Análisis de rendimiento de Solaris

31 de agosto de 2014

Índice

1. Objetivo

El objetivo de este documento es realizar un análisis de los principales motivos por los que un sistema bajo Solaris tiene un rendimiento bajo. Con los conocimientos adquiridos a través de este documento el lector debe ser capaz de

- Identificar cuellos de botella en us sistema Solaris en marcha.
- Realizar las recomendaciones apropiadas para aumentar el rendimiento.
- No se van a tener en conocimientos de tunning del kernel.

No es el objetivo de este documento tratar el tunning del kernel, ya que requiere conocimientos detalladas de la configuración de los sistemasen la máquina, y una mala configuración del nucleo por falta de información de los aplicativos en el sistema puede provocar que el rendimiento caiga mucho.

2. Procedimiento a seguir

3. Principales cuellos de botella

3.1. Cuellos de botella por disco

Normalmente los problemas de cuellos de botella por disco vienen motivados por las siguientes causas:

- Insuficiencia de capacidad (esto provoca normalmente gran fragmentación de los ficheros).
- Tiempo de respuesta de los discos bajo (demasiados procesos trabajando sobre el mismo disco hace caer el tiempo de servicio).
- Entorno (configuración) pobre (una mala configuración del disco puede hacer que las cabezas se muevan demasiado para encontrar los datos críticos).
- Tipo de configuración del RAID.
- Errores en los file system.
- Errores en las BDs.

3.2. Identificación de los cuellos de botella de disco

Se deben buscar distribuciones de I/O desiguales, discos muy cargados de trabajo tienen tiempos lentos, ¿preguntarse porque?

Tiempos de servicio por encima $> 50ms$ indican demasiado movimiento de las cabeza lectora.

Discos con una ocupación mayor del $65\,\%$ significan que son discos que trabajan demasiado, y vamos a tener retardos seguros, hay que intentar que el ratio de ocupación se manttenga dentro del $35\,\%$.

La longitud de la cola de espera indica el número de peticiones I/O esperando para servicio mientras el disco esta ocupado.

La cache de los arrays activadas proporcionan tiempos de servicios muy bajos, hay que saber que tipo de array se tiene y de que tipos de cache dispone.

El comando **iostat**. Los modificadores que nos interesan son los siguientes:

- -x estadisticas extendidas
- -n usa el nombre de los controladores
- -p da información individual por partición

iostat -x 30

El tiempo de servicio se genera de los siguientes componentes:

1. Tiempo de espera en cola
2. Comandos SCSI
3. Rastreo de las cabezas (1-15 msec)
4. Latencia de Rotación (0-10 msec)
5. Tiempo de transferencia de datos
6. Respuestas interrumpidas

Claramente el movimiento de la cabeza y la espera por un sector de la cabeza al girar es la pila de tiempo real en nuestro caso (esto es para el ejemplo de varias find). Incrementando la velocidad del lector (RPM) bajamos la latencia. Para una lectura particular, el correcto sector puede estar bajo la cabeza ahora (latencia 0) o puede haber pasado. Los vendedores de lectores de discos dicen que la latencia es de $\frac{1}{2}$ revolución.

Discos fragmentados, directorios largos, datos aleatorios producen más rastreos de cabezas las cuales pueden ser de 1 cilindro o de todos los cilindros (de dentro hacia fuera).

El trazado/formato/plan de los discos ahora se llama ZBR (banda de grabación de bit). Esto significa que hay más sectores en los cilindros de fuera (menos número de cilindros) que en los cilindros de dentro. Por tanto se puede reducir el número de rastreos de la cabeza poniendo los datos críticos en los cilindros bajos.

De la información que da este commando, las columnas importantes son: svc t, wait, %w, %b

- **svc t**: tiempo de servicio en el próximo cursor. Un tiempo de servicio significativamente lento, afecta (perjudicando) el rendimiento del disco.
- **wait**: es el número de entradas en la cola del dispositivo esperando para ser serviddas.

- **%w**: es el tanto por ciento de tiempo en en el que la cola ha estado ocupada. 100 % significa que la cola nunca ha estado vacia durante el periodo de tiempo.
- **%b**: es el tanto por ciento de tiempo que el dispositivo está ocupado. ¡35 % es bueno, ¿60 % es demasiado malo; esta regla es muy socorrida para medir el rendimiento.

Un valor de tiempo de servicio alrededor de 35 con un %b al 100 % es ligeramente alto pero se puede considerar razonable.

3.2.1. Que se puede hacer sobre los cuellos de botella de los discos

- Balancear la carga (particionando, striping)
- Mas discos
- UFS logging
- Distribuir la swap
- Usar los cilindros bajos para los datos críticos
- Poner la información relacionada en las mismas particiones
- No llenar el disco
- Añadir memoria (SSA/UFS/DB cache)

Realizar striping es la clásica solución de tener 6 operarios para realizar una zanja. Una operación de disco distribuida por software en multiples discos reduciendo los rastreos y el tiempo de transferencia. El problema de esto es que se corre un peligro inherente que normalmente se evita realizando mirror.

El loggin UFS es posible realizarlo desde Solaris 7, esto permite eliminar la necesidad de realizar las operaciones fsck después de una caida del sistema.

Las áreas de swap pueden estar distribuidas a través de los dispositivos y ser escritas de una manera distribuida.

Los datos relacionados en la misma partición significan que las cabezas no se tienen que mover muy lejos y muy a menudo.

Un disco lleno fuerza la fragmentación de los ficheros y por tanto incrementa el movimiento de las cabezas.

Mas memoria en las matrices de cache, o permitir a la computadora una mejor cache de datos reduce la necesidad de activar los mecanismos de los dispositivos. Por ejemplo, el incremento de la SGA de la DB, reduce la necesidad de viajar a través del código UFS, buscando y copiando los datos en la RAM de la UFS cache.

3.2.2. Bases de datos y file system

Los parámetros por defecto del comando newfs no son apropiados para las BD.

Los bloqueos de UFS de única escritura previenen multiples escrituras en el mismo fichero de DB.

Usar varios ficheros de BD.

Usar raw disk

Montar los filesystem usando la opción de I/O directorios

Unix se diseño para programadores y está optimizado para grandes números de pequeños archivos. Este modelo no es óptimo con ficheros grandes de BD.

La semantica de POSIX dicta que si un usuario esta escribiendo en un fichero, este es bloqueado hasta que la actualización se completa. Esto inhibe el rendimiento de las modernas DB multitreah. Usando raw disk, ficheros pequeños de DB o VXFs se pueden abandonar esa situación ya que se pueden romper los bloqueos de ficcheros de varios procesos.

Direct I/O es una opción de montaje que elimina el doble buffer de datos.

Comparación de Raw con UFS.

Concurrencia Direct I/O

This option isn't necessarily of any benefit to a Webserver filesystem unless you're sending out large fiels like in a code repository or large multimedia files such a movies o MP3s. If this is the case, then enablinig this option will speed up operation when reading from the filesystem. For small, static pages this option will have little to no effect.

From de mount_ufs man page, we see this explication:

"If forcedirectio is specified and suported by the file system, then for the duration of the mount forced direct I/O will be used. If the filesystem is mounted using forcedirectio, then data is transferred directly between user address space and the disk. If the filesystem is mounted using noforcedirectio, then the data is buffered in kernel address space when data is transferred between user address space and the disk. forcedirectio is a performance option that benefits only from large sequential data transfers. The default behavior is noforcedirectio"