Making swapping scalable

1. El subsistema de intercambio es donde las páginas anónimas (aquellas que contienen datos de programa no respaldados por archivos en el sistema de archivos).
2. En el momento en que un sistema Linux llega al punto donde está intercambiando páginas anónimas, la batalla de rendimiento ya se ha perdido. Por lo tanto, no es nada raro ver sistemas Linux configurados sin espacio de intercambio en absoluto.
3. El caso de usar swapping es cada vez más fuerte, por lo que hay valor en hacer swapping más rápido.
4. El intercambio es cada vez más atractivo, ya que aumenta el rendimiento de los dispositivos de almacenamiento, especialmente los dispositivos de almacenamiento de estado sólido (SSD).
5. No hace mucho tiempo, mover una página hacia o desde un dispositivo de almacenamiento era una operación increíblemente lenta, que tomaba varios órdenes de magnitud más tiempo que un acceso directo a la memoria.
6. Si el intercambio se puede hacer lo suficientemente rápido, la penalización de rendimiento para el exceso de memoria se vuelve insignificante, lo que lleva a una mejor utilización del sistema en su conjunto.
7. Como señaló Tim Chen en un conjunto de parches recientemente publicado, el kernel actualmente impone una sobrecarga significativa en las fallas de página que deben recuperar una página de intercambio. El conjunto de parches soluciona este problema aumentando la escalabilidad del subsistema de intercambio de varias maneras.
8. En los kernels actuales, un swap device (una partición dedicada o un archivo especial dentro de un sistema de archivos) está representado por una estructura swap_info_struct.
9. Entre los muchos campos de esa estructura es swap_map, un puntero a una matriz de bytes, donde cada byte contiene el recuento de referencia para una página almacenada en el dispositivo de intercambio.
10. Algunos de los códigos de intercambio son bastante antiguos; Una buena cantidad se remonta al comienzo de la era Git.
11. Mantener los datos en el dispositivo de intercambio debe reducir al mínimo la cantidad de búsqueda necesaria para acceder a ella. Funciona bastante menos bien en los dispositivos de estado sólido, por un par de razones:
    1. No hay retraso de búsqueda en tales dispositivos
    2. Los requisitos de nivel de desgaste de los SSD se cumplen mejor mediante la difusión del tráfico a través del dispositivo.
12. En un intento por mejorar el rendimiento en los SSD, el código de intercambio se cambió en 2013 para la versión 3.12. Cuando el subsistema de intercambio sabe que está trabajando con un SSD, divide el dispositivo en clústeres.
13. El "intercambio" se considera generalmente ser una palabra sucia entre los usuarios de largo plazo de Linux, que irán a menudo a las longitudes considerables evitarlo.
14. La memoria accesible por el usuario en un sistema Linux se divide en dos clases amplias: respaldada por archivos y anónima.
15. Las páginas respaldadas por archivos (o páginas de caché de páginas) corresponden a un segmento de un archivo en discos.
16. Las páginas anónimas no corresponden a un archivo en disco; Contienen los datos de tiempo de ejecución generado y utilizado por un proceso.
17. Recuperar una página anónima requiere escribir su contenido en el dispositivo de intercambio.
18. Como regla general, la recuperación de páginas anónimas (intercambio) se considera que es considerablemente más caro que recuperar páginas respaldadas por archivos. Una de las principales razones de esta diferencia es que las páginas con respaldo de archivos se pueden leer (y escribir en) almacenamiento persistente en trozos grandes y contiguos, mientras que las páginas anónimas tienden a dispersarse al azar en el dispositivo de intercambio.
19. En un dispositivo de almacenamiento giratorio, las operaciones de E / S dispersas son caras, por lo que un sistema que está haciendo un gran intercambio se ralentizará considerablemente.
20. Es mucho más rápido leer un montón de páginas con respaldo de archivos almacenadas secuencialmente y, dado que el archivo suele estar actualizado en el disco, es posible que esas páginas no necesiten escribirse en tiempo de recuperación.
21. Intercambiar es mucho más lento que muchos administradores tratan de configurar sus sistemas para hacer el menor intercambio posible.
22. Un paso intermedio es utilizar el botón de sintonización swappiness para polarizar el sistema fuertemente hacia la recuperación de páginas de archivos respaldados. Configuración swappiness a cero hará que el kernel para intercambiar solamente cuando la presión de memoria alcanza niveles graves.
23. Con la proliferación de dispositivos de E / S aleatorios rápidos como los SSD y la memoria persistente, sin embargo, el intercambio vuelve a interesarse de nuevo, no sólo como un desbordamiento de último recurso, sino como una extensión de memoria que puede usarse para optimizar el equilibrio entre la memoria El caché de página y el trabajo anónimo incluso durante una carga moderada.

24. No sólo el sistema debe estar más dispuesto a intercambiar memoria anónima, el intercambio puede ser una opción mejor que recuperar páginas de caché de página. Eso podría ser cierto si el dispositivo de intercambio es más rápido que las unidades utilizadas para almacenar archivos.
25. **Esa rotación cuesta un poco de tiempo de CPU. Si una lista particular de LRU tiene muchas páginas referenciadas en ella, la exploración de esa lista utilizará una cantidad relativamente grande de tiempo para un pago relativamente pequeño en páginas recuperables.**
26. **El conjunto de parches viene con una serie de puntos de referencia para mostrar su impacto en el rendimiento. Un punto de referencia de PostgreSQL va de 81 a 105 transacciones por segundo con los parches aplicados; La tasa de refracción se reduce a la mitad y el tiempo de CPU del núcleo se reduce.**
27. **Un punto de referencia de E / S de flujo continuo, que no debería crear una presión de memoria grave, se mantiene esencialmente sin cambios.**
28. **Sin embargo, los cambios en la gestión de la memoria están llenos de peligros potenciales, y es totalmente posible que otras cargas de trabajo se vean afectadas por estos cambios.**
29. **La manera de ganar confianza en que esto no sucederá es una prueba y una revisión más amplias. Este conjunto de parches es bastante joven; Ha habido algunas revisiones favorables, pero que la prueba todavía no ha sucedido.**
30. **Por lo tanto, puede pasar un tiempo antes de que este código vaya cerca de un kernel principal. Pero ha sido claro por un tiempo que el subsistema MM va a necesitar una serie de cambios para que su diseño se ajuste al hardware actual.**