cache)
- Caché del diccionario de datos (Data dictionary cache)
- Información de sesión

El orden que a continuación se presenta es importante para su respectiva ejecución. Esto es debido a que el algoritmo que Oracle utiliza, maneja los datos en el "shared pool", y por lo general dicha cantidad de datos es mayor a la que se almacena en el caché del diccionario de datos; Por lo que un "cache hit ratio" del caché de librerías, a menudo asegura que el "cache hit ratio" del caché del diccionario también sea aceptable. La reservación de espacio en el "shared pool" para información de sesión, es necesario únicamente cuando se usa la arquitectura de multi-thread server.

6.1.1.1 Afinamiento del "cache" de librerías

El caché de librerías contiene las áreas de SQL y PL/SQL que son compartidas (en memoria compartida por los procesos de Oracle). A continuación se indicará como se hace el afinamiento realizando las siguientes tareas:

- Examinando la actividad del caché de librerías.
- Reduciendo las perdidas del caché de librerías
- Aumentando la velocidad para el acceso de las áreas compartidas de SQL y PL/SQL en el caché de librerías.

6.1.1.1.1 Examinando la actividad del "caché" de librerías

Las perdidas (o misses) en el caché de librerías pueden ocurrir en cualquiera de los siguientes pasos en el procesamiento de sentencias SQL:

Parseo Si una aplicación realiza una llamada de parseo para cualquier sentencia SQL, y la respectiva representación de parseo de la sentencia no existe en el área compartida del caché de librerías, Oracle parseará la sentencia y reservará un área compartida para el SQL. El objetivo, es reducir la cantidad de veces en que una llamada de parseo no es encontrada en el library cache tanto como sea posible (esto es una perdida o miss).

Ejecución Si una aplicación realiza una llamada de ejecución para una sentencias SQL, y el área compartida de SQL, contiene la representación de la

sentencia ya parseada, la cual ha sido desalojada del caché de librerías, para obtener espacio para otra sentencia, Oracle, implícitamente reparseará la sentencia, reservando una nueva área para dicha sentencia, y ejecutará esta. La forma para reducir las perdidas en el caché de librerías en llamadas de ejecución, es reservando mas memoria a este caché.

Para determinar si las perdidas en el caché de librería están afectando el rendimiento de la base de datos Oracle, se puede examinar la tabla dinámica llamada V$LIBRARYCACHE. Estas estadísticas reflejan la actividad del caché de librería desde la última vez que se levanto la base de datos.

Cada tupla de esta tabla contiene estadísticas un tipo específico de objeto que es mantenido en el caché de librería. El objeto descrito por cada tupla, es identificado por el valor de la columna "NAMESPACE". Las tuplas de la tabla con los valores de "NAMESPACE" reflejan la actividad del caché de librería, para los bloque de las sentencias SQL y PL/SQL:

- SQL AREA
- TABLE/PROCEDURE'
- BODY
- TRIGGER

Los registros con otros valores de NAMESPACE reflejan la respecttiva actividad del caché de librería para la definición de objetos que son usados por Oracle para mantenimiento de dependencia.

Estas columnas de la tabla V$LIBRARYCACHE reflejan las pérdidas en el caché de librerías sobre llamadas de ejecución:

PINS Esta columna muestra el numero de veces que un elemento en el caché de librería fue ejecutado.

RELOADS Esta columna muestra el número de perdidas en el caché de librerías, en el paso de ejecución.

Una selección (query) utilizada para examinar las estadísticas en la tabla V$LIBRARYCACHE en periodos de tiempo puede ser:

```
SELECT SUM(pins) "Ejecuciones",
SUM(reloads) "Perdidas de cache en ejecución"
FROM v$librarycache;
```

Figura 43 Instrucción de selección usada para monitorear el caché de librerías

La salida de esta selección puede ser parecido a la siguiente:

```
Ejecuciones   Perdidas de "cache" en ejecución
----------    ----------------------------------------
320871                                          549
```

Figura 44 Ejemplo de la salida de la instrucción anterior

Al interpretar los valores de la tabla v$librarycache, podemos examinar los datos retornados y sacar las siguientes deducciones:

- La columna resultante de sumar todos los valores de "PINS" indica que las sentencias SQL, bloques PL/SQL, y definición de objetos fueron accesados por ejecuciones en un total de 320,871 veces.

- La suma de la columna "RELOADS" indica que 549 de estas ejecuciones resultan en perdidas del caché de librerías, causando que Oracle implícitamente reparsee la sentencia o bloque o vuelva a cargar una definición de objeto porque ha estado mucho tiempo en el "cache" de librerías..

El radio del total de "RELOADS" al total de "PINS", es cerca de 0.17%. Este valor significa que solo un 0.17% de las ejecuciones resultan en re-parseos.

El total de "RELOADS" deberían ser cercano a 0. Si el radio de "RELOADS" y "PINS" es cercano al 1%, entonces se recomienda reducir dichas perdidas en los caché de librerías, haciendo lo siguiente:

- Añadiendo mas memoria al caché de librerías.
- Escribiendo sentencias SQL idénticas, tantas como sea posible.

Reservando memoria adicional al caché de librerías. Con esto se puede reducir la perdidas en el caché de librerías en llamadas de ejecución. Se debe de aumentar al área de caché de librería hasta que el valor de los "RELOADS" sea cercano a 0. Para hacer esto, se requiere incrementar el valor en la variable de inicialización SHARED_POOL_SIZE. El máximo valor en este parámetro es dependiente del sistema operativo.

Para tomar ventaja de la memoria adicional disponible en las áreas compartidas de SQL, podría ser necesario incrementar el número de cursores permitidos por sesión. Esto se hace incrementando el valor del parámetro de inicialización OPEN_CURSORS.

Se debe de tener cuidado de no producir pagineo o swapeo, ya que valores muy grandes en estos parámetros podría inducir a esto.

Otra opción que se puede tener para reducir las perdidas en el caché de librería es el hecho de escribir sentencias de SQL idénticas. Se puede reducir el número de perdidas de "cache" en las llamadas de parseo, asegurándose que las sentencias SQL y bloques de PL/SQL compartidos, sean realmente compartidas en el área compartida de SQL tanto como sea posible. Para dos diferentes ocurrencias de las sentencias SQL o bloque de PL/SQL, en un área compartida de SQL, deben de ser idénticas, de acuerdo a los siguientes criterios:

- El texto de la sentencia SQL o bloque PL/SQL debe de ser idéntico, carácter por carácter, incluyendo espacios y diferencias entre mayúsculas y minúsculas.

Por ejemplo en la siguiente sentencia no puede usar la misma área compartida de SQL:

```
SELECT * FROM emp;

SELECT *  FROM emp;
```

Estas sentencias tampoco pueden usar la misma área de SQL:

```
SELECT * FROM emp;
SELECT * FROM Emp;
```

Referenciar los objetos del esquema, en las sentencias SQL o bloques PL/SQL, deben ser resueltas con los mismos objetos en el mismo esquema.

Por ejemplo, si los esquemas de los usuarios BOB y EDGAR, ambos conteniendo la tabla EMP, y ambos usuarios realizando la siguiente sentencia, sus sentencias no pueden usar la misma área compartida SQL:

```
SELECT * FROM emp;
SELECT * FROM emp;
```

Si ambas sentencias usan la misma tabla y califican la misma tabla con el esquema, como en la siguiente sentencia, entonces ellos pueden usar la misma área SQL compartida:

```
SELECT * FROM bob.emp;
```

Las variables Bind en las sentencias SQL, deben ser de igual nombre y tipo de dato. Por ejemplo, estas sentencias no pueden usar la misma área compartida de SQL:

```
SELECT * FROM emp WHERE deptno = :department_no;
SELECT * FROM emp WHERE deptno = :d_no;
```

Las sentencias SQL deben ser optimizadas usando el mismo optimizador.

Las áreas compartidas de SQL, deben ser usadas para reducir las pérdidas en el "library cache", para usuarios que están ejecutando la misma aplicación. Para esto es necesario estar de acuerdo con los desarrolladores, para mantener estándares, y utilizar las mismas sentencias SQL y bloques de PL/SQL de una aplicación que use las mismas áreas de memoria compartida SQL:

Usar variables bind, en vez de especificar constantes en sus sentencias, cuando esto sea posible.

Por ejemplo, la siguientes sentencias no pueden usar la misma área compartida, porque no son iguales:

```
SELECT ename, empno FROM emp WHERE deptno = 10;
SELECT ename, empno FROM emp WHERE deptno = 20;
```

Para que esta sentencia pueda utilizar la misma área compartida, es necesario sustituir las constantes con variables bind, de esta manera:

```
SELECT ename, empno FROM emp
WHERE deptno = :department_no;
```

De esta manera, las dos ocurrencias de la sentencia, pueden usar la misma área compartida SQL.

Asegurarse que usuarios individuales de aplicaciones, no cambien el tipo de optimización, y las metas para sus sesiones individuales.

Se puede incrementar la probabilidad de que las sentencias SQL para diferentes aplicaciones, puedan ser compartidas, estableciendo políticas para los desarrolladores de estas aplicaciones:

Estandarizar la convención de nombre para variables "bind", y convención de espaciados para los bloque de PL/SQL y sentencias SQL.

Usar procedimientos almacenados en la base de datos cuando sea posible. Si múltiples usuarios utilizan los mismos procedimientos almacenados, automáticamente estarán usando las mismas áreas de PL/SQL. Debido a que los procedimientos son almacenados de forma parseada, en tiempo de corrida, ya no es necesario parsearlos otra vez.

Aumentando velocidad de acceso a las áreas compartidas de SQL en llamadas de ejecución

Si no se tienen perdidas en las librerías de "cache", aun se puede ser capaz de mejorar la velocidad en las llamadas de ejecución, configurando el valor del parámetro de inicialización CURSOR_SPACE_FOR_TIME. Este parámetro especifica cuando un área compartida de SQL puede ser quitada del "cache" de librería, para hacer espacio a una nueva sentencia SQL. El valor por defecto de este parámetro es FALSO, y significa que el área compartida de SQL puede ser quitada del "cache" de librería a pesar de que el cursor de la aplicación asociado con su sentencia SQL esta aún abierto. El valor de

VERDADERO significa que las áreas de SQL pueden ser quitadas cuando todos los cursores de una aplicación con su sentencia están cerrados.

Dependiendo del valor del parámetro CURSOR_SPACE_FOR_TIME, Oracle se comporta de diferente manera cuando una aplicación hace una llamada de ejecución. Si el valor es FALSO, Oracle debe de tomar tiempo para revisar que una área compartida de SQL contiene una sentencia que esta en el "cache" de librería. Si el valor es VERDADERO, Oracle no necesita hacer este chequeo porque el área compartida de SQL nunca será eliminada mientras un cursor de aplicación asociado este abierto. La configuración de este parámetro a VERDADERO hace que Oracle ahorre una pequeña cantidad de tiempo y pueda mejorar el rendimiento en las llamadas de ejecución. Este valor, también previene que sean quitadas la áreas privadas de SQL hasta que el cursor de aplicación asociado sea cerrado.

No se debe configurar el parámetro CURSOR_SPACE_FOR_TIME a VERDADERO, si existen perdidas en el "cache" de librería en las llamadas de ejecución. Tales perdidas indican que el "shared pool" no es suficientemente grande para cargar las áreas compartidas de SQL de todos los cursores abiertos concurrentemente. Si el valor es VERDADERO y no hay espacio en el "shared pool" para nuevas sentencias SQL, las sentencias no pueden ser parseadas, y Oracle retornará un error, indicando que no hay suficiente espacio para memoria compartida. Si el valor es FALSO, y no hay espacio para nuevas sentencias, Oracle quitará áreas existentes que están siendo compartidas. Sin embargo quitar áreas compartidas de SQL del "cache" de librería, implicará de perdidas mas tarde, aunque esto es preferible a obtener

un error que pare por completo s aplicación, ya que no puede parsear sentencias SQL.

No se debe configurar el parámetro CURSOS_SPACE_FOR_TIME a VERDADERO, si la cantidad de memoria disponible para cada usuario en las áreas privadas de SQL es escaso. Este valor también previene de quitar de áreas privadas de SQL a los cursores abiertos asociados con la aplicación. Si las áreas de SQL privadas para todos los cursores abiertos concurrentemente están llenos, de manera que no hay espacio para reservar áreas privadas de SQL para nuevas sentencias SQL, la sentencia no será parseada y Oracle retornará un error, indicando que no hay suficiente memoria.

"cache" para cursores de sesión

Si una aplicación realiza varias llamadas a parseo en el mismo conjunto de sentencias SQL, el hecho de re-abrir los cursores de la sesión, puede afectar el rendimiento del sistema. Los cursores de la sesión, pueden ser almacenados en un "cache" de cursores de sesión. Esta característica puede ser usada, particularmente en aplicaciones diseñadas usando Oracle Forms, y esto es debido a que pueden cambiarse entre formas cerradas, todas los cursores de sesión asociadas con una forma.

Oracle usa el área compartida de SQL para determinar si mas de tres requerimientos de parseo han sido obtenidos de una sentencia dada. Si es así, Oracle asume que los cursores de la sesión asociados con la sentencia podrían estar en "cache" y mover entonces dichos cursores hacia la sección de "cache" para cursores en la sesión. Subsecuentemente los requerimientos

para parsear estas sentencias SQL, para la misma sesión, serán hallados en el respectivo "cache" de cursores en la sesión.

Para habilitar este "cache" de cursores, se debe de configurar el parámetro de inicialización llamado SESSION_CACHED_CURSORS. Este parámetro es un entero positivo que especifica el número máximo de cursores en la sesión que pueden ser mantenidos en el "cache". Un algoritmos del 'Menos recientemente utilizado' (LRU - Least Recently Used) es usado como política para decidir eliminar cursores, cuando hagan falta nuevas entradas de cursores.

También se pueden habilitar "cache" para cursores de sesiones de manera dinámica, por medio de la instrucción ALTER SESSION SET SESSION_CACHED_CURSORS.

Para determinar si el "cache" de cursores de una sesión es suficientemente larga para la instancia, se pueden examinar las estadísticas de sesión "session cursor cache hits" en la vista V$SESSTAT. Estas estadísticas cuentan el número de veces que una llamada de parseo es realizada para un cursor en el "cache" de cursor para sesión. Si estas estadísticas tienen un porcentaje pequeño del total de veces de llamadas de parseo para una sesión, entonces se debe de considerar configurar el parámetro SESSION_CACHED_CURSORS a un valor mas grande.

6.1.1.2 Afinación del "cache" de diccionario de datos

6.1.1.2.1 Examinando la actividad del "cache" del diccionario de datos

Se debe de determinar si las perdidas en el "cache" del diccionario de datos, afectan el rendimiento de Oracle. Se puede examinar dicha actividad seleccionando información de la tabla V$ROWCACHE como se describe a continuación.

Normalmente se esperan perdidas en el "cache" del diccionario solamente en algunos casos. Por ejemplo, cuando la instancia de la base de datos se levanta, el "cache" del diccionario de datos no contiene datos, así que cualquier sentencia SQL, tendrá como resultado, perdidas en el "cache" del diccionario de datos. Mientras mas datos sean leídos a "cache", la probabilidad de perdidas en el "cache" del diccionario, decrementará. Eventualmente, la base de datos alcanzará un "estado estable" en el que la mayoría de datos del diccionario estarán en "cache". En este punto, muy pocas perdidas ocurrirán. Para afinar este "cache", se debe de examinar su actividad solamente hasta después que su aplicación halla sido ejecutada.

Las estadísticas de la vista V$ROWCACHE, reflejan la actividad del diccionario de datos, que son mantenidos en la tabla dinámica V$ROWCACHE. Por defecto, esta tabla solamente esta disponible para el usuario SYS, y para aquellos usuarios que tengan el permiso SELECT ANY TABLE, como el usuario SYSTEM.

Cada tupla en esta tabla contiene estadísticas para un tipo sencillo de elemento del diccionario de datos. Estas estadísticas, reflejan toda la actividad

del diccionario de datos desde la última vez que la instancia se levanto. Las columnas de V$ROWCACHE, reflejan el uso y la efectividad del "cache" del diccionario de datos:

PARAMETER

Esta columna identifica un elemento en particular del diccionario de datos. Para cada tupla, el valor de esta columna contiene el prefijo 'dc_'. Por ejemplo, en la tupla que contiene las estadísticas para descripción de archivos, dicha columna tiene el valor 'dc_files'.

GETS

Esta columna muestra el número total de requerimientos de información para un elemento correspondiente (en particular).
Por ejemplo, en las tuplas que contienen estadísticas para descripción de archivos, esta columna contiene el número total de requerimientos para archivos de descripción de datos.

GETMISES

Esta columna muestra el número de requerimientos de datos que resultaron de perdidas en el "cache".

Una operación de selección, que puede ayudar a examinar la aplicación que se esta ejecutando por medio de las estadísticas de la tabla V$ROWCACHE durante un período, podría ser la siguiente:

```
SELECT SUM(gets)  "Data Dictionary Gets",
SUM(getmisses) "Data Dictionary Cache Get Misses"
FROM v$rowcache;
```

Figura 45 Instrucción de selección para monitorear el diccionario de datos

La salida de esta selección podría ser como la siguiente:

Data Dictionary Gets	Data Dictionary Cache Get Misses
1439044	3120

Figura 46 Ejemplo de salida de la instrucción anterior

Interpretando los valores de V$ROWCACHE, se puede observar lo siguiente:

La suma de la columna de GETS indica que hubo un total de 1,439,044 requerimientos del diccionario de datos.

La suma de la columna GETMISSES indica que 3120 requerimientos para el diccionario de datos, resultaron en perdidas.

La razón (o el radio) de la suma de GETMISSES sobre GETS fue cerca del 0.2%.

6.1.1.2.2 Reduciendo las pérdidas en el "cache" del diccionario de datos

Para accesos frecuentes del "cache" del diccionario, la razón (o radio) del total de GETMISSES al total de GETS debería de ser menos del 10% al 15%. Si esta razón continua incrementándose arriba de este límite, mientras la

aplicación esta ejecutándose, se debe de considerar incrementar la cantidad de memoria disponible para el "cache" del diccionario de datos. Esto se hace incrementando el parámetro de inicialización SHARED_POOL_SIZE. El valor máximo de este parámetro depende del sistema operativo.

6.1.1.2.3 Afinando el "shared pool" con la opción de servidores multi - enlazados (Multi-Threaded Server)

En la arquitectura de Servidores Multi - Enlazados, Oracle almacena información de la sesión en el "shared pool", en lugar de usar la memoria del proceso usuario. La información de la sesión, incluye áreas de SQL privadas y áreas para ordenamientos. Si se está usando esta arquitectura, se puede necesitar aumentar el tamaño del "shared pool" para acomodar esta información.

6.1.1.2.4 La tabla V$SESSTAT

Oracle recolecta estadísticas de la memoria total usada por una sesión, y la almacena en esta tabla dinámica (V$SESSTAT). Por defecto, esta tabla solo puede ser accesada por el usuario SYS, y por todo aquel usuario con permisos de SELECT ANY TABLE, tal como SYSTEM. Estas estadísticas son usadas para medir el uso de la memoria por las sesiones:

Memoria de sesión (session memory):

Los valores de esta estadística constituye la cantidad de memoria (en bytes) reservada para la sesión.

Máxima memoria de sesión (Max Session Memory):

El valor de esta estadística es la máxima cantidad de memoria (en bytes) alguna vez reservada para la sesión.

6.1.1.2.5 Seleccionando la tabla V$SESSTAT

Se puede utilizar esta selección para decidir que tan grande se puede hacer el "shared pool", si se utiliza la arquitectura de servidores multi-enlazados. La siguiente selección se puede hacer mientras la aplicación esta corriendo:

```
SELECT SUM(value) || ' bytes' "Total de memoria para todas las sesiones"
FROM v$sesstat, v$statname
WHERE name = 'session memory'
AND v$sesstat.statistic# = v$statname.statistic#

SELECT SUM(value) || ' bytes' "Maxima memoria total para todas las sesiones"
FROM v$sesstat, v$statname
WHERE name = 'max session memory'
AND v$sesstat.statistic# = v$statname.statistic#
```

Figura 47 Instrucciones de selección para monitoreo de memoria en Oracle.

Estas selecciones también se pueden hacer de la tabla dinámica V$STATNAME, para obtener identificadores internos para la memoria de sesión, y para la máxima memoria de sesión. Los resultados de estas selecciones pueden ser como los siguientes:

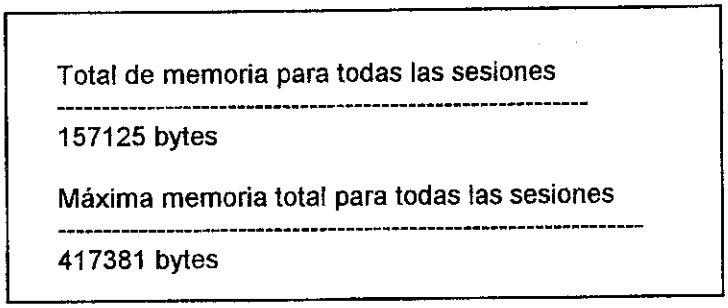


Figura 48 Salida de las instrucciones de selección anteriores.

Interpretando la tabla V$SESSTAT

El resultado de la primera selección, indica que la memoria actual reserva para todas las sesiones 157,125 bytes. Este valor es el total de memoria, cuya localización depende de cuántas sesiones estén conectadas a Oracle. Si la sesión esta conectada a procesos servidores dedicados, esta memoria es entonces, parte de la memoria de los procesos usuario. Si las sesiones están conectadas por medio de procesos servidores compartidos, esta memoria es parte del "shared pool". El resultado de la segunda selección, indica que la suma del tamaño máximo de memoria para todas las sesiones es de 417,381 bytes. El segundo resultado es más grande que el primero porque algunas sesiones tienen que quitar memoria desde que reservan su máxima cantidad.

Se puede usar el resultado de estas selecciones para determinar que tan grande se debe hacer el "shared pool" si se tiene la arquitectura de servidores multi-enlazados. El primer valor será una mejor estimación que el segundo, a menos que todas las sesiones ganen su máxima memoria al mismo tiempo.

6.1.2 Afinamiento del "cache" de Buffers

6.1.2.1 Examinando la actividad del "cache" de buffers

Oracle colecta estadísticas que reflejan el acceso y almacenamiento de datos, en la tabla dinámica llamada V$SYSSTAT. Como las anteriores tablas dinámicas, esta también esta disponible únicamente para el usuario SYS, y para cualquier otro que tenga el privilegio SELECT ANY TABLE, como SYSTEM. Estas estadísticas normalmente son usadas para afinar el "cache" de buffers:

db block gets, consistent gets:
La suma de estos valores estadísticos, es el número total de requerimientos para los datos. Este valor incluye los requerimientos de datos que han sido satisfechos por acceso a buffers en memoria.

Physical reads:
Estas estadísticas constituyen el número total de requerimientos para los datos resultantes de accesos a los archivos de datos que se encuentran en disco (esto es accesos directos a disco).

Se recomienda examinar estas estadísticas durante ciertos períodos de tiempo, mientras las aplicaciones están corriendo. Esto se puede hacer con la siguiente instrucción de selección:

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('db block gets', 'consistent gets', 'physical reads');
```

Figura 49 Instrucción de selección para obtener los valores necesarios para el calculo de "Hit Ratio"

La salida de esta selección puede ser como la siguiente:

NAME	VALUE
db block gets	85792
consistent gets	278888
physical reads	23182

Figura 50 Datos necesarios para obtener el "Hit Ratio" de la selección anterior.

A continuación, se procede a calcular el llamado 'hit ratio' (vista en el capítulo número 2 de esta obra) para obtener una idea de como se esta usando la memoria RAM para el "cache" de buffers:

Hit Ratio = 1 - (physical reads / (db block gets + consistent gets))

Utilizando estas estadísticas, se determina que el Hit Ratio, para el "cache" de buffers es del 94%.

6.1.2.2 Rediciendo las perdidas en el "cache" de buffers

Si su hit ratio es bajo, o sea menor del 60% o 70%, entonces se debe de incrementar el número de buffers en dicho "cache", para mejorar el rendimiento. Esto se logra en una instancia de Oracle, por medio del parámetro de inicialización llamado DB_BLOCK_BUFFERS.

Una ventaja que tiene Oracle, es que se pueden coleccionar estadísticas, para poder estimar el rendimiento, si se aumenta el tamaño del

"cache" de buffers. Con estas estadísticas, se pueden estimar cuantos buffers se deben de añadir a su "cache", aún sin haberlos colocado realmente.

6.1.2.2.1 La tabla X$KCBRBH

La tabla virtual SYS.X$KCBRBH contiene estadísticas que estiman el rendimiento de un "cache" grande. Cada tupla en la tabla refleja el relativo valor de rendimiento a la hora de añadir un buffer al "cache". Esta tabla solamente puede ser accesada por el usuario SYS. Las siguientes son las columnas que componen la tabla X$KCBRBH:

INDX:
El valor de esta columna es uno menos que el número de buffers que podría potencialmente ser añadido al "cache".
COUNT:
Esta columna contiene el número de "cache hits" adicionales que podría obtener añadiendo el número INDX+1 al "cache".

Por ejemplo, en la primera tupla de la tabla, el valor de INDX es de 0 y el valor de COUNT es el número del "cache hits", ganados al haber añadido el primer grupo de buffers al "cache". En la segunda tupla, el valor de INDX es 1 y el valor de COUNT es el número de "cache hits" para el segundo conjunto de buffers adicionales.

6.1.2.2.2 Habilitando la tabla X$KCBRBH

El hecho de colectar estadísticas en la tabla X$KCBRBH, es controlada por el parámetro de inicialización llamado DB_BLOCK_LRU_EXTENDED_STATISTICS. El valor de este parámetro determina el número de tuplas en la tabla X$KCBRBH. El valor por defecto es de 0, el cual significa que el comportamiento por defecto es de no colectar estadísticas.

Por ejemplo, si se coloca el valor del parámetro a 100, entonces Oracle, colectará 100 tuplas de estadísticas, cada tupla reflejará la adición de un buffer, arriba de los 100 buffers extras.

El colectar estas estadísticas, incurre en alguna sobrecarga en el rendimiento. Esta sobrecarga es proporcional al número de tuplas en la tabla. Para evitar la sobrecarga, se recomienda colectar estadísticas solo cuando se este afinando el "cache" de buffers, y deshabilitar dichas estadísticas cuando se halla terminado el afinamiento.

6.1.2.2.3 Selección de la tabla X$KCBRBH

De la información de la tabla X$KCBRBH, se puede predecir la ganancia potencial, a la hora de incrementar el tamaño del "cache". Por ejemplo, para determinar cuanto mas de cache hit podría ocurrir si se añaden 20 buffers de "cache", seleccionar la tabla X$KCBRBH con la siguiente sentencia SQL:

```
SELECT SUM(count) ach
FROM sys.x$kcbrbh
WHERE indx < 20;
```

Figura 51 Selección para observar la ganancia potencial a la hora de aumentar bloques.

También se puede determinar como los nuevos buffers cache, afectan el "hit ratio". Simplemente lo que se debe de hacer es usar la fórmula de hit ratio, basado en los valores de la estadística del "db block gets", "consistent gets", y "physical reads", y el número de "hits cache" adicionales (ACH) retornados por la selección:

Hit Ratio = 1 - (physical reads - ACH / (db block gets + consistent gets))

6.1.2.2.4 Agrupando tuplas en la tabla X$KCBRBH

Otra manera de examinar la tabla X$KCBRBH es agrupando los buffers adicionales en intervalos largos. Para esto se puede ver la siguiente selección:

```
SELECT 250*TRUNC(indx/250)+1||' a '||250*(TRUNC(indx/250)+1)
Intervalo", SUM(count) "Buffer Cache Hits"
FROM sys.x$kcbrbh
GROUP BY TRUNC(indx/250);
```

Figura 52 Instrucción de selección usada para observar la ganancia en "cache hits" a la hora de incrementar los bloque de memoria

El resultado de esta selección podría ser como la siguiente:

Intervalo	Buffer Cache Hits
1 a 250	16080
251 a 500	10950
501 a 750	710
751 a 1000	23140

Figura 53 Ejemplo de la ganancia obtenida por intervalos de 250 bloques.

Examinando estos resultados, se puede deducir lo siguiente:

Si 250 buffers son añadidos al "cache", entonces 16,080 "cache hits" podrían ser ganados.
Si 250 buffers mas son añadidos para un total de 500 buffers adicionales, 10,950 "cache hits" podían ganarse, en adición de 16,080 "cache hits" desde los primeros 250 buffers. Esto significa que añadiendo 500 buffers podría producir un total de 27,030 "cache hits" adicionales.

Si 250 buffers mas fueran añadidos, para un total de 750 buffers adicionales, 710 "cache hits" podrían ganarse, produciendo un total de 27,740 "cache hits" adicionales.

Si 250 buffers fueran añadidos, para un total de 1000 buffers adicionales, entonces 23,140 "cache hits" serían ganados, produciendo un total de 50,880 "cache hits" adicionales.

Basados en estas observaciones, se podría decidir cuantos buffers se pueden añadir al "cache". En este caso, la persona que hace el afinamiento tomará la decisión

6.1.2.3 Removiendo buffers innecesarios

Si el hit ratio es algo, su "cache" es probablemente los suficientemente grande, para cargar la data mas frecuentemente accesada. En este caso, también se puede reducir el tamaño del "cache", y aún mantener un buen rendimiento. El objetivo de reducir el "cache", podría ser, dejar mayor cantidad de memoria RAM, para procesos de usuario, o del sistema operativo.
Para reducir el "cache", entonces simplemente se reduce el tamaño al parámetro de inicialización DB_BLOCK_BUFFERS. El valor mínimo para este parámetro es de 4.

Oracle puede colectar estadísticas para predecir el rendimiento del buffer "cache", basándose en un tamaño de "cache" pequeño. Examinando estas estadísticas, puede ayudar, el determinar que tan pequeño se desea colocar el buffer "cache", sin afectar el rendimiento.

6.1.2.3.1 La tabla X$KCBCBH

Esta tabla virtual, contiene estadísticas que estiman el rendimiento de un "cache" pequeño. Su estructura es similar a la X$KCBRBH. La siguientes son las columnas de dicha tabla:

INDX:
El valor de esta columna es el número potencial de buffers en el "cache".

COUNT:
El valor de esta columna es el número de "cache hits" atribuibles al número de buffers INDX.

El número de tuplas en esta tabla es igual al número de buffers en el "buffer cache". Cada tupla es la tabla refleja el número de "cache", atribuidos a un solo buffer. Por ejemplo, en la segundo tupla, el INDX el valor es de 1 y el valor de COUNT es el número de "cache hits" para el segundo buffer. En la tercera tupla, INDX es 2 y el valor de COUNT es el número de "cache hits" del tercer buffer.

Las primeras tuplas de la tabla contienen información especial. El valor INDX es 0 y el valor de COUNT es el número total de bloques movidos hacia el primer buffer en el "cache".

6.1.2.3.2 Habilitando la tabla X$KCBCBH

La recolección de estadísticas en la tabla X$KCBCBH es controlada por el parámetro de inicialización DB_BLOCK_LRU_STATISTICS. El valor de este parámetro, determina si Oracle colecta las estadísticas. El valor por defecto de este parámetro es FALSO, lo cual significa que por defecto el comportamiento es el de no colectar estadísticas. Para habilitar las estadísticas en esta tabla, colocar el parámetro DB_BLOCK_LRU_STATISTICS a VERDADERO. Al igual

que en el tema anterior, esta recolección de estadísticas, incurren en sobrecarga en el rendimiento. Se debe de proceder de la misma manera.

6.1.2.3.3 Seleccionando información de la tabla X$KCBCBH

De la información de la tabla X$KCBCBH, se puede predecir el número de perdidas adicionales de "cache" que se podría incurrir, si el número de buffers "cache" fuese reducido. Si el actual "buffer cache" contiene 100 buffers, se conocería un estimado de cuántas pérdidas podrían ocurrir si solo se dejaran 90. Para determinar el número de perdidas adicionales en "cache", se puede realizar la siguiente sentencia SQL:

```
SELECT SUM(count) acm
FROM sys.x$kcbcbh
WHERE indx >= 90;
```

Figura 54 Instrucción para calcular la perdida en "cache Hits" a la hora de eliminar bloques.

También se puede determinar el hit ratio, basado en el tamaño del "cache". Simplemente se debe de usar la fórmula del hit ratio, basada en los valores del db block gets, consistent gets y physical reads, y el número de perdidas adicionales del "cache" (ACM additional cache misses) retornado por la sentencia:

Hit Ratio = 1 - (physical reads + ACM / (db block gets + consistent gets))

Otra forma de examinar la tabla X$KCBCBH es agrupar los buffers en intervalos. Por ejemplo, si su "cache" contiene 100 buffers, se podría desear dividir el "cache" en cuatro intervalos de 25 buffers.
Para realizar esto, se utiliza la siguiente sentencia SQL:

```
SELECT 25*TRUNC(indx/25)+1||' a '||25*(TRUNC(indx/25)+1)
"Intervalo", SUM(count) "Buffer Cache Hits"
FROM sys.x$kcbcbh
WHERE indx > 0
GROUP BY TRUNC(indx/25);
```

Figura 55 Instrucción de selección usada para observar la pérdida en cache Hits a la hora de decrementar los bloque de memoria

La cláusula WHERE de la selección, elimina la posibilidad de colectar estadísticas de la primera tupla de la tabla. El resultado es el siguiente:

Intervalo	Buffer Cache Hits
1 a 25	1900
26 a 50	1100
51 a 75	1360
76 a 100	230

Figura 56 Ejemplo de la pérdida obtenida por intervalos de 250 bloques.

examinando la salida, podemos deducir lo siguiente:

Los últimos 25 buffers en el "cache" (buffer 76 al 100) contribuyen a 230 "cache hits". Si el "cache" se reduce a 75 buffers, 230 "cache hits" se perderían.

El tercer intervalo de 25 buffers (buffer 51 al 75) contribuyen en 1,360 "cache hits". Si estos buffers son removidos del "cache", 1,360 "cache hits" se perderían, en adición a los 230 "cache hits" perdidos en los buffers del 76 al 100. Removiendo 50 buffers, resultaría en una perdida de 1,590 "cache hits".

El segundo intervalo de buffers (del 26 al 50) contribuyen en 1,100 "cache hits". Al remover 75 buffers del "cache", podría resultar en una perdida total de 2,690 "cache hits".
Los primeros 25 buffers en el "cache" (buffers del 1 al 25) contribuyen en 1,900 "cache hits".

De acuerdo a estos resultados podemos ver que el único intervalo que sería insignificante eliminar son el buffer 76 al 100, ya que relativamente el número de "cache hits" es pequeño e insignificante con relación a los otros. En pocas palabras se podrían eliminar 25 buffers.

6.2 Afinamiento de Disco E/S

6.2.1 Importancia del afinamiento de E/S

El rendimiento de diferentes aplicaciones de software es normalmente limitado por los accesos a disco (Entradas/Salidas). A menudo, la actividad de CPU debe ser suspendida, mientras la actividad de E/S es completada. Estas aplicaciones se dicen que están "Amarradas a disco E/S". Oracle esta diseñado, para que el rendimiento necesario, no este limitado por las E/S.

6.2.2 Reducción de contención de disco

6.2.2.1 Monitoreo en la actividad de disco

Esto se puede hacer de las siguientes maneras:

6.2.2.1.1 Por estadísticas de E/S en los archivos de Oracle

- Por estadísticas del sistema operativo.

Monitoreando la actividad de disco en Oracle

Para examinar los accesos a disco para archivos de la base de datos, se puede hacer por tablas de rendimiento dinámicas, como la V$FILESTAT. Los valores de las columnas, reflejan el número de accesos a disco para cada archivo de datos (datafile):

PHYRDS
Los valores de esta columna, son el número de lecturas de cada archivo de la base de datos.

PHYWRTS
Los valores de esta columna, es el número de escrituras a cada archivo de la base de datos.

El monitoreo de estos valores se pueden hacer, por períodos de tiempo, durante la ejecución de la aplicación, con la siguiente sentencia:

```
SELECT name, phyrds, phywrts
FROM v$datafile df, v$filestat fs
WHERE df.file# = fs.file#;
```

Figura 57 Instrucción usada para monitoreo de accesos a disco en "Data Files"

Esta sentencia también retorna el nombre de cada archivo da la base de datos, de la tabla V$DATAFILE. La salida de la sentencia es como la siguiente:

NAME	PHYRDS	PHYWRTS
/oracle/ora70/dbs/ora_system.dbf	7679	2735
/oracle/ora70/dbs/ora_temp.dbf	32	546

Figura 58 Ejemplo de salida de lecturas y escrituras físicas de archivos de datos.

El total de E/S para un disco, es la suma de las columnas PHYRDS y PHYWRTS, para todos los archivos de la base de datos, manejados por la instancia de Oracle en el disco. Se recomienda, determinar este valor, para cada uno de los discos del sistema. También se debe de determinar la razón en la que ocurre las E/S, para cada disco, dividiendo el total de E/S, por el intervalo de tiempo fuera (time over), cuyas estadísticas fueron colectadas.

6.2.2.1.2 Monitoreando la actividad del disco por medio del sistema operativo

En los discos se encuentran almacenados los archivos de datos y los archivos de redo logs, además de esto, también existen almacenados otros archivos que no están relacionados con Oracle. El acceso a estos archivos solamente pueden ser monitoreados a través de utilitarios del sistema operativo, en lugar de la tabla V$FILESTAT. Dichos utilitarios son dependientes del sistema operativo, y se recomienda la revisión de estos utilitarios, en la documentación del respectivo sistema operativo; Por ejemplo, un utilitario bastante completo es el 'VMSTAT' de Unix.

Normalmente se debe de utilizar estas facilidades del sistema operativo, para examinar el total de accesos de entrada/salida a los discos. Se debe de tratar de reducir grandes cantidades de accesos a los discos que contienen los archivos de datos de la base de datos.

6.2.2.2 Distribuir los accesos de Entrada/Salida

Se deben de considerar las estadísticas de la tabla de V$FILESTAT y las del utilitario del sistema operativo. De acuerdo a esto, se recomienda consultar la documentación de hardware de los discos duros, para determinar los límites de su capacidad. Normalmente, cualquier disco, operando cerca de su capacidad máxima, es potencialmente un candidato para que exista contención. Por ejemplo, entre cuarenta o mas accesos de entrada/salida por segundo, son excesivos para la mayoría de discos en los sistemas operativos UNIX o VMS.

Para reducir la actividad de sobrecarga al disco, se recomienda mover uno o mas de los archivos que están siendo mayormente accesados a los discos que no tienen mucha carga. Se debe de aplicar este principio para cada disco, hasta que todos los discos tengan la misma cantidad de accesos de E/S. Este es el concepto de distribuir E/S.

A continuación se presentan algunas guías para distribuir las E/S:

- Separar archivos de datos y archivos de bitácora en diferentes discos.
- Separar datos de tablas en diferentes discos ('Stripe').
- Separar tablas e índices en diferentes discos.
- Reducir los accesos de E/S que no están relacionados con Oracle.

6.2.2.2.1 Separando archivos de datos y "redo log files"

Oracle procesa constantemente los accesos a archivos de datos y archivos de bitácora. Si estos archivos están en discos comunes, hay una contención potencial.

Se recomienda que colocar cada archivo de datos en discos separados; De esta manera, múltiples procesos pueden accesar diferentes archivos de manera concurrente, sin contención de disco.

También se recomienda colocar los archivos de bitácora en discos separados, que no tengan otra actividad. Los Archivos de bitácora son escritos por el proceso de escritura de Logs (LGWR) cuando una transacción es llevada a cabo (cometida). La información en un archivos de bitácora es escrita secuencialmente. Esta escritura secuencial puede tomar lugar mucho mas rápido si no hay actividad concurrente en el mismo disco.

Dedicar un disco separado a archivos de redo logs, generalmente asegura que el proceso 'LGWR' corre bien, sin ninguna otra atención de afinamiento. Normalmente los cuellos de botella relacionados con el LGWR son raros.

Nota: Copias espejo de los archivos de bitácora, o mantenimiento de múltiples copias de cada uno de los archivos de bitácora, se recomienda colocarlos en discos separados, y aunque bajara un poco el rendimiento del proceso LGWR, éste sin embargo no será considerable, debido a que las escrituras del LGWR se hacen en paralelo, y espera hasta que cada parte de la escritura paralela sea completada.

El hecho de dedicar discos separados y colocar los archivos de redo log en espejo, es una precaución importante. A la hora de separar los archivos de datos y los archivos de redo logs en diferentes discos, se asegura que ambos, los archivos de datos y los archivos de redo logs no pueden ser perdidos en la falla de un solo disco. Espejear los archivos de redo logs, asegura que un archivo de redo log no puede ser perdido en una falla de un solo disco.

6.2.2.2.2 Dividiendo datos de tablas ('Striping')

"Striping" es la practica de dividir datos de tablas grandes en pequeñas porciones, y almacenarlas en distintos archivos de datos colocados en discos separados. Esto permite que múltiples procesos accesen diferentes porciones de la tabla concurrentemente, sin que exista contención. El "Striping" ayuda en la optimización de accesos aleatorios a tablas, generalmente cuando estas contienen muchas tuplas. Puede ser realizado manualmente, o a través de utilitarios propios del sistema operativo. (Normalmente esta practica es bastante útil cuando se usa la opción de querys paralelos).

El siguiente es el procedimiento para subdividir una tabla ('striped table'):

1. Crear un tablespace con la sentencia CREATE TABLESPACE. Especificar los archivos de datos en la cláusula DATAFILE. Cada uno de los archivos deben de están en diferente disco.

```
CREATE TABLESPACE stripedtabspace
DATAFILE 'file_on_disk_1' SIZE 500K,
'file_on_disk_2' SIZE 500K,
'file_on_disk_3' SIZE 500K,
'file_on_disk_4' SIZE 500K,
'file_on_disk_5' SIZE 500K;
```

Figura 59 Instrucción para creación de tablespace con distintos archivos de datos.

2. Crear ahora la tabla con la sentencia CREATE TABLE. Especificar el tablespace en que se desea crear con la cláusula TABLESPACE (en este caso seria el llamado stripedtabspace).

También especificar el tamaño de los extents de la tabla en la cláusula STORAGE. Cada uno de los extents en un archivo de dato separado (datafile). Los extents de la tabla pueden ser casi tan pequeños como los archivos de datos en el tablespace, por ejemplo:

```
CREATE TABLE stripedtab (
col_1  NUMBER(2),
col_2  VARCHAR2(10) )
TABLESPACE stripedtabspace
STORAGE ( INITIAL 495K  NEXT 495K
MINEXTENTS 5  PCTINCREASE 0 );
```

Figura 60 Instrucción para creación de tabla.

Este paso resulta en la creación de la tabla llamada STRIPEDTAB. Ésta tiene cinco extents inicialmente, cada uno de 495 kilobytes. Cada extent esta colocado en un archivo de datos (datafiles), los cuales son nombrados en la cláusula DATAFILE de la sentencia CREATE TABLESPACE. Estos archivos están todos separados en diferentes discos. Los cinco extents, son reservados inmediatamente, por el hecho de haber colocado el MINEXTENTS en cinco.

6.2.2.2.3 Separando tablas e índices

Es recomendable colocar las estructuras de la base de datos que son accesadas frecuentemente en archivos de datos (datafiles) sobre discos separados. Para hacer esto, se deben de conocer las estructuras de la base de datos que son utilizadas frecuentemente. Por ejemplo, separar una tabla que es bastante accesada de su índice. Esta separación distribuye las E/S de la tabla e índice a través de discos separados.

Para separar tablas de índices se deben de seguir los siguientes pasos:

1. Crear un tablespace con la sentencia CREATE TABLESPACE. Especificar el datafile en la respectiva cláusula:

```
CREATE TABLESPACE tabspace_1
DATAFILE 'file_on_disk_1';
```

Figura 61 Creación de "tablespace" para datos.

1. Crear una tabla con el comando CREATE TABLE. Especificar el tablespace en la cláusula TABLESPACE:

```
CREATE TABLE tab_1 (
  col_1  NUMBER(2),
  col_2  VARCHAR2(10) )
TABLESPACE tabspace_1;
```

Figura 62 Creación de tabla sobre el "tablespace" de datos.

2. Crear otro tablespace. Especificar un archivos de datos en otro disco:

```
CREATE TABLESPACE tabspace_2
DATAFILE 'file_on_disk_2';
```

Figura 63 Creación del "tablespace" de índices.

3. Crear el índice. Especificar el nuevo tablespace:

```
CREATE INDEX ind_1 ON tab_1 (col_1)
TABLESPACE tabspace_2;
```

Figura 64 Creación de índice sobre el "tablespace" de índices.

Con estos pasos, el resultado es la creación de la tabla llamada TAB_1 en el archivo FILE_ON_DISK_1 y la creación de el índice IND_1 en el archivo FILE_ON_DISK_2.

6.2.2.2.4 Eliminando otros accesos de E/S al disco

Si es posible, se recomienda eliminar accesos de E/S que no estén relacionados a Oracle en discos que contengan archivos de la base de datos. Esta medida es muy útil adoptarla para optimizar el acceso a los archivos de bitácora. Tiene la característica de reducir la contención de discos, además permite monitorear toda la actividad de tales discos, a través de la tabla dinámica de rendimiento llamada V$FILESTAT.

6.2.2.2.5 Modificando el archivo SQL.BSQ

El archivo SQL.BSQ es ejecutado cuando se realiza la sentencia CREATE DATABASE. Este archivo contiene la definición de tablas actual que constituye en su mayoría al Servidor de Oracle. Las vistas que normalmente puede utilizar un DBA son extraídas de estas tablas. La corporación Oracle,

recomienda que cualquier usuario estrictamente limite sus modificaciones del archivo SQLBSQ.

- Si es necesario, puede incrementar el valor de los siguientes parámetros: INITIAL, NEXT, MINEXTENTS, MAXEXTENTS, PCTINCREASE, FREELISTS, FREELIST GROUPS, y OPTIMAL.
- Con la excepción de PCTINCREASE, no se debería decrementar la configuración de los parámetros de almacenamiento (storage parameter) hacia valores que sean menores del por defecto. (Si el valor del parámetro MAXEXTENTS es largo, entonces se puede colocar el parámetro PCTINCREASE pequeño o cero).

- Ningún otro cambio al archivo SQL.BSQ es soportado. En particular, no se debe de añadir, borrar o renombrar columnas.

6.3 Otros elementos importantes para la afinación del "Kernel" de Oracle

6.3.1 Reducción de la contención de los segmentos de deshecho (Rollback segments)

6.3.1.1 Identificación de contención en los segmentos de "rollbacks"

La contención de los segmentos de rollbacks, es reflejado por la contención de buffers que contienen a los bloque de segmentos de rollbacks. Se puede determinar si existe contención de rollbacks, utilizando la tabla dinámica llamada V$WAITSTAT. Esta tabla contiene estadísticas que reflejan la contención a nivel de bloques. Por defecto, esta tabla solamente está disponible para el usuario SYS, y para todos aquellos usuarios que tengan el privilegio SELECT ANY TABLE. Sus estadísticas reflejan la contención de diferentes clases de bloques:

system undo header:
El valor de estas estadísticas, contiene el número de esperas para los buffers, contenidas en los bloques de espera del segmento de rollback llamado SYSTEM.
System undo block:
El valor de estas estadísticas, contiene el número de esperas para los buffers, contenidas en las bloques, en el segmento de rollback SYSTEM.

Undo header:

Estas estadísticas, contienen el número de esperas para los buffers, contenidas en las cabeceras de bloque, de los segmentos rollback que no son SYSTEM.

Undo Block:

Estas estadísticas, contienen el número de esperas para los "buffers", contenidas en otros bloques de los segmentos de "rollback" diferentes del "SYSTEM".

Se recomienda monitorear estas estadísticas, en un período de tiempo, mientras las aplicaciones están siendo ejecutadas; La siguiente selección puede ser usada para esto:

```
SELECT class, count
FROM v$waitstat
WHERE class IN ('system undo header', 'system undo block',
'undo header', 'undo block');
```

Figura 65 Instrucción de selección para monitoreo de "rollbacks".

El resultado de esta selección puede ser como la siguiente:

CLASS	COUNT
system undo header	2089
system undo block	633
undo header	1235
undo block	942

Figura 66 Ejemplo del monitoreo de "rollbacks"

Ahora se debe de comparar el número de esperas para cada clase de bloque, con el número total de requerimientos de datos, sobre un período de tiempo. Se puede monitorear el número total de requerimientos de datos en un período de tiempo con la siguiente selección:

```
SELECT SUM(value)
FROM v$sysstat
WHERE name IN ('db block gets', 'consistent gets');
```

Figura 67 Selección para monitorear el número total de requerimientos de datos

El resultado es el siguiente:

```
SUM(VALUE)
----------
    929530
```

Figura 68 Ejemplo de la salida del total de requerimientos de datos

Si algún número de esperas, para cualquier clase de bloque, es mayor al 1% del total de requerimientos, entonces se debe de considerar agregar mas segmentos de rollbacks para reducir la contención.

6.3.1.2 Creando segmentos de "rollback"

Para reducir la contención de buffers contenidos en los bloques de segmentos de rollbacks, ser recomienda crear mas de dichos segmentos. La siguiente tabla muestra algunas guías generales para escoger cuantos

segmentos de "rollback" pueden ser reservados, basados en el número de transacciones concurrentes de su base de datos.

No. de Transacciones concurrentes	No. de Segmentos de rollback recomendados
n<16	4
16<=n<32	8
32<n	n/4 pero no mayor de 50

Tabla 3 Condiciones para decidir el número de "rollback segments"

6.3.2 Afinando ordenamientos

6.3.2.1 Reservando memoria para las áreas de ordenamientos (Sort Areas)

El valor por defecto del tamaño de áreas de ordenamiento, es adecuado para cargar la mayoría de ordenamientos (normalmente en una base de datos Oracle, el valor por defecto es de 65kb por usuario). Sin embargo, si la aplicación lleva a cabo varios ordenamientos largos de datos, el valor no será el óptimo, por lo que se recomienda incrementar dicho tamaño.

6.3.2.1.1 Reconocimiento de ordenamientos largos

La tabla que recolecta las estadísticas necesarias para observar la actividad de ordenamientos, se llama V$SYSSTAT.

Sorts (Memoria):

Estas estadísticas contienen el número de ordenamientos pequeños, suficientes para ser llevados a cabo completamente en las áreas de ordenamiento de RAM, sin accesar segmentos temporales en disco.

Sorts (Disco):

Estas estadísticas contienen el número de ordenamientos largos, llevados a cabo en áreas de ordenamientos, pero accesando segmentos temporales a disco.

Cuando la aplicación este ejecutándose, se recomienda el monitoreo de estas estadísticas durante cierto período, por medio de esta selección:

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('sorts(memory)', 'sorts(disk)');
```

Figura 69 Instrucción de selección usada para obtener el número de ordenamientos en disco y en memoria.

El resultado de esta selección puede ser como el siguiente:

```
NAME                                                    VALUE
---------------------------------------------------- ---------
sorts(memory)                                             965
sorts(disk)                                                 8
```

Figura 70 Ejemplo de la salida de la selección de ordenamientos (sorts).

6.3.2.2 Incrementar el tamaño del área de ordenamientos

Si existiera un número considerable de ordenamientos, que requieran accesos a los segmentos temporales, se podría aumentar el tamaño del área de ordenamientos, por el parámetro llamado SORT_AREA_SIZE que es un parámetro de inicialización colocado en los archivos de parámetros. El valor máximo de este parámetro varia dependiendo del sistema operativo.

6.3.2.3 ¿Qué beneficios se ganan cuando se tienen áreas de ordenamientos largos?

Cuando el área de ordenamientos es grande, también se incrementa el tamaño de cada ejecución, mientras que se decrementa el número total de ejecuciones. Esto implica que también se reducirá el número de mezclas (merges) que el servidor de Oracle debe de llevar a cabo para obtener el resultado final ordenado.

6.3.2.4 Posibles incovenientes cuando se tienen áreas de ordenamiento grandes

Cuando se incrementan las áreas de ordenamiento, esto causa que cada proceso de Oracle que hace ordenamientos, reserve mas memoria. Este incremento reduce la cantidad de memoria disponible para áreas privadas de

SQL y de PL/SQL. También afecta la cantidad de memoria disponible a nivel del sistema operativo, y puede inducir a pagineo y a swapeo. Antes de incrementar el tamaño de las áreas de ordenamientos, se debe de asegurar de tener suficiente memoria disponible en su sistema operativo para acomodar dichas áreas.

Se debe de considerar decrementar el tamaño de área retenida para ordenamientos, esto es el tamaño en que Oracle reduce sus áreas de ordenamientos, si no se espera que sus datos sean referenciados pronto. Este decremento se puede hacer por el parámetro llamado SORT_AREA_RETAINED_SIZE. Un valor pequeño en este parámetro, reduce la memoria usada, pero puede causar accesos adicionales en lectura y escritura de datos y en segmentos temporales en disco.

6.3.2.5 Como optimizar el rendimiento de ordenamientos por el "tablespace" TEMPORAL

Se puede optimizar el rendimento de los ordenamientos, especificando un tablespace como temporal (TEMPORARY) desde su creación (o por medio de una instrucción de alteración), y llevando a cabo los ordenamientos en este "tablespaces".

Normalmente, un ordenamiento puede requerir mucho espacio para reservar llamadas, que son almacenadas y eliminadas de segmentos temporales. Si el tablespace es especificado como TEMPORARY, solamente un segmento de ordenamiento es tomado del tablespace, y es usado para cada instancia que requiera la operación de ordenamiento. Este esquema ignora el mecanismo de reservación de espacio, y puede mejorar el rendimiento de

ordenamientos de tamaño mediano, que no pueden estar completamente en memoria.

6.3.2.6 Evitando ordenamientos

Una causa por la que se pueden originar ordenamientos, esta en la creación de índices. Cuando se crea un índice para una tabla, este proceso envuelve ordenamientos de todas las tuplas en la tabla base, de la columna o columnas que están siendo usadas en el índice.

Oracle también puede crear índices sin ordenamientos. Si las tuplas de la tabla son cargadas en orden ascendente, se pueden crear índices mucho mas rápidamente.

6.3.2.7 La opción NOSORT

Para crear un índice sin ordenamiento, se debe cargar las tuplas en la tabla en orden ascendente, específicamente sobre las columnas del índice.

Cuando se crea un índice se puede usar la opción NOSORT en el comando CREATE INDEX. Por ejemplo, se creara el índice llamado EMP_INDEX sobre la columna ENAME de la tabla de EMP, sin hacer ordenamientos de las tuplas:

```
CREATE INDEX emp_index
ON emp(ename)
NOSORT;
```

Figura 71 Creación de un índice para que no utilice ordenamiento.

Se recomienda usar esta opción, cuando la máquina es monoprocesadora; Cuando la máquina es multiprocesadora, entonces se recomienda tomar ventaja del procesamiento paralelo, agregándole entonces el respectivo grado de paralelismo.

7. Caso de estudio

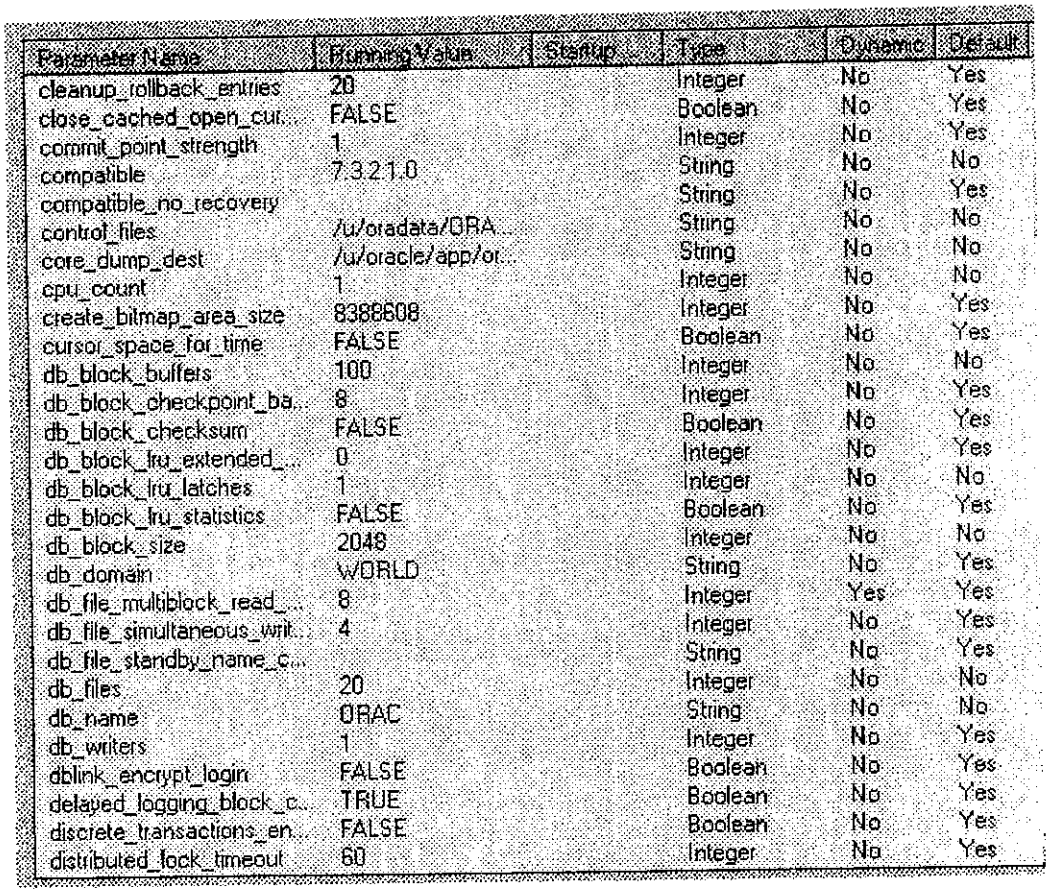
Entre los manejadores que se han estudiado en los anteriores capítulos, se ha escogido Oracle para realizar el caso de estudio, tanto por la gran cantidad de instalaciones que se encuentran en nuestro medio, como por la disponibilidad que se tiene de éste.

En el siguiente capítulo se tomarán aspectos estudiados teóricamente en los anteriores capítulos, y aplicados prácticamente en este. Las pruebas se harán específicamente con la versión 7.3.3.0 de Oracle Enterprise, sobre una máquina Hewlett Packard con HP/UX v. 10.01.

7.1 Manejo de memoria

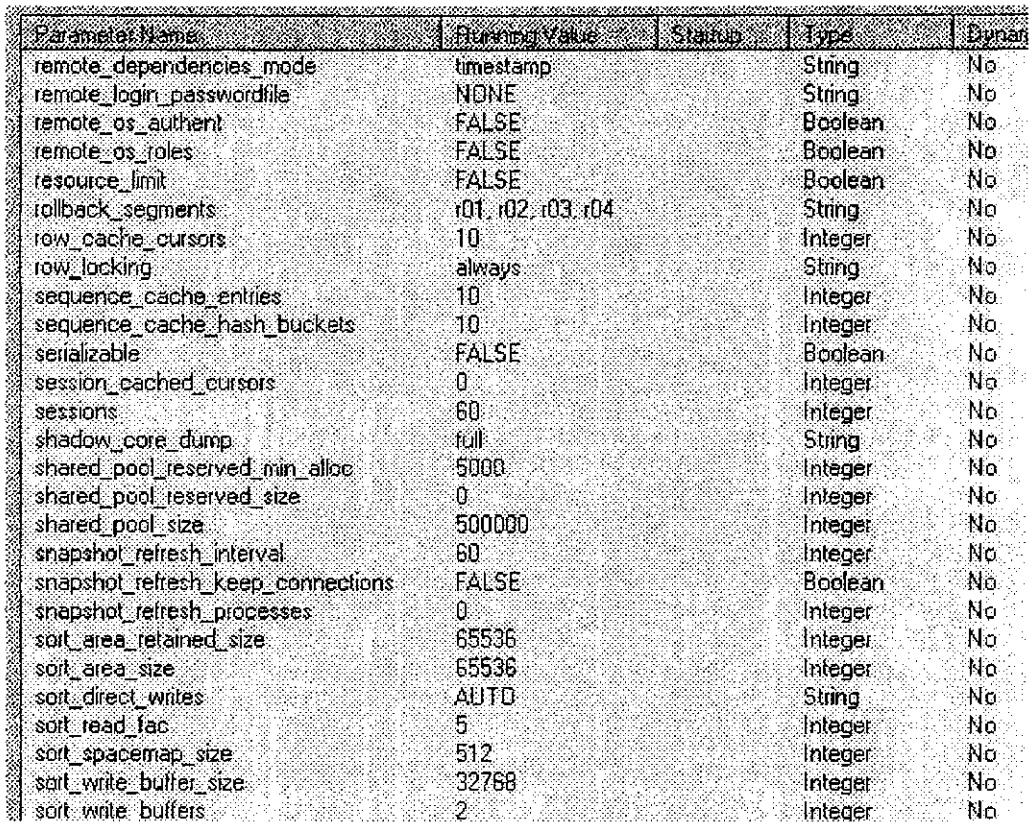
Como se ha visto en los capítulos anteriores, después del afinamiento de aplicaciones, conviene realizar afinamiento de memoria. Según lo se ha visto, sobre este manejador de base de datos, se debe de realizar afinamiento del "shared pool size", el cual esta dividido en el "cache" de librerías, y el "cache" del diccionario. A continuación corresponde el afinamiento del caché de buffers; Veamos el siguiente ejemplo:

Tenemos una base de datos, cuyos parámetros de inicialización son los siguientes:



Parameter Name	Running Value	Startup	Type	Dynamic	Default
cleanup_rollback_entries	20		Integer	No	Yes
close_cached_open_cur...	FALSE		Boolean	No	Yes
commit_point_strength	1		Integer	No	Yes
compatible	7.3.2.1.0		String	No	No
compatible_no_recovery			String	No	Yes
control_files	/u/oradata/ORA...		String	No	No
core_dump_dest	/u/oracle/app/or...		String	No	No
cpu_count	1		Integer	No	No
create_bitmap_area_size	8388608		Integer	No	Yes
cursor_space_for_time	FALSE		Boolean	No	Yes
db_block_buffers	100		Integer	No	No
db_block_checkpoint_ba...	8		Integer	No	Yes
db_block_checksum	FALSE		Boolean	No	Yes
db_block_lru_extended_...	0		Integer	No	Yes
db_block_lru_latches	1		Integer	No	No
db_block_lru_statistics	FALSE		Boolean	No	Yes
db_block_size	2048		Integer	No	No
db_domain	WORLD		String	No	Yes
db_file_multiblock_read_...	8		Integer	Yes	Yes
db_file_simultaneous_writ...	4		Integer	No	Yes
db_file_standby_name_c...			String	No	Yes
db_files	20		Integer	No	No
db_name	ORAC		String	No	No
db_writers	1		Integer	No	Yes
dblink_encrypt_login	FALSE		Boolean	No	Yes
delayed_logging_block_c...	TRUE		Boolean	No	Yes
discrete_transactions_en...	FALSE		Boolean	No	Yes
distributed_lock_timeout	60		Integer	No	Yes

Figura 72 Gráfica que muestra la actual configuración de la instancia de Oracle.



Parameter Name	Running Value	Startup	Type	Dynan
remote_dependencies_mode	timestamp		String	No
remote_login_passwordfile	NONE		String	No
remote_os_authent	FALSE		Boolean	No
remote_os_roles	FALSE		Boolean	No
resource_limit	FALSE		Boolean	No
rollback_segments	r01, r02, r03, r04		String	No
row_cache_cursors	10		Integer	No
row_locking	always		String	No
sequence_cache_entries	10		Integer	No
sequence_cache_hash_buckets	10		Integer	No
serializable	FALSE		Boolean	No
session_cached_cursors	0		Integer	No
sessions	60		Integer	No
shadow_core_dump	full		String	No
shared_pool_reserved_min_alloc	5000		Integer	No
shared_pool_reserved_size	0		Integer	No
shared_pool_size	500000		Integer	No
snapshot_refresh_interval	60		Integer	No
snapshot_refresh_keep_connections	FALSE		Boolean	No
snapshot_refresh_processes	0		Integer	No
sort_area_retained_size	65536		Integer	No
sort_area_size	65536		Integer	No
sort_direct_writes	AUTO		String	No
sort_read_fac	5		Integer	No
sort_spacemap_size	512		Integer	No
sort_write_buffer_size	32768		Integer	No
sort_write_buffers	2		Integer	No

Figura 73 Continuación de la lista de parámetros de la actual instancia de Oracle.

como se puede notar, los parámetros DB_BLOCK_BUFFERS, y SHARED_POOL_SIZE se encuentran con los valores de 100 (bloques) y 500000 (bytes) respectivamente, estos son los parámetros que de manera significativa modifican el tamaño del área compartida de Oracle (llamado también SGA - SYSTEM GLOBAL AREA - área global del sistema).

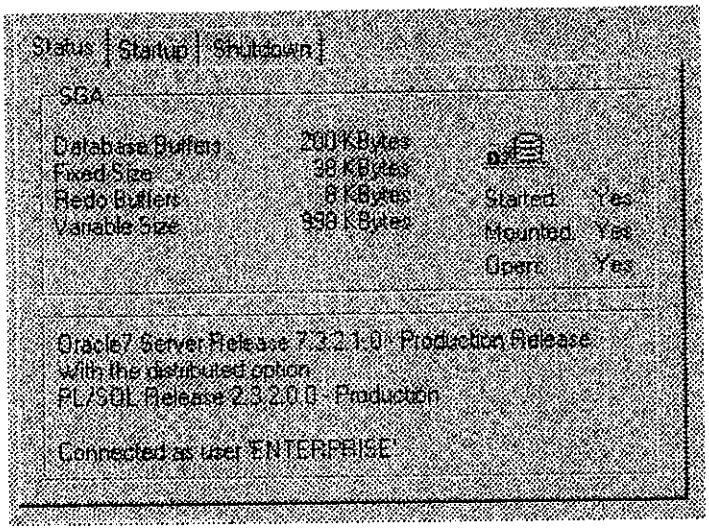


Figura 74 Valores actuales que conforman el área global del sistema (System Global Area - SGA) de la actual instancia de Oracle.

Según la figura anterior, el total de SGA es de 1.2 Megabytes. Se realizaron ciertas pruebas, que implican la ejecución de la siguiente selección:

```
select dname, ename, mgr, sal
from emp e, dept d
where e.deptno=d.deptno
```

Figura 75 Instrucción de selección usada para generar actividad en disco.

Esta misma selección se realizo en diez procesos distintos, de manera que el total de sesiones activas en la base de datos es representada en la siguiente gráfica:

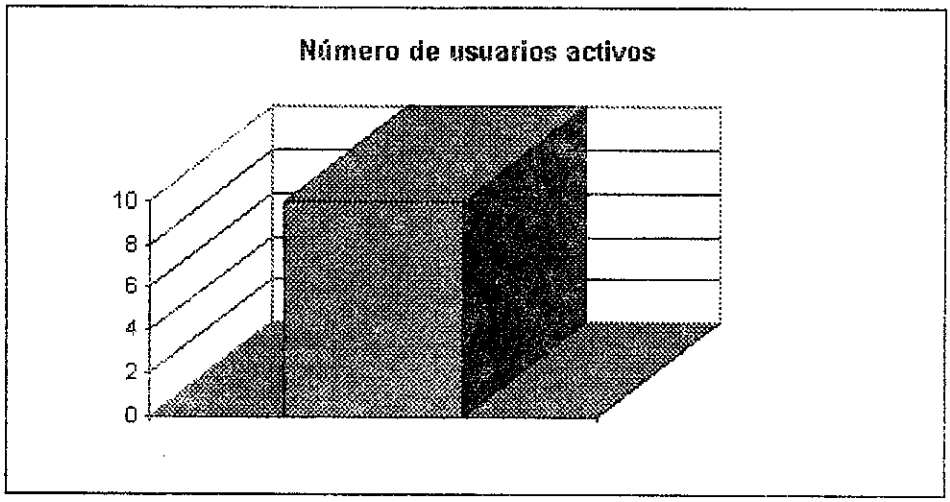


Figura 76 Gráfica que muestra el número total de usuarios activos.

Con este tamaño de SGA, y con esta carga de usuarios, se obtuvo un "hit ratio" de "cache" de librerías del:

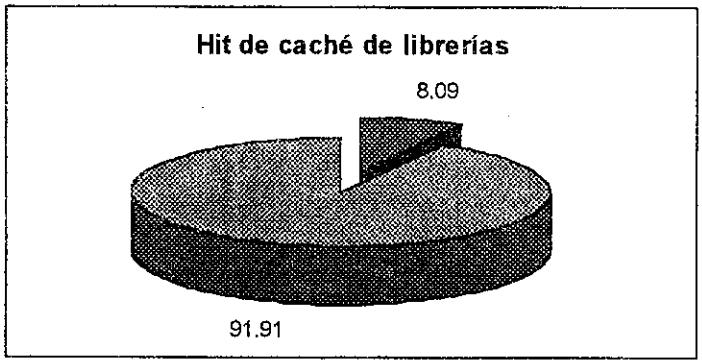


Figura 77 Gráfica que muestran los porcentajes de aciertos y de perdidas para el "caché" de librerías.

El caché del diccionario quedo así:

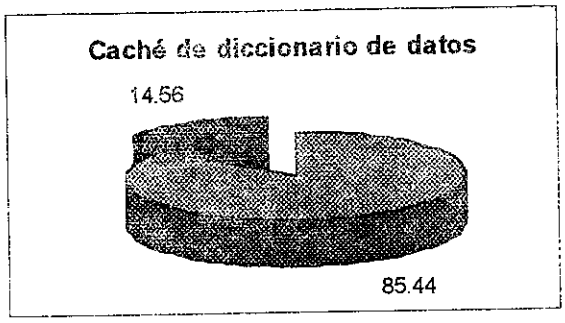


Figura 78 Gráfica que muestran los porcentajes de aciertos y de perdidas para el caché del diccionario de datos.

se obtuvo un "hit ratio" de "cache" buffers de un 68.33% (mostrado en la siguiente gráfica):

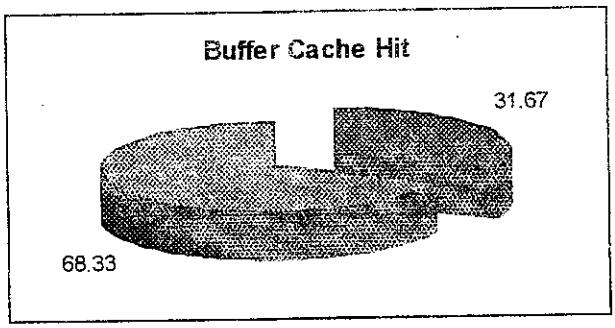


Figura 79 Gráfica que muestran los porcentajes de aciertos y de perdidas para el buffer "cache" de datos.

A continuación se realizo el aumento de los parámetros del "kernel" de "Oracle", de manera que quedaron de esta forma:

Parameter Name	Running Value	Startup Value	Type	D
cleanup_rollback_entries	20		Integer	N
close_cached_open_cursors	FALSE		Boolean	N
commit_point_strength	1		Integer	N
compatible	7.3.0.0		String	N
compatible_no_recovery			String	N
control_files	/u/oradata/...		String	N
core_dump_dest	/u/oracle/ap...		String	N
cpu_count	1		Integer	N
create_bitmap_area_size	8388608		Integer	N
cursor_space_for_time	FALSE		Boolean	N
db_block_buffers	1000		Integer	N
db_block_checkpoint_batch	8		Integer	N
db_block_checksum	FALSE		Boolean	N
db_block_lru_extended_st...	5000		Integer	N
db_block_lru_latches	1		Integer	N
db_block_lru_statistics	TRUE		Boolean	N
db_block_size	2048		Integer	N
db_domain	WORLD		String	N
db_file_multiblock_read_co...	16		Integer	Y
db_file_simultaneous_writes	4		Integer	N
db_file_standby_name_con...			String	N
db_files	20		Integer	N
db_name	ORAC		String	N
db_writers	1		Integer	N
dblink_encrypt_login	FALSE		Boolean	N
delayed_logging_block_cle...	TRUE		Boolean	N
discrete_transactions_enab...	FALSE		Boolean	N

Parameter Name	Running Value	Startup Value	Type	D
sequence_cache_hash_bu...	23		Integer	N
serializable	FALSE		Boolean	N
session_cached_cursors	0		Integer	N
sessions	115		Integer	N
shadow_core_dump	full		String	N
shared_pool_reserved_min...	5000		Integer	N
shared_pool_reserved_size	0		Integer	N
shared_pool_size	16000000		Integer	N
snapshot_refresh_interval	60		Integer	N
snapshot_refresh_keep_co...	FALSE		Boolean	N
snapshot_refresh_processes	0		Integer	N
sort_area_retained_size	65536		Integer	N
sort_area_size	65536		Integer	N
sort_direct_writes	AUTO		String	N
sort_read_fac	5		Integer	N
sort_spacemap_size	512		Integer	N
sort_write_buffer_size	32768		Integer	N
sort_write_buffers	2		Integer	N
spin_count	2000		Integer	N
sql92_security	FALSE		Boolean	N
sql_trace	FALSE		Boolean	N
temporary_table_locks	115		Integer	N
text_enable	FALSE		Boolean	N
thread	0		Integer	N
timed_statistics	TRUE		Boolean	Y
transactions	126		Integer	N
transactions_per_rollback_...	16		Integer	N

Figura 80 Parámetros del kernel de la actual instancia después de su modificación

Como podemos observar, el valor del parámetro DB_BLOCK_BUFFERS es ahora de 1000 bloque, y el del SHARED_POOL_SIZE es de 18000000 bytes (18Mb), por lo que el SGA queda de la siguiente manera:

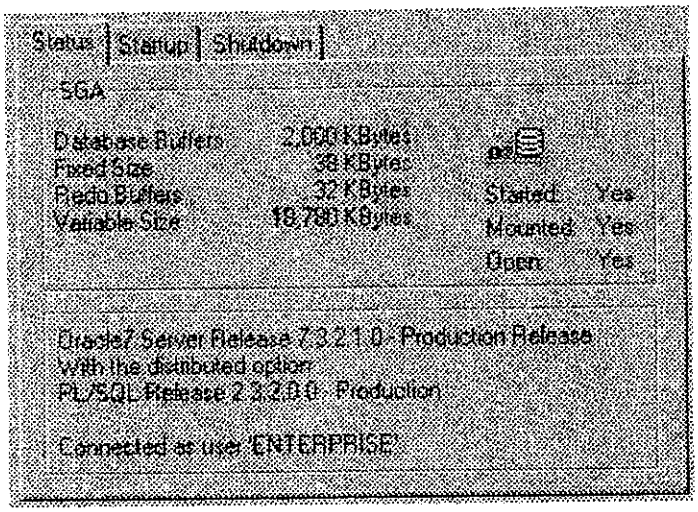


Figura 81 Valores actuales del SGA después de la modificación de los parámetros de la instancia de Oracle.

En este momento hemos aumentado el área global de sistema a un total de 21,301,580 bytes. Ya que hemos aumentado los respectivos parámetros del "kernel" de Oracle, observemos como es comportamiento del "Hit Ratio":

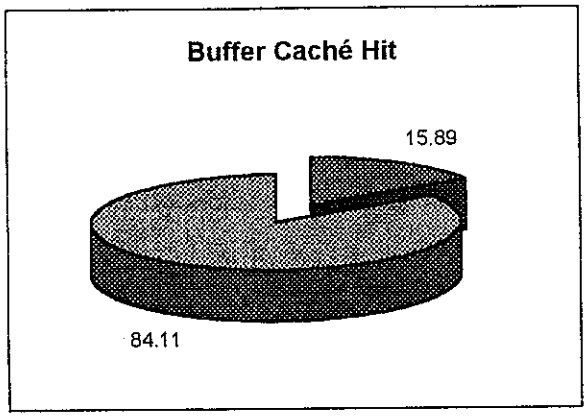


Figura 82 Gráfica que muestran los porcentajes de aciertos y de perdidas para el "buffer cache" de datos después de la modificación de la instancia.

El caché de librerías queda de la siguiente manera:

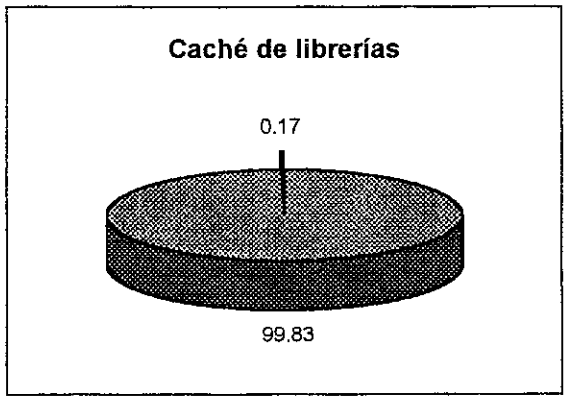


Figura 83 Gráfica que muestran los porcentajes de aciertos y de perdidas para el "cache" de librerías después de la modificación de la instancia.

Y el caché de diccionario de datos quedó así:

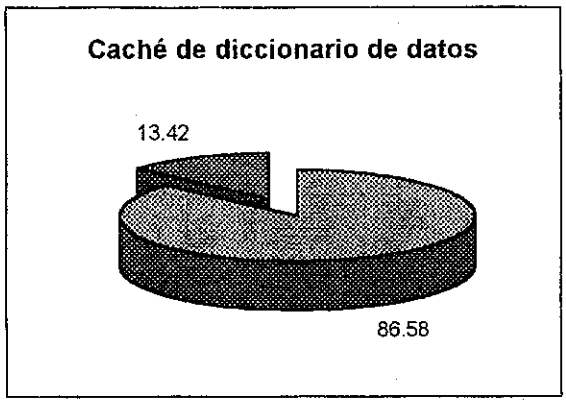


Figura 84 Gráfica que muestran los porcentajes de aciertos y de perdidas para el "cache" del diccionario de datos después de la modificación de la instancia.

Esto implica que definitivamente el rendimiento de la base de datos ha mejorado bastante. Y ahora se encuentra el "Hit Ratio" a un valor aceptable. Estas gráficas fueron obtenidas por el producto llamado "Enterprise Manager", y básicamente lo que hace es utilizar las sentencias "SQL select' vistas en el capítulo número 6.

7.2 Manejo de disco

En el siguiente caso, se crea un total de 100,00 tuplas en una tabla llamada prueba, que pertenece a un usuario llamado "test"; dicho usuario se encuentra creado sobre un tablespace del mismo nombre que solamente contiene un archivo de datos "Datafile". A continuación se tomarán estadísticas de dicha información cuando se estén creando este conjunto de tuplas.

Figura 85: Posición y nombres de los archivos de datos en la base de datos actual.

Como podemos notar, entre los archivos de datos de mayor actividad, encontramos al de des-hecho ("rollback"), lo cual es completamente normal, puesto que la mayoría de acciones en este momento es de inserción.

El otro "tablespace" que esta siendo utilizado es el de "test", el cual tiene la carga de las inserciones.

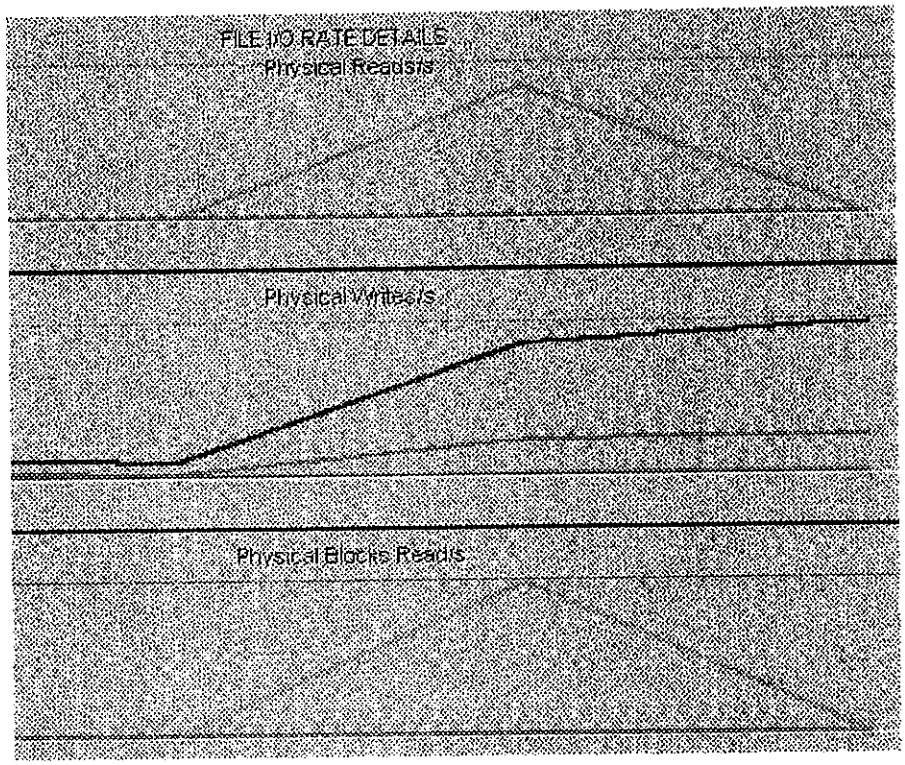


Figura 86: Gráfica que representa el movimiento de los archivos de datos

En esta gráfica, podemos notar esta actividad nuevamente, pero con valore numéricos. En este caso, el campo de archivo "File#' identifica al archivo de datos correspondientes. Para este caso el archivo de rollbacks es el número 14, y el de "test" es el número 17, los cuales son los de mayor actividad.

```
     FILE#     PHYRDS    PHYWRTS  PHYBLKRD PHYBLKWRT    READTIM   WRITETIM
---------- ---------- ---------- ---------- ---------- ---------- ----------
         1        795         53       2396         53       1284        174
         2          7       7296          7       7296          0      32248
         3       1034        106      14480        106        205        538
         4          1          0          1          0          4          0
         5          0          0          0          0          0          0
         6          0          0          0          0          0          0
         7          0          0          0          0          0          0
         8          0          0          0          0          0          0
         9          0          0          0          0          0          0
        10          0          0          0          0          0          0
        11          0          0          0          0          0          0
        12          0          0          0          0          0          0
        13          0          0          0          0          0          0
        14        369        155        369        155        192        476
        15          0          0          0          0          0          0
        16          0          0          0          0          0          0
        17         23       1804        237       1804         88       8269

17 rows selected.

SQL> l
  1* select * from v$filestat
SQL>
```

Figura 87: Muestra la instrucción de selección y los resultados de escrituras y lecturas físicas y lógicas de archivos de datos

En donde:

FILE#: Es la columna que identifica de manera única a cada uno de los archivos de datos.

PHYRDS: Es el total de lecturas físicas para un determinado archivo de datos.

PHYWRTS: Es el total de escrituras físicas para un determinado archivo de datos.

READTIME: Es el tiempo en que realizo dichas lecturas físicas.

WRITETIME: Es el tiempo en que realizo dichas escrituras físicas.

Debido a que la vista llamada V$FILESTAT solamente muestra el identificador del archivo de datos, se debe de ejecutar la siguiente selección para determinar el nombre de archivo de dato exacto:

```
select file_id, file_name, tablespace_name
 from dba_data_files;
```

Figura 88: Instrucción de selección para monitoreo de tablespaces y archivos de datos

y ésto dará como resultado una lista que incluye el identificador del archivo de datos, el nombre del archivo y su camino de acceso a nivel del sistema operativo, y el nombre de su "tablespace".

```
FILE_ID FILE_NAME                                  TABLESPACE_NAME
------- ------------------------------------------ ------------------
      1 /u/oracle/oradata/ORAC/system01.dbf        SYSTEM
      2 /u/oracle/oradata/ORAC/rbs01.dbf           RBS
      3 /u/oracle/oradata/ORAC/temp01.dbf          TEMP
      4 /u/oradata/ORAC/tools01.dbf                TOOLS
      5 /u/oracle/oradata/ORAC/users01.dbf         USERS
      6 /u/oracle/oradata/ORAC/summit01.dbf        SUMMIT
      7 /u/oracle/oradata/ORAC/holly_tab01.dbf     HOLLY_TAB
      8 /u/oracle/oradata/ORAC/holly_idx01.dbf     HOLLY_IDX
      9 /u/oracle/oradata/ORAC/canal3              CANAL3
     10 /u/oracle/oradata/ORAC/produc01.dbf        PRODUC
     11 /u/oracle/oradata/ORAC/prue01.dbf          PRUEBA
     12 /u/oracle/oradata/ORAC/prue02.dbf          PRUEBA
     13 /u/oracle/oradata/ORAC/bodega.dbf          BODEGA
     14 /u/oracle/oradata/ORAC/enterpri01.dbf      ENTERPRISE
     15 /u/oracle/oradata/ORAC/tbs_nma01.dbf       TBS_NMA
     16 /u/oracle/oradata/ORAC/tbs_idx01.dbf       TBS_INDICE
     17 /u/oracle/oradata/ORAC/test01.dbf          TEST

17 rows selected.

SQL> l
  1* select file_id, file_name, tablespace_name from dba_data_files
SQL>
```

Figura 89: Ejemplo de salida de archivos de datos y tablespaces.

Algo que es recomendable según se observo en el capítulo anterior, es el hecho de dividir los objetos en diferentes archivos que estén físicamente en distintos discos (stripped). Además es recomendable que las tablas estén en diferente disco que su respectivo índice, para evitar contención. Normalmente estas prácticas son muy útiles cuando se tienen servidores multiprocesadores, en los cuales se pueden levantar distintos procesos para que cada uno se encargue de cierta parte de una instrucción DML.

En este punto realizaremos un "Stripped". Veamos:

Crearemos nuevamente el tablespace de "TEST", de manera que sus archivos de datos queden en distintos sistemas de archivos o "FileSystems" (los cuales han sido creados por el administrador del sistema en diferentes discos físicos:

```
create tablespace test
datafile '/u/oradata/ORAC/test01.dbf' size 4M,
'/u/oracle/oradata/ORAC/test02.dbf' size 4M,
'/u/oracle/app/oracle/product/7.3.2/dbs/test03.dbf' size 4M;
```

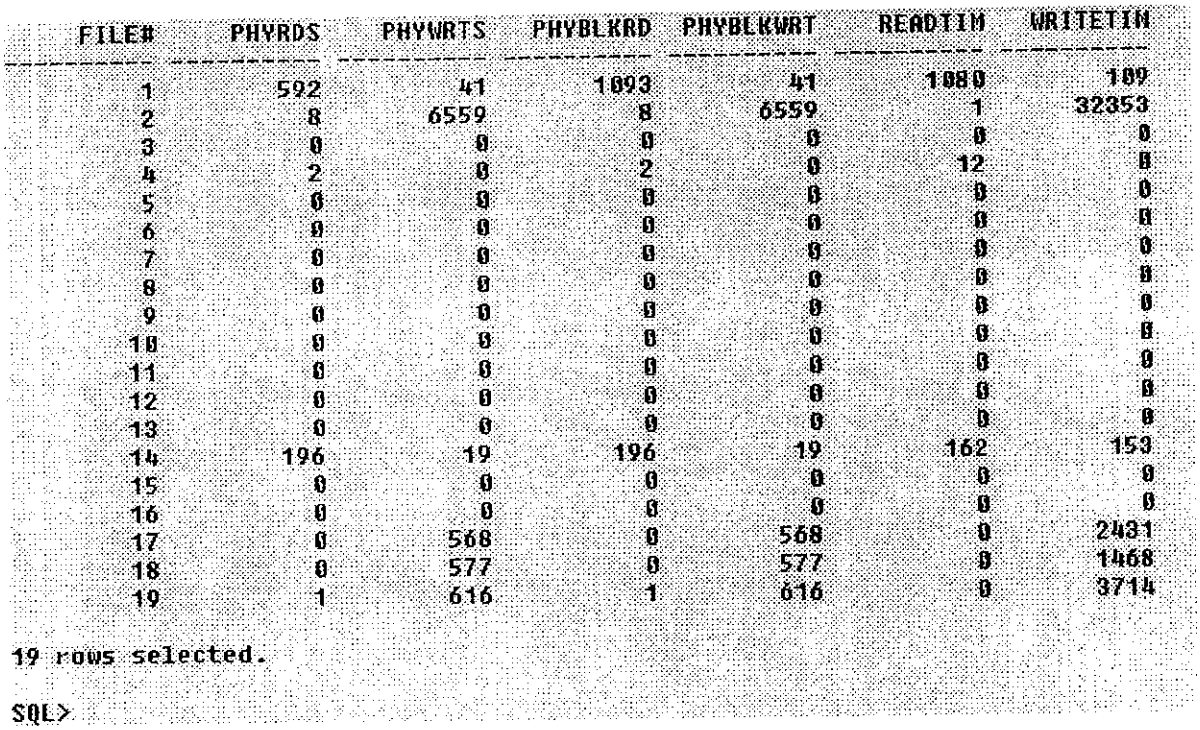
Figura 90: Instrucción de creación de tablespace con tres archivos de datos (data files).

```
FILE_ID FILE_NAME                                          TABLESPACE_NAME
------- -------------------------------------------------- ------------------
      1 /u/oracle/oradata/ORAC/system01.dbf                SYSTEM
      2 /u/oracle/oradata/ORAC/rbs01.dbf                   RBS
      3 /u/oracle/oradata/ORAC/temp01.dbf                  TEMP
      4 /u/oradata/ORAC/tools01.dbf                        TOOLS
      5 /u/oracle/oradata/ORAC/users01.dbf                 USERS
      6 /u/oracle/oradata/ORAC/summit01.dbf                SUMMIT
      7 /u/oracle/oradata/ORAC/holly_tab01.dbf             HOLLY_TAB
      8 /u/oracle/oradata/ORAC/holly_idx01.dbf             HOLLY_IDX
      9 /u/oracle/oradata/ORAC/canal3                      CANAL3
     10 /u/oracle/oradata/ORAC/produc01.dbf                PRODUC
     11 /u/oracle/oradata/ORAC/prue01.dbf                  PRUEBA
     12 /u/oracle/oradata/ORAC/prue02.dbf                  PRUEBA
     13 /u/oracle/oradata/ORAC/bodega.dbf                  BODEGA
     14 /u/oracle/oradata/ORAC/enterpri01.dbf              ENTERPRISE
     15 /u/oracle/oradata/ORAC/tbs_nma01.dbf               TBS_NMA
     16 /u/oracle/oradata/ORAC/tbs_idx01.dbf               TBS_INDICE
     17 /u/oradata/ORAC/test01.dbf                         TEST
     18 /u/oracle/oradata/ORAC/test02.dbf                  TEST
     19 /u/oracle/app/oracle/product/7.3.2/dbs/test03.dbf  TEST

19 rows selected.

SQL>
```

Figura 91: Ejemplo de como han quedado distribuidos los archivos de datos del tablespace "test" en diferentes sistemas de archivos (file systems)

En la figura anterior podemos notar como quedan los respectivos archivos de datos, que pertenecen al tablespace "TEST".

Veamos la carga que representa para cada archivo de dato, a la hora de correr el mismo proceso de 100,000 registros, (examinado por la vista V$FILESTAT):



FILE#	PHYRDS	PHYWRTS	PHYBLKRD	PHYBLKWRT	READTIM	WRITETIM
1	592	41	1093	41	1080	109
2	8	6559	8	6559	1	32353
3	0	0	0	0	0	0
4	2	0	2	0	12	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	196	19	196	19	162	153
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	568	0	568	0	2431
18	0	577	0	577	0	1468
19	1	616	1	616	0	3714

19 rows selected.

SQL>

Figura 92: Monitoreo de los archivos de datos para el tablespace "Test".

Para los archivos de datos 17, 18 y 19 (que son los correspondientes test01, test02 y test03) la carga ha sido distribuida en los tres distintos discos.

En una máquina multiprocesadora con tres procesadores paralelos, esto implicaría que instrucciones como selecciones, el tiempo de respuesta seria de la tercera parte de lo que normalmente ocuparía.

Como se puede ver a lo largo de este trabajo, la tarea de afinamiento, requiere de conocimiento de la herramienta, de las aplicaciones, del sistema operativo e incluso del hardware. Las personas encargadas de realizarlo, tienen una ardua tarea, y bajo sus hombros esta el peso de mantener el sistema funcionando adecuada y eficientemente. Esperando que este trabajo sea de utilidad, tanto para tomar la decisión, de que manejador de base de datos es el que mas se adecua a su organización, como para obtener el

concepto general de la forma en que diferentes manejadores de bases de datos deben ser afinados: en general todos funcionan casi de la misma manera, todos tienen archivos de bitácora, todos usan porciones de memoria compartida para realizar sus operaciones, todos pueden sufrir de contención, Etc.

Este trabajo es como un pequeño vistazo de lo que realmente se puede hacer. El afinador (dependiendo del campo en que se desenvuelva, administrador de base de datos, programador, diseñador, analista, etc) tendrá entonces la responsabilidad de investigar y aplicar los conceptos que se adecuen a su caso.

CONCLUSIONES

1. Cuando lo sistemas de bases de datos están en un nivel óptimo de funcionamiento, generalmente se puede dar un mejor servicio (queda en manos del recurso humano su adecuada utilización).

2. Para poder obtener un sistema con un grado óptimo de funcionamiento, es necesario realizar el proceso de afinación.

3. La afinación de una base de datos, es un proceso repetitivo, y en muchas ocasiones "de prueba y error".

4. La afinación de un sistema computacional se debería realizar aún cuando está en la etapa de análisis, y durante todo el ciclo de vida del sistema, de lo cual se deduce que existe variedad de personas implicadas en este proceso.

5. Dar lineamientos de cómo, cuándo y por qué se debe realizar un afinamiento es bastante útil, porque es un inicio que normalmente no se tiene a la hora de afinar sistemas.

6. El conocimiento de las características de distintos manejadores de bases de datos (en general de diverso software) ayuda generalmente a la toma de decisiones, que ayudan a obtener el software que más se adecue a las necesidades de la empresa.

7. La utilización de software (creado ya sea por terceras persona o por el manejador), especializado en afinamiento y monitoreo, puede ayudar de manera positiva a mantener el sistema computacional en un estado óptimo, y a facilitar la toma de decisiones a los respectivos afinadores del sistema para realizar cambios específicos que mejoren el tiempo de respuesta.

8. La mayoría de manejadores de bases de datos relacionales son muy parecidos en su arquitectura, por lo que para afinarles se puede tomar una metodología; sin embargo, es necesario investigar elementos cómo la sintaxis, valores que pueden ser adecuados, etc. para afinar bien un manejador de base de datos específico.

9. Existe una estrecha relación entre el hardware, el sistema operativo, y el manejador de bases de datos, por lo que afinar el manejador de base de datos, sin haberlo hecho con el sistema operativo (por ejemplo) resulta un proceso poco eficiente.

RECOMENDACIONES

- Utilizar el monitoreo y el afinamiento frecuentemente. Esto permite ser previsor. De lo contrario, se cae en el error de tratar de solucionar los problemas cuando éstos suceden, con el inconveniente de que muchas veces, ya es un sistema en producción, lo cual implica un desperdicio de tiempo tanto para el afinador como para las personas que utilizan el sistema.

- Después de seleccionar el manejador de base de datos que más se adecue a las necesidades de la empresa, tratar de adquirir la mayor cantidad posible de conocimiento sobre el producto. Esto permite (a parte de poder afinar bien el producto - caso de Ingres) obtener diferentes opciones de cómo realizar alguna acción, y analizar y escoger la mejor.

- Contratar gente específica para cada puesto, por ejemplo, un conjunto de analistas (probablemente ellos también sean los programadores), un administrador de la base de datos, un administrador del sistema operativo (que en algunos casos podría ser el mismo administrador de la base de datos), etc. No es recomendable que si a un programador le dio un problema correspondiente a la administración de la base de datos, que él arregle dicho problema, debido a que muchas veces no se cuenta con el correspondiente conocimiento para resolverle eficientemente.

BIBLIOGRAFÍA

1. Baird II, Willard. **DBMS - Tuning your Oracle7 Database**. USA: Oracle Corporation, 1,995.

2. Correy, Michael J., Abbey, Michael, Dechichio, Daniel J. **Caracterizacion de Oracle para una productividad y rendimientos óptimos**. México: McGraw Hill, 1,995.

3. Corrigan, Peter and Gurry, Mark. **Oracle performance tuning.** USA: Prentice Hall, 1,992

4. Dennis E. Shasha. **Database tuning: A principled approach.** USA: Prentice Hall, 1,992

5. Kirkwood, John. **Sybase architecture and administration**. USA: Prentice Hall, 1,993.

6. **Oracle 7 Server - Administrator's Guide.** USA: Oracle Corporation, 1,992.

7. **Proceedings IOUW 95**. USA: Oracle Corporation, Septiembre 17-22, 1,995 (Disco compacto).

8. Suto, Elizabeth. **Informix On-Line - Performance tuning.** USA: Prentice Hall, 1,995.

9. **System 10, Performance tuning Version 2. Student manual course**. USA: Sybase Inc., 1,994.

10. Loney, Kevin M. **Oracle magazine Article - HIT RATIOS**. Volume VI/Number 3. Pages, 43-46. USA: Spring 1,991.

11. Loney, Kevin M. **Oracle Magazine Article -Performance tuning User's guide**. volume V/Number 2. Pages, 51-54. USA: Spring 1,991.