Red Hat Enterprise Linux 6

Gu?a de ajuste de rendimiento

Optimización de resultados en Red Hat Enterprise Linux 6

Edición 4.0

Last Updated: 2018-04-16

Red Hat Enterprise Linux 6 Gu�a de ajuste de rendimiento

Optimización de resultados en Red Hat Enterprise Linux 6
Edición 4.0

Red Hat Expertos en el tema

Edited by

Don Domingo

Laura Bailey

Legal Notice

Resumen

La Guía de ajuste de rendimiento describe cómo optimizar el rendimiento de un sistema que ejecuta Red Hat Enterprise Linux 6. También documenta mejoras relacionadas con rendimiento en Red Hat Enterprise Linux 6. Aunque esta guía contiene procedimientos que se ensayan y prueban en el campo, Red Hat le recomienda ensayar todas las configuraciones planeadas en un entorno de prueba antes de aplicarlo al entorno de producción. Haga también una copia de seguridad de sus datos y configuraciones de preajuste.

Table of Contents

CAPÍTULO 1. VISIÓN GENERAL

La *Guía de ajuste de rendimiento* es una referencia completa sobre configuración y optimización de Red Hat Enterprise Linux. Aunque este lanzamiento también contiene información sobre funcionalidades de rendimiento en Red Hat Enterprise Linux 5, todas las instrucciones provistas aquí son específicas para Red Hat Enterprise Linux 6 .

Este libro se divide en capítulos que discuten subsistemas específicos en Red Hat Enterprise Linux. La *Guía de ajuste de rendimiento* se enfoca en tres temas principales por subsistema:

Funcionalidades

Cada capítulo de subsistema describe funcionalidades de rendimiento únicas (o implementadas de una forma diferente) en Red Hat Enterprise Linux 6. Estos capítulos también describen actualizaciones para Red Hat Enterprise Linux 6 que mejoraron el rendimiento de una forma significativa de subsistemas específicos en Red Hat Enterprise Linux 5.

Análisis

Este libro enumera los indicadores de rendimiento para cada subsistema. Los valores típicos para dichos indicadores se describen en el contexto de servicios específicos, ayudándole así a entender su importancia en el mundo real de los sistemas de producción.

Además, la *Guía de ajuste de rendimiento* también presenta diversas formas de recuperar datos de rendimiento para un sistema. Observe que algunas de las herramientas presentadas aquí se documentan en más detalle en otros documentos.

Configuración

Las instrucciones sobre cómo ajustar el rendimiento de un subsistema en Red Hat Enterprise Linux 6, son quizás la información más importante en este libro. La *Guía de ajuste de rendimiento* explica cómo ajustar un subsistema de Red Hat Enterprise Linux 6 para servicios específicos.

Recuerde que el ajuste de rendimiento de un subsistema puede algunas veces afectar de forma adversa el rendimiento de otro subsistema. La configuración predeterminada de Red Hat Enterprise Linux 6 es óptima para la *mayoría* de los servicios que se ejecutan en cargas *moderadas*.

Los procedimientos enumerados en la *Guía de ajuste de rendimiento* fueron probados a profundidad por ingenieros de Red Hat tanto en laboratorio como en campo. No obstante, Red Hat le recomienda que ensaye en un entorno de prueba seguro todas las configuraciones planeadas, antes de aplicarlas a sus servidores de producción. También debería hacer una copia de seguridad de todos los datos e información de configuración antes de iniciar el ajuste de su sistema.

1.1. AUDIENCIA

Este libro es apto para dos tipos de lectores:

Analista de negocios/sistemas

Este libro enumera y explica las funcionalidades de rendimiento de Red Hat Enterprise Linux 6 en un alto nivel, proporcionando información suficiente sobre cómo rinden los subsistemas para cargas de trabajo específicas (tanto predeterminadas como optimizadas). El nivel de detalle utilizado para describir las funcionalidades de rendimiento de Red Hat Enterprise Linux 6 ayuda a los clientes potenciales y a los ingenieros de ventas a entender la conveniencia de esta plataforma al proveer servicios intensivos de recursos en un nivel aceptable.

La *Guía de ajuste de rendimiento* también proporciona, en lo posible, enlaces de documentación más

detallada sobre cada funcionalidad. En ese nivel de detalle, los lectores pueden entender el rendimiento de estas funcionalidades para formar una estrategia de alto nivel al implementar y optimizar Red Hat Enterprise Linux 6. Esto permite a los lectores desarrollar *y* evaluar propuestas de infraestructura.

Esta característica enfocada en el nivel de documentación es apropiada para lectores con un alto nivel de entendimiento sobre los subsistemas de Linux y de redes a nivel empresarial.

Administrador de sistemas

Los procedimientos enumerados en este libro son apropiados para administradores de sistemas certificados con RHCE [1] nivel de destrezas (o su equivalente, es decir, 3 o 5 años de experiencia en implementar y administrar Linux). La *Guía de ajuste de rendimiento* tiene como objetivo proporcionar información detallada en lo posible sobre los efectos de cada configuración; esto significa la descripción de cualquier pérdida o ganancia en rendimiento que pueda ocurrir.

La destreza subyacente en el ajuste de rendimiento radica en no saber cómo analizar y ajustar un subsistema. En su lugar, un administrador de sistemas conocedor de ajustes de rendimiento sabe cómo balancear y optimizar un sistema de Red Hat Enterprise Linux 6 *para propósitos específicos*. Esto significa que *también* conoce qué pérdidas y ganancias y precio son aceptables al intentar implementar una configuración diseñada para impulsar un rendimiento de subsistema específico.

1.2. ESCALABILIDAD HORIZONTAL

Los esfuerzos de Red Hat por mejorar el rendimiento de Red Hat Enterprise Linux 6 se enfocan en la *escalabilidad*. Las funcionalidades de aumento de rendimiento se evalúan principalmente con base en la forma como afecta el rendimiento de plataforma en diferentes áreas del espectro de carga de trabajo—es decir, del servidor web solo a la unidad central de granja de servidores.

El enfoque en escalabilidad permite a Red Hat Enterprise Linux mantener la versatilidad para diferentes tipos de cargas de trabajo y propósitos. Al mismo tiempo, esto significa que cuando su empresa crece y su carga de trabajo aumenta en escala, la reconfiguración de su entorno de servidor es menos prohibitiva (en términos de coste y hora-persona) y más intuitiva.

Red Hat hace mejoras a Red Hat Enterprise Linux tanto para *escalabilidad horizontal* como para *escalabilidad vertical*; sin embargo, la escalabilidad horizontal suele ser el caso de uso más aplicable. La idea detrás de la escalabilidad horizontal es usar múltiples *computadores estándar* para distribuir cargas de trabajo pesadas con el fin de mejorar el rendimiento y la confiabilidad.

En una sala de servidores típica, estos computadores estándar vienen en forma de servidores montados en una unidad de rack (1U) y de cuchilla. Cada computador estándar puede ser tan pequeño como un sistema de dos sockets, aunque algunas salas de servidores usan grandes sistemas con más conectores. Algunas redes de grado empresarial mezclan sistemas grandes y pequeños; en cuyo caso, los sistemas grandes son servidores de alto rendimiento (por ejemplo, los servidores de base de datos) y los pequeños son servidores de aplicaciones dedicadas (por ejemplo, los servidores web o de correo).

Este tipo de escalabilidad simplifica el crecimiento de su infraestructura de TI: una empresa mediana con una carga apropiada solamente podría necesitar únicamente dos servidores de caja de pizza para ajustarse a todas sus necesidades. Cuando la empresa emplea más gente, expande sus operaciones y aumenta sus volúmenes de ventas, sus requerimientos de TI aumentan tanto en volumen como en complejidad. La escalabilidad horizontal permite a TI implementar máquinas adicionales con configuraciones (casi) idénticas a las de sus predecesores.

En resumen, la escalabilidad horizontal añade una capa de abstracción que simplifica la administración del hardware del sistema. Al desarrollar la plataforma de Red Hat Enterprise Linux para escalar de

forma horizontal, el aumento de habilidad y rendimiento de servicios de TI puede ser tan sencillo como añadir nuevas máquinas configuradas.

1.2.1. Computación paralela

Los usuarios se benefician de la escalabilidad horizontal de Red Hat Enterprise Linux, no solo porque simplifica la administración de hardware del sistema; sino porque la escalabilidad horizontal es una filosofía de desarrollo apropiada dadas las tendencias actuales de avance de hardware.

Considere lo siguiente: la mayoría de las aplicaciones más complejas tienen miles de tareas que deben realizarse de forma simultánea, con diferentes métodos de coordinación entre tareas. Mientras los primeros computadores tenían un procesador de núcleo individual para intercambiar todas estas tareas, los procesadores virtuales de ahora tiene múltiples núcleos de tareas. En efecto, los computadores modernos colocan múltiples núcleos en un conector individual, lo cual crea incluso en escritorios y portátiles de un conector individual, sistemas de multiprocesadores.

En 2010, los procesadores Intel y AMD tenían de 2 a 16 núcleos. Dichos procesadores eran servidores en forma de caja de pizza o cuchilla, los cuales ahora cuentan con unos 40 núcleos. Estos sistemas de alto rendimiento y bajo costo brindan grandes capacidades y funcionalidades dentro de la corriente principal.

Para alcanzar el mejor rendimiento y utilización de un sistema, cada núcleo debe mantenerse ocupado. Esto significa que 32 tareas independientes deben estar ejecutándose para aprovechar un servidor de cuchilla de 32 núcleos. Si un chasis contiene 10 de estas cuchillas de 32 núcleos, entonces toda la configuración puede procesar un mínimo de 320 tareas de forma simultánea. Si estas tareas hacen parte de una individual, dichas tareas deberán estar coordinadas.

Red Hat Enterprise Linux fue desarrollado para adaptar bien a las tendencias del hardware y asegurarse de que las empresas puedan beneficiarse totalmente de ellas. La Sección 1.3, “Sistemas distribuidos” explora en gran detalle las tecnologías que habilitan la escalabilidad horizontal de Red Hat Enterprise Linux.

1.3. SISTEMAS DISTRIBUIDOS

Para realizar una escalabilidad horizontal total, Red Hat Enterprise Linux usa varios componentes de *computación distribuida*. Las tecnologías que componen la computación distribuida se dividen en tres capas:

Comunicación

La escalabilidad horizontal requiere que muchas tareas se realicen de forma simultánea (en paralelo). Como tal, estas tareas deben tener *comunicación de interprocesos* para coordinar su trabajo. Además, una plataforma con escalabilidad horizontal debe poder compartir tareas a través de varios sistemas.

Almacenamiento

El almacenamiento a través de discos locales no es suficiente para los requerimientos de escalabilidad horizontal. Alguna forma de almacenaje compartido o distribuido es necesaria, una con una capa de abstracción que permita una capacidad de volumen para almacenaje individual crecer sin problemas con la adición de un nuevo hardware de almacenamiento.

Administración

La labor más importante en computación distribuida es la capa de *administración*. Esta capa de administración coordina todos los componentes de software y hardware, administrando de forma eficiente la comunicación, almacenaje y el uso de recursos compartidos.

Las siguientes secciones describen con más detalle las tecnologías dentro de cada capa.

1.3.1. Comunicación

La capa de comunicación asegura el transporte de datos y se compone de dos partes:

- Hardware
- Software

La forma más simple y rápida de comunicación es a través de *memoria compartida*. Esto implica el uso de operaciones de lectura y escritura de memoria familiar; que memoria compartida tenga un alto ancho de banda, latencia baja y baja sobrecarga de operaciones de memoria de lectura y escritura.

Ethernet

La forma más común de comunicación entre computadores es por Ethernet. Hoy en día, *Gigabit Ethernet* (GbE) se proporciona de forma predeterminada en sistemas y la mayoría de servidores incluyen 2a 4 puertos de Gigabit Ethernet. GbE proporciona un buen ancho de banda y latencia. Esta es la base de la mayoría de sistemas distribuidos en uso hoy en día. Incluso cuando los sistemas incluyen hardware de redes más rápidas, es común usar GbE para una interfaz de administración dedicada.

10GbE

Ethernet de 10 Gigabits (10GbE) está creciendo rápidamente en aceptación para servidores de alta gama e incluso para servidores de rangos medios. 10GbE proporciona diez veces el ancho de banda de GbE. Una de sus ventajas principales es con modernos procesadores multinúcleos, donde restaura el equilibrio entre comunicación y computación. Puede comparar un sistema de núcleo único mediante GbE con un sistema de ocho núcleos mediante 10GbE. Utilizado de esta forma, 10GbE es valioso para mantener el rendimiento de todo el sistema general y evitar cuellos de botella en comunicación.

Infortunadamente, 10GbE es costoso. Mientras el costo de NIC de 10GbE se reduce, el precio de interconexión (especialmente el de fibra óptica) permanece alto y los interruptores de redes de 10GbE son extremadamente costosos. Podemos esperar que estos precios declinen con el tiempo, pero 10GbE hoy es muy utilizado en redes de salas de servidores y aplicaciones de rendimiento crítico.

Infiniband

Infiniband ofrece incluso un mayor rendimiento que 10GbE. Además de las conexiones de redes TCP/IP y UDP utilizadas con Ethernet, Infiniband soporta comunicación de memoria compartida. Esto permite a Infiniband operar entre sistemas vía *acceso remoto de memoria* (RDMA).

El uso de RDMA permite a Infiniband trasladar datos directamente entre sistemas sin la sobrecarga de las conexiones o sockets TCP/IP. A la vez, esto reduce la latencia, lo cual es crítico para algunas aplicaciones.

Infiniband es muy utilizado en aplicaciones de *Computación técnica de alto rendimiento* (HPTC), las cuales requieren alto ancho de banda, baja latencia y baja sobrecarga. Muchas aplicaciones de supercomputación se benefician de ello al punto que la mejor forma de mejorar el rendimiento es invirtiendo en Infiniband en lugar de procesadores más rápidos o más memoria.

RoCCE

RDMA en Ethernet (RoCCE) implementa comunicaciones de estilo Infiniband (incluidas RDMA) en una infraestructura de 10GbE. Dado el costo de mejoras asociadas con el volumen creciente de productos de 10GbE, se puede esperar un uso más amplio de RDMA y RoCCE en un amplio rango de sistemas y aplicaciones.

Cada uno de estos método de comunicación tiene total soporte de Red Hat para uso con Red Hat Enterprise Linux 6.

1.3.2. Almacenamiento

Un entorno con computación distribuida usa múltiples instancias de almacenamiento compartido. Lo cual significa que:

- Los sistemas múltiples almacenan datos en un solo sitio
- La unidad de almacenamiento (e.j. un volumen) compuesto de varios aparatos de almacenaje

El ejemplo de almacenamiento más común es la unidad de disco local montada en un sistema. Esto es apropiado para operaciones de Tecnología informática donde todas las operaciones se albergan en un host o incluso, en una pequeña cantidad de hosts. Sin embargo, esto se dificulta y complica cuando la infraestructura escala a docenas o incluso cientos de sistemas, que administran cuantos discos locales de almacenaje sean posibles.

El almacenaje distribuido adiciona una capa para aliviar y automatizar la administración de hardware de almacenaje cuando el negocio escala. Al tener un grupo de instancias de almacenaje, se reduce el número de dispositivos que el administrador necesita manejar.

La consolidación de las funcionalidades de almacenaje de múltiples aparatos de almacenaje dentro de un volumen ayuda tanto a los usuarios como a los administradores. Este tipo de almacenaje distribuido proporciona una capa de abstracción de grupos de almacenaje: los usuarios ven una sola unidad de almacenaje, a la cual un administrador puede fácilmente adicionar más hardware. Algunas tecnologías que habilitan almacenaje distribuido también proporcionan beneficios adicionados, tales como conmutación y multirrutas.

NFS

El *Sistema de archivos de red* (NFS) permite múltiples servidores o usuarios montar y usar la misma instancia de almacenaje remoto a través de TCP o UDP. NFS es comúnmente utilizada por múltiples aplicaciones para guardar datos. También es conveniente para almacenaje en de grandes cantidades de datos.

SAN

Redes de área de almacenaje (SAN) usan tanto Canal de fibra o protocolo iSCSI para proveer acceso al almacenaje. La infraestructura de fibra de canal (tal como adaptadores de bus de host de canal de fibra, cambia y almacena arrays) combina rendimiento de alto rendimiento y alta banda ancha y almacenamiento masivo. Los SAN separan almacenaje del procesamiento, proporcionando flexibilidad considerable en el diseño del sistema.

La otra ventaja de SAN es que proporciona un entorno administrativo para realizar tareas administrativas de hardware de mayor almacenamiento. A saber:

- Control de acceso de almacenaje
- Administración de grandes cantidades de datos
- Aprovisionamiento de sistemas
- Copia de seguridad y replicación de datos
- Toma de instantáneas
- Soporte de conmutación de sistema

- Garantía de integridad de datos
- Migración de datos

GFS2

El *Sistema de archivos global 2* (GFS2) de Red Hat proporciona varias habilidades especializadas. La función básica de GFS2 es proporcionar un sistema de archivos único, incluido el acceso concurrente de lectura y escritura, compartido a través de múltiples miembros de un clúster. Esto significa que cada miembro del clúster ve exactamente los mismos datos "en disco" en el sistema de archivos denominado GFS2.

GFS2 permite a todos los sistemas tener acceso concurrente al disco. Para mantener integridad de datos, GFS2 usa un *Gestor de cerrojo distribuido* (DLM), el cual permite al sistema escribir a un sitio específico al mismo tiempo.

GFS2 es especialmente bien apto para aplicaciones de conmutación que requieren alta disponibilidad en almacenaje.

Para obtener mayor información sobre GFS2, consulte *Sistema de archivos global 2*. Para mayor información sobre almacenamiento en general, consulte la *Guía de administración de almacenaje*. Las dos están disponibles en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

1.3.3. Redes convergentes

La comunicación en redes se realiza normalmente a través de Ethernet con tráfico de almacenamiento mediante un entorno de canal de fibra SAN. Es común tener una red dedicada o enlace de serie para administración de sistemas y quizás aún *heartbeat*[2]. Como resultado, un servidor único suele estar en múltiples redes.

Las conexiones múltiples en cada servicio son costosas, voluminosas y complejas de manejar. Esto da lugar a la necesidad de consolidar todas las conexiones en una. El *Canal de fibra en Ethernet* (FCoE) y el *Sistema de Interfaz para computadoras pequeñas en Internet* (iSCSI) responden a esta necesidad.

FCoE

Con los comandos de FCoE, los comandos de canal de fibra estándar y los paquetes de datos se transportan en una infraestructura física 10GbE a través de una sola *tarjeta de red convergente* (CNA). El tráfico de Ethernet estándar TCP/IP y las operaciones de almacenamiento de canal de fibra se pueden transportar a través del mismo enlace. FCoE usa una tarjeta de interfaz de red física (y un cable) para múltiples conexiones lógicas de almacenamiento y red.

FCoE tiene las siguientes ventajas:

Número reducido de conexiones

FCoE reduce a la mitad el número de conexiones de red a un servidor. Usted puede tener múltiples conexiones para rendimiento o disponibilidad; sin embargo, una sola conexión proporciona conexión de almacenamiento y redes. Esto es muy útil para servidor de caja de pizza y servidor de cuchilla, ya que ambos tienen espacio muy limitado para componentes.

Menor costo

El número reducido de conexiones significa una reducción de cables, interruptores y otro equipo de redes. La historia de Ethernet también presenta grandes economías de escala; el costo de redes cae de forma dramática cuando la cantidad de dispositivos en el mercado pasa de millones a mil millones, como se vio declinar el precio de dispositivos de Ethernet de 100Mb y Ethernet de gigabits.

Igualmente, 10GbE serán más baratos cuando más negocios adapten su uso. Puesto que el hardware de CNA se integra en un solo chip, el uso extendido también aumentará su volumen en el mercado, lo cual se traducirá en una caída de precio significativa con el tiempo.

iSCSI

Internet SCSI (iSCSI) es otro tipo de protocolo de red convergente; es una alternativa para FCoE. Al igual que canal de fibra, iSCSI proporciona almacenamiento de nivel de bloques en una red. Sin embargo, iSCSI no proporciona una administración total del entorno. La ventaja principal sobre iSCSI en FCoE es que iSCSI proporciona mucha de la habilidad y flexibilidad de canal de fibra, pero a un bajo costo.

---

[1] Ingeniero certificado de Red Hat . Para obtener mayor información, consulte http://www.redhat.com/training/certifications/rhce/.

[2] *Heartbeat* es el intercambio de mensajes entre sistema para garantizar que cada sistema aún esté funcionando. Si un sistema "pierde latido" o 'heartbeat' se asume que ha fallado y se apaga y, otro sistema ocupa su lugar.

CAPÍTULO 2. FUNCIONALIDADES DE RENDIMIENTO DE RED HAT ENTERPRISE LINUX 6

2.1. SOPORTE PARA 64 BITS

Red Hat Enterprise Linux 6 soporta procesadores de 64 bits; dichos procesadores pueden usar en teoría hasta 16 *exabytes* de memoria. Con el lanzamiento de Red Hat Enterprise Linux 6 (GA) se ha probado y certificado el soporte de hasta 8TB de memoria física.

El tamaño de la memoria soportada por Red Hat Enterprise Linux 6 se espera que crezca en las siguientes actualizaciones menores, puesto que Red Hat sigue introduciendo y mejorando más características que permiten el uso de bloques de memoria más grandes. Ejemplos de estas mejoras (en el momento del lanzamiento de Red Hat Enterprise Linux 6) son:

- Páginas gigantes y páginas gigantes transparentes
- Mejoras Memoria de Acceso No-Uniforme

Estas mejoras se detallarán en las secciones siguientes.

Páginas gigantes y páginas gigantes transparentes

La implementación de *páginas gigantes* en Red Hat Enterprise Linux 6, permite al sistema administrar eficientemente el uso de memoria mediante cargas de trabajo de memoria. Las páginas gigantes usan 2 MB de páginas comparados con el tamaño estándar de página de 4 KB, lo cual permite a las aplicaciones escalar bien del procesamiento de GB a incluso TB de memoria.

Las páginas gigantes son difíciles de crear, administrar y usar. Para solucionarlo, Red Hat Enterprise 6, ofrece el uso de *páginas gigantes transparentes* (THP). Las THP automáticamente gestionan muchas de las complejidades implicadas en el uso de páginas gigantes.

Para obtener mayor información sobre páginas gigantes y páginas gigantes transparentes (THP), consulte la Sección 5.2, “Páginas gigantes y páginas gigantes transparentes”.

mejoras NUMA

Muchos de los sistemas ahora soportan *Memoria de Acceso No-Uniforme* (NUMA). NUMA simplifica el diseño y la creación de hardware para grandes sistemas; sin embargo, también añade una capa de complejidad para el desarrollo de aplicaciones. Por ejemplo, NUMA implementa memoria local y remota, donde acceder a la memoria remota puede tardar varias veces más que el acceso a la memoria local. Esta característica (entre otras) tiene muchas implicaciones en el rendimiento que impactan a los sistemas operativos, aplicaciones y configuraciones de sistema sobre los que deberían ser desplegados.

Red Hat Enterprise Linux 6 se ha mejorado para optimizar el uso de NUMA, gracias a varias características adicionales que ayudan a gestionar usuarios y aplicaciones en sistemas NUMA. Estas características incluyen afinidad de CPU, CPU pinning (conjunto de CPUs), numactl y grupos de control, que permiten a procesos (afinidad) o a aplicaciones (pinning) para "anclar" a una CPU específica o conjunto de CPUs.

Para obtener mayor información sobre el soporte NUMA en Red Hat Enterprise Linux 6, consulte la Sección 4.1.1, “Topología de CPU y Numa”.

2.2. CERROJOS EN BUCLE DE TIQUETES

Una parte clave de cualquier diseño de sistema, es garantizar que un proceso no altere la memoria utilizada por otro proceso. Los datos no controlados en memoria pueden corromper la información y

ocasionar fallas del sistema. Para evitarlo, el sistema operativo le permite a un proceso bloquear una parte de memoria, realizar una operación y luego desbloquear o liberar la memoria.

Una implementación de cerramiento de memoria común, es a través de 'Spinlocks' o *cerrojos en bucle*, los cuales le permiten a un proceso verificar continuamente si el cerrojo está disponible y tomarlo cuando lo esté, tan pronto como sea posible. Si hay varios procesos compitiendo por el mismo cerrojo, el primero en solicitar el cerrojo después de que haya sido liberado, lo obtendrá. Cuando todos los procesos tienen el mismo acceso a memoria, este enfoque es "justo" y funcionará bastante bien.

Lamentablemente, en un sistema NUMA, no todos los procesos tienen el mismo acceso a los bloques. Los procesos en el mismo nodo de NUMA como el cerrojo tienen pocas ventajas al obtener el cerrojo. Los procesos en nodos remotos de NUMA experimentan bloqueos y disminución en el rendimiento

Para solucionarlo, Red Hat Enterprise Linux implementó los *cerrojos en bucle de tiquetes*. Esta funcionalidad adiciona un mecanismo de cola de reserva para el cerrojo que permite a *todos* los procesos tomar un cerrojo en el orden que lo solicitaron. De esta manera se eliminan problemas de tiempo y ventajas no equitativas en solicitudes de cerrojo.

Aunque los cerrojos en bucle de tiquetes tienen más sobrecarga que un cerrojo en bucle común, escalan mejor y proporcionan mejor rendimiento en sistemas NUMA.

2.3. ESTRUCTURA DE LISTA DINÁMICA

El sistema operativo requiere un conjunto de información en cada procesador en el sistema. En Red Hat Enterprise Linux 5, esta información se asignaba a una matriz de tamaño fijo en memoria. Este método era rápido y directo para sistemas que contenían pocos procesadores.

Sin embargo, cuando el número de procesadores para un sistema crece, este método produce sobrecarga significativa. Puesto que el tamaño de matriz de memoria es fijo, el recurso compartido puede convertirse en un cuello de botella cuando más procesadores intentan accederlo al mismo tiempo.

Con el fin de solucionar este problema, Red Hat Enterprise Linux 6 usa una *estructura de lista dinámica* para información de procesador. Esta estructura permite a la matriz utilizada para información de procesador ser asignado de forma dinámica: si existen únicamente ocho procesadores en el sistema, entonces solo se crearán ocho entradas en la lista. Si hay 2.048 procesadores, entonces también se crearán 2.048 entradas.

Una estructura de lista dinámica permite un cerramiento más específico. Por ejemplo, si se necesita actualizar al mismo tiempo información para procesadores 6, 72, 183, 657, 931 y 1546, puede realizarse con un paralelismo mayor. Situaciones como estas son más frecuentes en sistemas grandes de alto rendimiento que en sistemas pequeños.

2.4. KERNEL SIN INTERVALO

En versiones anteriores de Red Hat Enterprise Linux, el kernel utilizaba un mecanismo basado en temporizador que producía continuamente una interrupción del sistema. Durante cada interrupción, el sistema *sondeaba*; es decir, verificaba si había que realizar algún trabajo.

Según el parámetro, este interrupción del sistema o *intervalo de temporizador* puede presentarse varios cientos o miles de veces por segundo. Esto sucede cada segundo, independiente de la carga de trabajo del sistema. En un sistema liviano, impactará el *consumo de energía* al evitar que el procesador use efectivamente los estados de dormido. Este sistema usa menos energía cuando está en el estado dormido.

La forma más eficiente de energía para que un sistema opere es trabajar lo más rápido posible, ir al

estado de 'dormido' más profundo. Para implementarlo, Red Hat Enterprise Linux 6 usa el *kernel sin intervalo*. Con este kernel, el temporizador de interrupciones ha sido retirado del bucle inactivo, transformando a Red Hat Enterprise Linux 6 en un entorno completamente controlado por interrupciones

El kernel sin intervalo le permite al sistema ir a estados de 'dormido' más profundos durante los tiempos inactivos y responder rápidamente cuando haya trabajo que hacer.

Para obtener mayor información, consulte la *Guía de administración de energía*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

2.5. GRUPOS DE CONTROL

Red Hat Enterprise Linux proporciona muchas opciones útiles para ajustar el rendimiento. Los grandes sistemas, que escalan a cientos de procesadores, pueden ajustarse para entregar un gran rendimiento. Sin embargo, el ajuste de estos sistemas requiere una experiencia considerable y una carga de trabajo bien definida. Cuando los grandes sistemas eran costosos y escasos, se aceptaba darles un tratamiento especial. Ahora que estos sistemas son comunes, se requieren herramientas más efectivas.

En cosas más complicadas, se utilizan sistemas más poderosos para consolidación de servicios. Las cargas de trabajo, que podrían estar ejecutándose en cuatro a ochos servidores más antiguos, ahora se colocan dentro un servidor individual. Además, como se discutió anteriormente en Sección 1.2.1, “Computación paralela”, muchos sistemas de medio rango de hoy, contienen más núcleos que las máquinas de alto rendimiento de ayer.

Muchas aplicaciones modernas están diseñadas para procesamiento paralelo, mediante múltiples hilos o procesos para mejorar el rendimiento. Sin embargo, pocas aplicaciones pueden hacer uso efectivo de más de ocho hilos. Por lo tanto, múltiples aplicaciones deben ser instaladas en un sistema de CPU de 32 a una capacidad máxima.

Considere la situación: los sistemas actuales pequeños y baratos están ahora en paridad con el rendimiento de las máquinas de ayer. Las máquinas de alto rendimiento otorgan a los arquitectos de sistemas la capacidad de consolidar más servicios a un número menor de máquinas.

Sin embargo, algunos recursos (tales como E/S y comunicaciones de redes) son compartidos y no crecen tan rápido como el conteo de CPU. Como tal, un sistema que alberga varias aplicaciones puede experimentar rendimiento total degradado cuando una aplicación acapara demasiado de un solo recurso.

Para solucionarlo, Red Hat Enterprise Linux 6 ahora soporta *grupos de control* (cgroups). Los cgroups le permiten a los administradores asignar recursos para tareas específicas cuando se necesiten. Esto significa, por ejemplo, que pueden asignar el 80% de cuatro CPU, 60GB de memoria y 40% de E/S de disco a una aplicación de base de datos. Una aplicación que se ejecute en el mismo sistema puede recibir dos CPU, 2GB de memoria y 50% del ancho de banda de red.

Como resultado, ambas bases de datos y aplicaciones de red entregan buen rendimiento, ya que el sistema evita que ambas consuman de forma excesiva los recursos del sistema. Además, varios aspectos de cgroups son de *auto-ajuste*, lo que permite al sistema responder según los cambios en carga de trabajo.

Un Cgroup tiene dos componentes importantes:

- Una lista de tareas asignadas al Cgroup
- Recursos asignados a aquellas tareas

Tareas asignadas al Cgroup se ejecutan *dentro* del Cgroup. Cualquier tarea hija que generen también se ejecuta dentor del Cgroup. Esto permite a un administrados manejar una aplicación entera como una unidad. Una administrador también puede configurar asignaciones para los siguientes recursos:

- CPUsets
- Memoria
- E/S
- Redes (ancho de banda)

Dentro de CPU, los cgroups permiten al los administradores configurar la cantidad de CPU, la afinidad para CPU o nodos específicos [3], y la cantidad de tiempo de CPU utilizada por una serie de tareas. El uso de cgroups para configurar CPUsets es vital para garantizar un buen rendimiento total, evitando así que una aplicación consuma recursos excesivos al costo de otras tareas, mientras que se garantiza de forma simultánea que a la aplicación no le falte tiempo de CPU.

El ancho de banda de E/S y de red se manejan mediante otros controladores de recursos. Una vez más, los controladores recursos le permiten determinar cuánto ancho de banda pueden consumir las tareas y garantizar que las tareas en un cgroup ni consuman recursos excesivos ni carezcan de ellos.

Los cgroups le permiten al administrador definir y asignar a un alto nivel, los recursos de sistemas que necesitan (y consumirán) varias aplicaciones. El sistema maneja de forma automática y equilibra las aplicaciones, entregando de ese modo un buen rendimiento predecible de todo el sistema.

Para obtener mayor información sobre cómo usar grupos de control, consulte la *Guía de administración de recursos*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

2.6. MEJORAS DE ALMACENAJE Y SISTEMAS DE ARCHIVOS

Red Hat Enterprise Linux 6 también introduce varias mejoras de almacenaje y administración de sistemas de archivos. Dos de los avances más importantes en esta versión son el soporte para ext4 y XFS. Para obtener mayor información sobre mejoras de rendimiento relacionadas con almacenaje y sistemas de archivos, consulte el Capítulo 7, *Sistemas de archivos*.

Ext4

Ext4 es el sistema de archivos predeterminado de Red Hat Enterprise Linux 6. Es la versión de la cuarta generación de la familia del sistema de archivos EXT, que soporta en teoría un tamaño de sistema de archivos máximo de 1 exabyte, y un archivo único máximo de 16TB. Red Hat Enterprise Linux 6 soporta un tamaño de sistema máximo de 16TB, y un archivo único de 16TB. Además de ofrecer una mayor capacidad, ext4 también incluye nuevas funcionalidades, tales como:

- Metadatos basados en extensión
- Asignación demorada
- Suma de verificación en el diario

Para obtener mayor información sobre el sistema de archivos ext4, consulte la Sección 7.3.1, “El sistema de archivos Ext4”.

XFS

XFS es un sistema de archivos de diario robusto y maduro de 64 bits que soporta grandes archivos y sistemas de archivos en un solo host. Este sistema de archivos fue desarrollado inicialmente por SGI, y

tiene una larga historia de ejecución en servidores extremadamente grandes y matrices de almacenamiento. Las funcionalidades XFS incluyen:

- Asignación demorada
- Inodos asignados de forma dinámica
- Indexación de Árbol-B para escalabilidad de administración de espacio libre.
- Desfragmentación en línea y crecimiento de sistema de archivos
- Metadatos sofisticados de algoritmos de lectura anticipada

Mientras que XFS escala a exabytes, el tamaño máximo del sistema de archivos XFS soportado por Red Hat es 100TB. Para obtener mayor información sobre XFS, consulte la Sección 7.3.2, “Sistema de archivos XFS”.

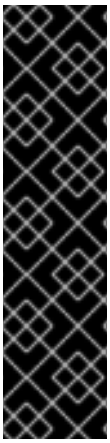
Grandes unidades de arranque

El BIOS tradicional soporta un disco máximo de 2.2TB. Los sistemas de Red Hat Enterprise Linux 6 que utilizan BIOS pueden soportar discos de más de 2.2TB mediante una nueva estructura de disco llamada *Tabla de particiones global* (GPT). GPT puede utilizarse únicamente para discos de datos; no se puede utilizar para unidades de arranque con BIOS; por lo tanto, las unidades de arranque solo pueden tener un tamaño máximo de 2.2TB. El BIOS fue creado en un principio para los PC de IBM; aunque el BIOS evolucionó considerablemente para adaptarse al hardware moderno, la *Interfaz de Firmware Extensible Unificada* (UEFI) fue diseñada para soportar el nuevo hardware emergente.

Red Hat Enterprise Linux 6 también soporta la UEFI, la cual puede utilizarse para remplazar el BIOS (aún con soporte). Los sistemas con UEFI que ejecutan Red Hat Enterprise Linux 6 permiten el uso de particiones GPT y 2.2TB (y mayores) tanto para la partición de arranque como para la partición de datos.

IMPORTANTE

Red Hat Enterprise Linux 6 no soporta UEFI para sistemas x86 de 32 bits.



IMPORTANTE

Observe que las configuraciones de arranque de UEFI y BIOS difieren significativamente una de la otra. Por lo tanto, el sistema instalado debe arrancar mediante el mismo Firmware utilizado durante la instalación. No se puede instalar el sistema operativo en un sistema que utilice BIOS y luego arrancar esta instalación en un sistema que utilice UEFI.

Red Hat Enterprise Linux 6 soporta la versión 2.2 de la especificación UEFI. El hardware que soporta la versión 2.3 o posterior de la especificación UEFI debe operar con Red Hat Enterprise Linux 6, pero la funcionalidad adicional definida por estas especificaciones no estarán disponibles. Las especificaciones están disponibles en http://www.uefi.org/specs/agreement/.

[3] Un nodo suele definirse como una serie de CPU o núcleos dentro de un socket.

CAPÍTULO 3. MONITORIZACIÓN Y ANÁLISIS DE RENDIMIENTO DE SISTEMAS

Este capítulo introduce brevemente las herramientas que pueden ser utilizadas para monitorizar y analizar el rendimiento de sistemas y aplicaciones, y señala las situaciones en las cuales cada herramienta es más útil. Los datos recogidos por cada herramienta pueden revelar cuellos de botella u otros problemas del sistema que contribuyen a un rendimiento menos óptimo.

3.1. EL SISTEMA DE ARCHIVOS 'PROC'

El 'sistema de archivos' **`proc`** es un directorio que contiene una jerarquía de archivos que representan el estado actual del kernel de Linux.

El directorio **`proc`** también contiene información sobre hardware del sistema y cualquier proceso que esté actualmente en ejecución. La mayoría de estos archivos son solo de lectura, pero algunos archivos (principalmente los que están en **`/proc/sys`**) pueden ser manipulados por usuarios y las aplicaciones para comunicar los cambios al kernel.

Para obtener mayor información sobre cómo ver y modificar archivos en el directorio **`proc`**, consulte la *Guía de implementación*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.2. MONITORES DE SISTEMA KDE Y GNOME

Los entornos de escritorio de KDE y GNOME tienen herramientas gráficas que lo ayudan en la monitorización y modificación de la conducta de nuestro sistema.

Monitor del sistema GNOME

El **Monitor del sistema GNOME** muestra información básica y le permite monitorizar sus procesos del sistema, recursos y uso del sistema de archivos. Ábralo con el comando **`gnome-system-monitor`** en la **Terminal**, o haga clic en el menú de **Aplicaciones** y seleccione **Herramientas del sistema** > **Monitor del sistema**.

El **Monitor del sistema de GNOME** tiene cuatro pestañas:

Sistema

Muestra la información básica sobre el hardware y software del computador

Procesos

Muestra los procesos activos y las relaciones entre los procesos, y la información detallada sobre cada uno de ellos. También le permite filtrar los procesos y realizar algunas acciones en dichos procesos (iniciar, detener, matar, cambiar prioridad, etc)

Recursos

Muestra el uso del tiempo de la CPU actual, la memoria y el espacio de intercambio y el uso de la red.

Sistema de archivos

Lista todos los sistemas de archivos montados junto con información básica de cada uno, tal como el tipo de sistema de archivos, punto de montaje y uso de memoria.

Para obtener mayor información sobre el **Monitor del sistema de GNOME**, consulte el menú de **Ayuda** en la aplicación o la *Guía de implementación*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Guarda del sistema de KDE

El **Guardián del sistema de KDE** le permite monitorizar la carga del sistema actual y los procesos que están en ejecución. También le permite realizar acciones en procesos. Ábralo mediante el comando **`ksysguard`** en la **Terminal**, o haga clic en el **Lanzador de aplicaciones Kickoff** y seleccione **Aplicaciones** > **Sistema** > **Monitor del sistema**.

Hay dos pestañas para el **Guardián del sistema de KDE**:

Tabla de procesos

Muestra una lista de los procesos que se ejecutan, de forma predeterminad, en orden alfabético. También puede clasificar los procesos por un número de propiedades, incluidas el uso de CPU, el uso de memoria física y compartida, el propietario y prioridad. Además, puede filtrar los resultados visibles, buscar procesos específicos o realizar algunas acciones en un proceso.

Carga del sistema

Muestra gráficas del historial de uso de CPU, espacio de memoria e intercambio y uso de red. Vaya sobre las gráficas para obtener un análisis detallado y claves de gráfico.

Para obtener mayor información sobre el **Guardián del sistema de KDE**, consulte el menú de **Ayuda** en la aplicación.

3.3. HERRAMIENTAS DE MONITORIZACIÓN DE LÍNEA DE COMANDOS

Aparte de las herramientas gráficas de monitorización, Red Hat Enterprise Linux proporciona varias herramientas que pueden ser utilizadas para monitorizar un sistema desde la línea de comandos. La ventaja de estas herramientas es que pueden ser utilizadas fuera del nivel de ejecución 5. Esta sección describe brevemente cada herramienta y sugiere los propósitos para los cuales la herramienta es más apta.

`top`

La herramienta **top** proporciona una vista dinámica en tiempo real de los procesos en un sistema en ejecución. Puede mostrar una variedad de información que incluye un resumen del sistema y las tareas que son administradas en el momento por el kernel de Linux. También tiene una habilidad limitada para manipular procesos. Tanto la operación como la información que presenta se pueden configurar muy bien y cualquier detalle de configuración puede hacerse persistente a través los reinicios.

Los procesos se presentan ordenados de forma predeterminada por porcentaje de uso de CPU, proporcionando así, una vista general de los procesos que están consumiendo la mayoría de los recursos.

Para obtener mayor información sobre **top**, consulte la página de manual: **`man top`**.

`ps`

La herramienta **ps** toma una instantánea de un grupo selecto de procesos activos. Este grupo se limita, de forma predeterminada, a los procesos pertenecientes al usuario actual y que están asociados a la misma terminal.

Puede proporcionar información más detallada sobre procesos que **top**, pero no es dinámica.

Para obtener información sobre el uso de **ps**, consulte la página de manual: **`man ps`**.

`vmstat`

vmstat (Estadísticas de memoria virtual) produce reportes instantáneos sobre los procesos de sistema, memoria, paginación, E/S de bloques, interrupciones y actividad de CPU.

Aunque no es dinámica como **top**, puede especificar un intervalo de muestra, el cual le permite observar la actividad del sistema en un tiempo casi real.

Para obtener información detallada sobre el uso de **vmstat**, consulte la página de manual: **`man vmstat`**.

`sar`

El **SAR** (Reportero de actividad del sistema) recolecta y reporta información sobre la actividad del sistema hasta el momento de hoy. La salida predeterminada cubre el uso de CPU de hoy en intervalos de 10 minutos desde el comienzo del día.

```
12:00:01 AM     CPU     %user     %nice   %system   %iowait    %steal
%idle
12:10:01 AM     all      0.10      0.00      0.15      2.96      0.00
96.79
12:20:01 AM     all      0.09      0.00      0.13      3.16      0.00
96.61
12:30:01 AM     all      0.09      0.00      0.14      2.11      0.00
97.66
...
```

Esta herramienta es una alternativa útil para intentar crear reportes periódicos sobre la actividad del sistema mediante **top** o herramientas similares.

Para obtener información detallada sobre el uso de**sar**, consulte la página de manual: **`man sar`**.

3.4. TUNED Y KTUNE

Tuned es un daemon que monitoriza y colecciona datos sobre el uso de varios componentes del sistema y utiliza esa información para ajustar de forma dinámica los parámetros del sistema como se requiere. Puede reaccionar a cambios en el uso de la CPU y red y ajustar parámetros para mejorar el rendimiento en dispositivos activos o reducir consumo de energía en dispositivos inactivos.

ktune se asocia con la herramienta **tuned-adm** para proporcionar un número de perfiles de ajuste que se preconfiguran para mejorar el rendimiento y reducir el consumo de energía en un número específico de casos. Modifique o cree nuevos perfiles para crear soluciones de rendimiento ajustadas a su entorno.

Los perfile provistos como parte de **tuned-adm** incluyen:

`default`

El perfil predeterminado de ahorro de energía. Es el perfil de ahorro de energía más básico. Habilita únicamente los complementos de disco y CPU. Observe que no es lo mismo que apagar **tuned-adm**, donde tanto **tuned**y **ktune** están inactivos.

`latency-performance`

Un perfil de servidores para ajuste de rendimiento de latencia típico. Desactiva los mecanismos de ahorro de energía de **tuned** y **ktune**. El modo de **`cpuspeed`** cambia a **`rendimiento`**. El elevador de

E/S cambia a **fecha límite** para cada dispositivo. Para calidad de administración de energía de servicio, el valor de requerimiento de **cpu_dma_latency** se registra con un valor de **0**.

throughput-performance

Un perfil de servidor para ajuste de rendimiento típico. Este perfil se recomienda si el sistema no tiene almacenamiento de clase empresarial. Es igual a **latency-performance**, excepto:

- ***kernel.sched_min_granularity_ns*** (granularidad de preferencia mínima del programador) se establece a **10** milisegundos,
- ***kernel.sched_wakeup_granularity_ns*** (granularidad de despertador del programador) se establece a **15** milisegundos,
- ***vm.dirty_ratio*** (relación sucia de máquina virtual) se establece a 40%, y
- las páginas gigantes transparentes se activan.

enterprise-storage

Este perfil se recomienda para configuraciones de servidor de tamaño empresarial con almacenamiento de clase empresarial, que incluye protección de cache y administración de controlador de batería de respaldo de cache en disco. Es similar al perfil de **throughput-performance** con una sola adición: los sistemas de archivos se remontan con **barrier=0**.

virtual-guest

Este perfil se recomienda para configuraciones de servidor de tamaño empresarial con almacenamiento de clase empresarial, que incluye protección de cache y administración de controlador de batería de respaldo de cache en disco. Es igual que el perfil de **throughput-performance**, excepto que:

- el valor ***readahead*** se establece a **4x**, y
- los sistemas de archivos no root/boot se remontan con **barrier=0**.

virtual-host

Basándose en el perfil de almacenaje de ***enterprise-storage***, ***virtual-host*** también decrece en swappiness de memoria virtual y habilita más retro-escritura agresiva de páginas sucias. Este perfil está disponible en Red Hat Enterprise Linux 6.3 y posterior, y es el perfil recomendado para hosts de virtualización, incluidos los hosts de KVM y de Red Hat Enterprise Virtualization.

Consulte la *Guía de administración de energía* de Red Hat Enterprise Linux 6, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, para obtener mayor información sobre **tuned** y **ktune**.

3.5. PERFILADORES DE APLICACIONES

Perfilado es el proceso de reunir información sobre una conducta de un programa durante su ejecución. Una aplicación es perfilada para determinar las áreas de un programa que pueden ser optimizadas para aumentar la velocidad completa del programa, reducir su uso de memoria, etc. Las herramientas de perfilado de aplicaciones ayudan a simplificar este proceso.

Existen tres herramientas de perfilado para usar con Red Hat Enterprise Linux 6: **SystemTap**, **OProfile**

y **Valgrind**. La descripción de estas herramientas no es uno de los objetivos de esta guía; sin embargo, esta sección proporciona enlaces si desea obtener mayor información y una visión general de las tareas para las cuales es apropiado cada perfilador.

3.5.1. SystemTap

SystemTap es una herramienta de trazado y sondeos que permite a los usuarios monitorizar y analizar en detalle las actividades del sistema operativo (en particular las actividades de kernel). Proporciona información similar a la salida de herramientas como **netstat**, **top**, **ps** y **iostat**, pero incluye filtraje adicional y opciones de análisis para la información recolectada.

SystemTap proporciona un análisis más preciso y profundo de las actividades del sistema y la conducta de aplicaciones para permitirle determinar con precisión los cuellos de botellas de las aplicaciones.

La función del complemento de Callgraph para Eclipse usa SystemTap como un segundo plano, el cual permite monitorizar a profundidad el estatus de un programa, incluidas las llamadas de función, retornos, tiempos y variables de espacio de usuario, y visualiza la información para una fácil optimización.

Para obtener mayor información sobre SystemTap, consulte la *Guía para principiantes de SystemTap*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.5.2. OProfile

OProfile (**oprofile**) es una herramienta de monitorización de rendimiento de todo el sistema. Utiliza hardware dedicado de monitorización de rendimiento de procesos para recuperar información sobre el kernel y los ejecutables del sistema, como por ejemplo cuando se refiere a la memoria, a la cantidad de solicitudes de cache L2 y al número de interrupciones de hardware recibidas. También puede ser utilizado para determinar el uso del procesador y las aplicaciones y servicios más utilizadas.

OProfile también puede utilizarse con Eclipse a través del complemento de Eclipse OProfile. Este complemento permite determinar fácilmente la mayoría de áreas que consumen tiempo de su código y realizan todas las funciones de línea de comandos de OProfile con una visualización copiosa de resultados.

Sin embargo, los usuarios deben tener en cuenta varias limitaciones de OProfile:

- Las muestras de monitorización de rendimiento no pueden ser precisas, puesto que el procesador puede ejecutar instrucciones que no funcionan o registrar una muestra de una instrucción cercana, en lugar de la instrucción que produjo la interrupción.
- Puesto que OProfile es un sistema amplio y espera que los procesos inicien y se detengan varias veces, se permite que las muestras acumulen múltiples ejecuciones. Es decir, que puede es posible que tenga que limpiar datos de muestras de ejecuciones anteriores.
- Se enfoca en la identificación de problemas con procesos de CPU limitada y por lo tanto, no identifica los procesos que están durmiendo mientras esperan cerrojos para otros eventos.

Para obtener mayor información sobre el uso de OProfile, consulte la *Guía de implementación*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, o la documentación **oprofile** en su sistema, localizado en **`/usr/share/doc/oprofile-<versión>`**.

3.5.3. Valgrind

Valgrind ofrece una serie de herramientas que colaboran con el rendimiento y exactitud de nuestras aplicaciones. Estas herramientas pueden detectar errores de memoria y de hilos relacionados, y

sobrecargas de montículo, pila y matrices, que le permiten localizar y corregir fácilmente errores en su código de aplicación. Pueden también perfilar la memoria cache, el montículo y predicción de saltos para identificar los factores que incrementan la velocidad de aplicaciones y minimizan el uso de memoria de aplicaciones.

Valgrind analiza su aplicación al ejecutarla en una CPU sintética e instrumentando el código de aplicación existente mientras se ejecuta. Imprime luego "commentary" identificando de forma clara el proceso implicado en la ejecución de la aplicación para un descriptor de archivos de usuario especificado. El nivel de instrumentación varía según la herramienta Valgrind en uso y sus parámetros, pero es importante observar que al ejecutar el código instrumentado puede tardar de 4 a 50 veces más que la ejecución normal.

Valgrind puede servir para usar su aplicación tal como está, sin recompilar. Sin embargo, puesto que Valgrind usa información de depuración para señalar los problemas en su código, si sus bibliotecas de aplicaciones y soporte no estuvieran compiladas con información de depuración habilitada, es sumamente recomendable incluir dicha información.

A partir de Red Hat Enterprise Linux 6.4, Valgrind se integra con gdb (Proyecto de depuración GNU) para mejorar la eficiencia de depuración.

Par obtener mayor información sobre Valgrind, consulte la *Guía del desarrollador*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, o al usar el comando **man valgrind** cuando sea instalado el paquete de valgrind. También encontrará documentación en:

- **/usr/share/doc/valgrind-<versión>/valgrind_manual.pdf**
- **/usr/share/doc/valgrind-<version>/html/index.html**

Para mayor información sobre el uso de Valgrind para perfilar la memoria del sistema, consulte la Sección 5.3, “Cómo utilizar Valgrind para perfilar el uso de memoria”.

3.5.4. Perf

La herramienta **perf** proporciona un número de contadores de rendimiento útiles que permiten al usuario evaluar el impacto de otros comandos en su sistema:

perf stat

Este comando proporciona las estadísticas generales para eventos de rendimiento, que incluyen instrucciones ejecutadas y ciclos de reloj consumidos. Utilice los indicadores para reunir estadísticas sobre eventos diferentes a los eventos de medidas predeterminados. A partir de Red Hat Enterprise Linux 6.4, es posible usar **perf stat** para filtrar monitorización basada en uno o más grupos de control (cgroups). Para obtener mayor información, consulte la página de manual: **man perf-stat**.

perf record

Este comando registra datos de rendimiento en un archivo que puede ser analizado más adelante mediante **perf report**. Para obtener mayor información, consulte la página de manual: **man perf-record**.

perf report

Este comando lee los datos de rendimiento de un archivo y analiza los datos registrados. Para obtener mayor información, consulte la página de manual: **man perf-report**.

perf list

Este comando lista los eventos disponibles en una determinada máquina. Dichos eventos varían según el hardware de monitorización de rendimiento y el software de configuración del sistema. Para obtener mayor informacón, consulte la página de manual: **`man perf-list`**.

`perf top`

Este comando realiza una función similar a la herramienta **top**. Genera y despliega un perfil de contador de rendimiento en tiempo real. Para obtener mayor información, consulte la página de manual:**`man perf-top`**.

Para obtener mayor información sobre **perf**, consulte la *Guía de desarrollador* de Red Hat Enterprise Linux, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.6. RED HAT ENTERPRISE MRG

El componente de tiempo real de MRG de Red Hat enterprise incluye a **Tuna**, una herramienta que permite a los usuarios ajustar los valores de ajustables que su sistema y ver los resultados de dichos cambios. Aunque fue desarrollado para usar con el componente de Realtime, puede utilizarse para ajustar los sistemas de i Red Hat Enterprise Linux estándar.

Con Tuna, puede ajustar o desactivar la actividad innecesaria del sistema, incluidos:

- Los parámetros de BIOS relacionados con la administración de energía, detección de errores e interrupciones de administración del sistema;
- los parámetros de redes, tales como interrupción de coalescencia, y el uso de TCP;
- Actividad de diario en sistemas de archivos de diario
- ingreso a sistema;
- Si las interrupciones o procesos de usuario son manejadas por una CPU o un rango de CPU específicas;
- Si el espacio swap es utilizado; y
- cómo tratar excepciones de falta de memoria.

Para obtener mayor información conceptual sobre el ajuste de Red Hat Enterprise MRG con la interfaz Tuna, consulte el capítulo de Ajuste general del sistema de la *Guía de ajuste de tiempo real 2*. Para obtener información sobre la interfaz Tuna, consulte la *Guía del usuario de Tuna*. Ambas guías están disponibles en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_MRG/.

CAPÍTULO 4. CPU

El término CPU, el cual significa *Unidad de procesamiento central*, es un nombre equivocado para la mayoría de los sistemas, puesto que *central* implica *única*, mientras que la mayoría de los sistemas modernos tienen más de una unidad de procesamiento o núcleo. Físicamente, las CPU vienen en un paquete unido a la placa madre en un *socket*. Cada socket en la placa madre tiene varias conexiones: a otros sockets de CPU, controladores de memoria, controladores de interrupción, y otros dispositivos periféricos. Un socket para el sistema operativo es una agrupación de varias CPU y recursos asociados. Este concepto es central a la mayoría de nuestras discusiones sobre ajuste de CPU.

Red Hat Enterprise Linux mantiene una riqueza de estadística sobre eventos de CPU de sistemas; estas estadísticas sirven en la planeación de una estrategia de ajuste para mejorar el rendimiento de CPU. Sección 4.1.2, “Ajuste de rendimiento de la CPU” discute algunas de las estadísticas útiles, dónde hallarlas, cómo analizarlas para ajuste de rendimiento.

TOPOLOGÍA

Los computadores antes tenían relativamente pocas CPU por sistema que permitían una arquitectura conocida como *Multiprocesador simétrico* (SMP). Esto significaba que cada CPU en el sistema tenía acceso similar (o simétrico) a la memoria disponible. En años recientes, el conteo por socket de CPU ha aumentado al punto de que el hecho de intentar dar acceso simétrico a todos los RAM en el sistema se ha convertido en algo muy costoso. La mayoría de sistemas de conteo de CPU de alto costo en estos días tienen una arquitectura conocida como *Acceso de memoria no uniforme* (NUMA) en lugar de SMP.

Los procesadores AMD han tenido este tipo de arquitectura por algún tiempo con sus interconexiones *Hyper Transport* (HT), mientras que Intel ha comenzado a implementar NUMA en sus diseños de *Quick Path Interconnect* (QPI). NUMA y SMP se ajustan de forma diferente, ya que usted debe justificar la *topología* del sistema al asignar recursos para una aplicación.

HILOS

Dentro del sistema operativo de Linux, la unidad de ejecución se conoce como *hilo*. Los hilos tienen un contexto de registro, una pila, y un segmento de código ejecutable que ejecutan en una CPU. Es trabajo del sistema operativo (SO) programar esos hilos en las CPU disponibles.

El Sistema operativo maximiza el uso de CPU al balancear los hilos a través de núcleos disponibles. Puesto que el SO se interesa principalmente en mantener ocupadas las CPU, no siempre tomará decisiones óptimas con respecto al rendimiento de aplicaciones. Si traslada un hilo de aplicación a una CPU en otro socket empeorará el rendimiento más que si espera a que la CPU actual esté disponible, puesto que las operaciones de memoria pueden desacelerarse a través de sockets. Para obtener aplicaciones de alto rendimiento, es mejor que el diseñador determine dónde se deben colocar los hilos. La Sección 4.2, “Programación de CPU” discute la mejor manera de asignar las CPU y memoria para ejecutar hilos de aplicaciones.

INTERRUPCIONES

Uno de los eventos menos obvios (pero, importante) que puede impactar el rendimiento de aplicaciones son las *interrupciones* (también conocidas como las IRQ en Linux). Estos eventos son manejados por el sistema operativo y utilizados por periféricos para señalar la llegada de datos o la culminación de una operación, tal como una escritura de red o un evento de temporizador.

La forma en que el SO o la CPU que ejecuta el código de aplicaciones maneja una interrupción, no afecta la funcionalidad de la aplicación. Sin embargo, puede impactar el rendimiento de la aplicación. Este capítulo discute algunos consejos sobre prevención de interrupciones de rendimiento de aplicaciones que están impactando de una forma adversa.

4.1. TOPOLOGÍA DE CPU

4.1.1. Topología de CPU y Numa

Los primeros procesadores de computadores eran *uniprocesadores*, es decir que el sistema tenía una sola CPU. La ilusión de procesos en ejecución en paralelo solamente era realizada por el sistema operativo al pasar rápidamente la única CPU de un proceso de ejecución a otro. En la búsqueda por aumentar el rendimiento del sistema, los diseñadores notaron que el aumento de la tasa de reloj para ejecutar instrucciones de una forma más rápida solamente funcionaba hasta cierto punto (por lo general limitaciones en la creación de una onda de reloj estable con la tecnología actual). Para obtener un mayor rendimiento de todo el sistema, los diseñadores le añadieron otra CPU, lo cual permitió dos corrientes de ejecución paralelas. Esta tendencia de añadir procesadores ha continuado con el tiempo.

La mayoría de los primeros sistemas de multiprocesadores fueron diseñados para que cada CPU tuviera la misma ruta para cada ubicación de memoria (por lo general un bus paralelo). Esto permitía a cada CPU acceder a cualquier sitio de memoria en la misma cantidad de tiempo que cualquier otra CPU en el sistema. Este tipo de arquitectura se conoce como Sistema de multiprocesador simétrico (SMP). El SMP es apropiado para pocas CPU, pero si el número de CPU está por encima de cierto punto (de 8 a 16), los trazados paralelos requeridos para permitir igual acceso a memoria, ocuparán demasiado espacio en la placa, y dejarán menos espacio para los periféricos.

Dos nuevos conceptos combinados para permitir un número superior de CPU en un sistema:

1. Buses seriales

2. Topologías de NUMA

Un bus serial es una ruta de comunicación de un solo cable con una muy alta tasa de reloj, el cual transfiere datos como ráfagas de paquetes. Los diseñadores de hardware comenzaron a usar los buses de seriales como interconexiones de alta velocidad entre varias CPU y entre varias CPU y controladores de memoria y otros periféricos. Es decir que en lugar de requerir entre 32 y 64 trazos en la placa desde *cada* CPU al subsistema de memoria, había ahora *un* trazo, lo cual reduce de manera substancial la cantidad de espacio requerido en la placa.

Al mismo tiempo, los diseñadores de hardware empaquetaban más transistores en el mismo espacio para reducir tamaños de circuitos. En lugar de colocar CPU individuales directamente en la placa principal, comenzaban a empacarlas dentro del paquete del procesador como procesadores multinúcleos. Entonces, en lugar de intentar proporcionar igual acceso a memoria desde cada paquete de procesador, los diseñadores recurrieron a una estrategia de Acceso de memoria no uniforme o NUMA, en la que cada combinación de paquete y socket tiene una o más áreas de memoria dedicadas para acceso de alta velocidad. Cada conector también está interconectado a otros sockets para acceso más lento a la otra memoria de los otros sockets.

A manera de ejemplo de NUMA, supongamos que tenemos una placa madre de dos sockets cada una poblada con un paquete quad-core. Es decir, que el número total de CPU en el sistema es ocho; cuatro en cada socket. Cada socket también tiene un banco de memoria conectado de cuatro GB de RAM, para una memoria de sistema total de ocho GB. Para este ejemplo, las CPU 0-3 están en el socket 0 y las CPU 4-7 están en socket 1. Cada socket en este ejemplo también corresponde a un nodo de NUMA.

Podría tomarse tres ciclos de reloj para que CPU 0 acceda a memoria desde banco 0: un ciclo para presentar la dirección al controlador de memoria, un ciclo para configurar acceso al sitio de memoria y un ciclo para leer o escribir al sitio. Sin embargo, podría tomarse seis ciclos de reloj para que CPU4 acceda a memoria desde el mismo sitio; debido a que está en un socket independiente, deberá ir a través de dos controladores de memoria en socket 0. Si la memoria no se impugna en dicha ubicación (es decir, si hay más de una CPU intentando acceder de forma simultánea al mismo sitio), los controladores de memoria necesitarán arbitrar y poner en serie el acceso a la memoria, a fin de que el acceso de memoria se prolongue. Si adiciona consistencia de cache (garantizar que las cache de CPU local contengan los mismos datos para el mismo sitio de memoria), complica el proceso más adelante.

Los procesadores de alta gama más recientes de Intel (Xeon) y AMD (Opteron) tienen topologías de NUMA. Los procesadores AMD usan una interconexión conocida como HyperTransport o HT, mientras que Intel usa una llamada QuickPath Interconnect u QPI. Las interconexiones difieren en la forma como se conectan físicamente a otras interconexiones, memoria, y dispositivos periféricos, pero en efecto son un interruptor que permite acceso transparente a un dispositivo conectado. En este caso, transparente se refiere a que no se requiere API de programación especial para usar la interconexión, no es una opción "sin costo" alguno.

Puesto que las arquitecturas del sistema son tan diversas, no es práctico caracterizar de forma específica la multa de rendimiento impuesta por acceder a memoria no-local. Decimos que cada *salto* o 'hop' a través de una interconexión impone al menos alguna multa de rendimiento constante por salto, por lo tanto al referirnos a ubicación de memoria que está a dos interconexiones de la CPU actual se impone por la menos *2N + tiempo de ciclo de memoria* unidades para acceder tiempo, donde N es la multa por salto.

Dada la multa de rendimiento, las aplicaciones sensibles a rendimiento deben evitar con regularidad el acceso a la memoria remota en un sistema de topología de NUMA. La aplicación debe configurarse para que permanezca en un nodo determinado y asigne memoria desde dicho nodo.

Para hacerlo, hay algunas cosas que las aplicaciones deben saber:

1. ¿Qué significa *topología* del sistema?
2. ¿En dónde se está ejecutando la aplicación?
3. ¿En dónde se encuentra el banco de memoria más cercano?

4.1.2. Ajuste de rendimiento de la CPU

Lea esta sección para entender cómo ajustar para obtener mejor rendimiento de CPU y para una introducción de varias herramientas que ayudan en el proceso.

NUMA se utilizaba en un principio, para conectar un procesador único a varios bancos de memoria. Debido que los fabricantes de CPU refinaron los procesos y encogieron los tamaños, los núcleos de CPU múltiples pudieron incluirse en un paquete. Dichos núcleos de CPU eran agrupados para que cada vez que se accediera a una memoria de banco local y cache, se pudieran compartir entre los núcleos; sin embargo, cada 'salto' a través de una núcleo , memoria y una cache, trae como consecuencia poco rendimiento.

El sistema de ejemplo en Figura 4.1, “El acceso de memoria remota y local en topología de NUMA” contiene dos nodos NUMA. Cada nodo tiene cuatro CPU, un banco de memoria y un controlador de memoria. Cualquier CPU en un nodo tiene acceso directo al banco de memoria en ese nodo. Al seguir las flechas en Nodo 1, los pasos son los siguientes:

1. Una CPU (0-3) presenta la dirección de memoria para el controlador local.
2. El controlador de memoria configura el acceso a la dirección de memoria.
3. La CPU realiza operaciones de lectura y escritura en esa dirección de memoria.

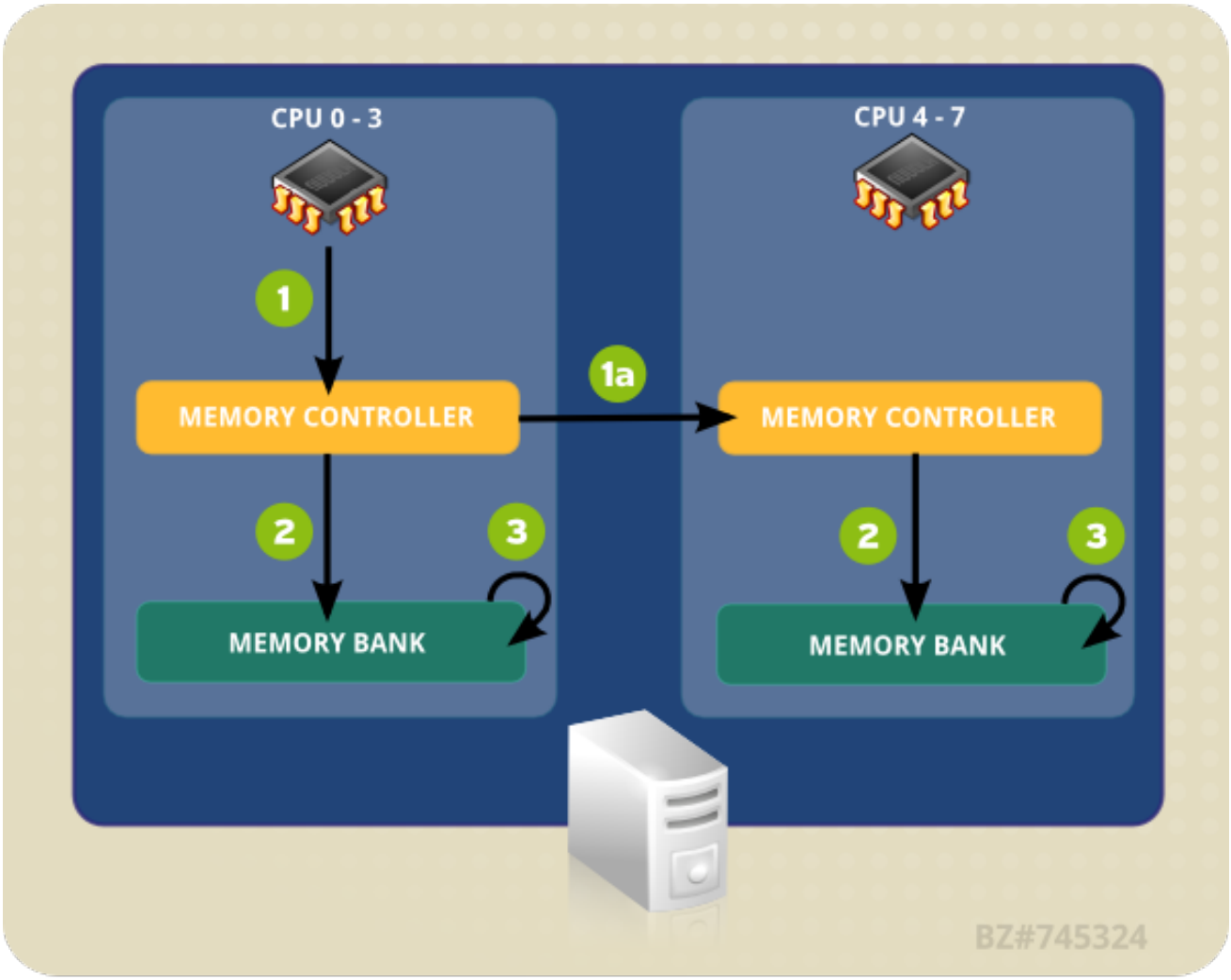


Figura 4.1. El acceso de memoria remota y local en topología de NUMA

Sin embargo, si una CPU en un nodo necesita código de acceso que reside en el banco de memoria de un NUMA diferente, la ruta que debe tomar será menos directa:

1. Una CPU (0-3) presenta la dirección de memoria remota para el controlador de memoria local.
    1. La solicitud de CPU para la dirección de memoria remota se pasa a un controlador de memoria remota, local al nodo que contiene la dirección de memoria.
2. El controlador de memoria remota configura el acceso a la dirección de memoria remota.
3. La CPU realiza operaciones de lectura y escritura en esa dirección de memoria remota.

Cada acción necesita pasar por controladores de memoria múltiple, por lo tanto el acceso puede tomar más de dos veces siempre y cuando intente acceder direcciones de memoria remota. El rendimiento primario tiene que ver con un sistema multinúcleos, para garantizar que la información viaje de forma tan eficiente como sea posible, a través de la ruta más rápida y corta.

Para configurar una aplicación para obtener un rendimiento de CPU óptimo, debe saber:

- la topología de los componentes del sistema (cómo se conectan sus componentes),
- el núcleo en el cual se ejecuta la aplicación, y

- la ubicación del banco de memoria más cercano.

Red Hat Enterprise Linux 6 se distribuye con un número de herramientas para ayudarlo a encontrar información y a ajustar su sistema. Las siguientes secciones ofrecen una visión general de las herramientas útiles para ajustar el rendimiento de CPU.

4.1.2.1. Configuración de afinidad de CPU contaskset

taskset recupera y establece la afinidad de CPU de un proceso en ejecución (por el ID de proceso). También puede servir para lanzar un proceso con una afinidad de CPU determinada, la cual vincula los procesos especificados a una CPU especificada o una serie de CPU. No obstante, **taskset** no garantiza la asignación de memoria local. Si requiere beneficios de rendimiento adicional de memoria local, recomendamos **numactl** en **taskset**; consulte la Sección 4.1.2.2, “Control de la política NUMA con **numactl**” para obtener mayor información.

La afinidad de CPU se representa como una máscara de bits. El orden más bajo de bits corresponde a la primera CPU lógica, y el más alto corresponde a la última CPU lógica. Estas máscaras suelen ser hexadecimales, por lo tanto **`0x00000001`** representa procesador 0, y **`0x00000003`** representa procesador 0 y 1.

Para establecer la afinidad de CPU de un proceso en ejecución, ejecute el siguiente comando, remplace *máscara* por la máscara del procesador o procesadores a la que desea vincular el proceso y *pid* por el ID del proceso cuya afinidad desea cambiar:

```
# taskset -p máscara pid
```

Para lanzar un proceso con una determinada afinidad, ejecute el siguiente comando, remplace *máscara* por la máscara del procesador o procesadores a la que desea vincular su proceso, y *programa* por el programa, opciones y argumentos del programa que desea ejecutar.

```
# taskset mask -- program
```

En lugar de especificar los procesadores como una máscara de bits, puede también usar la opción **`-c`** para proporcionar una lista delimitada por comas o un rango de procesadores, como:

```
# taskset -c 0,5,7-9 -- myprogram
```

Para mayor información sobre **taskset**, consulte la página de manual: **`man taskset`**.

4.1.2.2. Control de la política NUMA connumactl

`numactl` ejecuta procesos con programación especificada o política de ubicación de memoria. La política seleccionada se establece para este proceso y todos sus hijos. **`numactl`** también puede establecer política persistente para segmentos o archivos de memoria compartida y establece afinidad de CPU y afinidad de memoria de un proceso. Utiliza el sistema de archivos **`/sys`** para determinar la topología del sistema.

El sistema de archivos **`/sys`** contiene información sobre cómo se conectan las CPU, la memoria y los dispositivos periféricos mediante interconexiones NUMA. En particular, el directorio **`/sys/devices/system/cpu`** contiene información sobre cómo se conectan las CPU de un sistema a otro. El directorio **`/sys/devices/system/node`** contiene información sobre los nodos NUMA en el sistema, y las distancias relativas entre dichos nodos.

En un sistema NUMA, entre mayor sea la distancia entre un procesador y un banco de memoria, más

lento será el acceso del procesador a dicho banco de memoria. Las aplicaciones sensibles al rendimiento deben, por lo tanto, configurarse para que puedan asignar memoria desde el banco de memoria más cercano posible.

Las aplicaciones sensibles de rendimiento también deben configurarse para ejecutar un conjunto de núcleos, en particular, aplicaciones multihilos. Debido a que la memoria cache de primer nivel suele ser pequeña, si múltiples hilos se ejecutan en un núcleo, cada hilo en potencia desalojará datos en cache accedidos por un hilo previo. Cuando el sistema operativo intenta realizar multitareas entre estos hilos, y los hilos que continúan desalojando los datos en cache del otro, se consume un gran porcentaje de su tiempo de ejecución en el remplazo de línea de cache. Este problema se conoce como *hiperpaginación de cache*. Por lo tanto, se recomienda vincular una aplicación multihilos a un nodo, en lugar de vinculoarla a un núcleo individual, ya que esto permite a los hilos compartir líneas cache en múltiples niveles (primero-, segundo-, y último nivel de cache) y minimizar la necesidad para las operaciones de llenado de cache. Sin embargo, la vinculación de una aplicación a un núcleo individual puede funcionar si todos los hilos acceden a los mismos datos guardados en cache.

numactl le permite vincular una aplicación a un núcleo particular o nodo de NUMA y asignar la memoria asociada con un núcleo o conjunto de núcleos a una aplicación. Algunas opciones útiles proporcionadas por numactl son:

--show

Despliega lo parámetros de política NUMA del proceso actual. Este parámetro no requiere parámetros adicionales y puede ser utilizado como tal: **`numactl --show`**.

--hardware

Despliega un inventario de los nodos disponibles en el sistema

--membind

Solamente asigna memoria desde los nodos especificados. Cuando está en uso, la asignación de memoria fallará en estos nodos si es insuficiente. El uso para este parámetro es **`numactl --membind=`*`nodos programa`***, where *nodos* es la lista de nodos de la cual desea asignar memoria y *programa* es el programa cuyos requerimientos de memoria deberían asignarse desde ese nodo. Los números de nodos puede otorgarse como una lista delimitada por comas, un rango o una combinación de los dos. Para obtener mayor información sobre **numactl**, consulte la página de manual: **`man numactl`**.

--cpunodebind

Solamente ejecute un comando (y sus procesos hijos) en las CPU pertenecientes al nodo especificado. El uso para este parámetro es **`numactl --cpunodebind=`*`nodos programa`***, donde *nodos* es la lista de nodos a cuyas CPU será vinculado el programa especificado (*programa*). Los números de nodos pueden presentarse como una lista delimitada por comas, un rango o combinación de dos. Para mayor información sobre **numactl**, consulte la página de manual: **`man numactl`**.

--physcpubind

Solamente ejecute un comando (y sus procesos hijos) en las CPU especificadas. El uso para este parámetro es **`numactl --physcpubind=`*`cpu program`***, donde *cpu* es una lista delimitada por comas de los números de CPU como aparecen en los campos de **`/proc/cpuinfo`**, y *programa* es el programa que debe ejecutarse solo en esas CPU. Las CPU también pueden especificarse con relación a la **`cpuset`** actual. Para obtener mayor información sobre **numactl**, consulte la página de manual: **`man numactl`**.

--localalloc

Especifica que la memoria siempre debe asignarse en el nodo actual

--preferred

Donde sea posible, la memoria es asignada en el nodo especificado. Si la memoria no puede asignarse en el nodo especificado, se conmutará a otros nodos. Esta opción toma únicamente un nodo individual, así: **`numactl --preferred=`*nodo***. Para obtener mayor información sobre numactl, por favor consulte la página de manual: **`man numactl`**.

La biblioteca **libnuma** incluida en el paquete numactl, ofrece una interfaz de programación sencilla para la política de NUMA soportada por el kernel. Se utiliza para ajuste minucioso más que la herramienta **numactl**. Para mayor información, consulte la página de manual: **`man numa(3)`**.

4.1.3. numastat

IMPORTANTE

La herramienta **numastat** fue escrita en script de Perl por Andi Kleen y reescrita para Red Hat Enterprise Linux 6.4.

Mientras que el comando predeterminado (**`numastat`**, sin opciones ni parámetros) mantiene una compatibilidad estricta con la versión anterior a la herramienta, observe que las provisiones de iones o parámetros para este comando cambia de forma significativa tanto el contenido de salida como el formato.

numastat despliega estadísticas de memoria (tal como asignación de golpes y pérdidas) fpara procesos y sistema operativo en un NUMA por nodo. Ahora al ejecutar **`numastat`** se despliega de forma predeterminada el número de páginas de memoria ocupadas por las siguientes categorías de eventos para cada nodo.

El rendimiento óptimo de CPU se indica mediante los valores bajos de **`numa_miss`** y **`numa_foreign`**.

Esta versión actualizada de **numastat** también muestra si la memoria de proceso se extiende a través de un sistema o si se centraliza en nodos específicos mediante **numactl**.

La referencia cruzada de salida **numastat** con la salida por CPU de **top** para verificar que los hilos de procesos se ejecuten en los mismos nodos a los que se asigna la memoria.

Categorías de trazado predeterminado

numa_hit

El número de asignaciones intentadas en este nodo que no fueron exitosas.

numa_miss

El número de asignaciones designadas a otro nodo que fueron asignadas en este nodo debido a baja memoria en el nodo designado. Cada evento **`numa_miss`** tiene un evento correspondiente **`numa_foreign`** en otro nodo.

numa_foreign

El número de asignaciones que destinadas inicialmente para este nodo que fueron asignadas a otro nodo. Cada evento **numa_foreign** tiene un evento **numa_miss** correspondiente en otro nodo.

interleave_hit

El número de asignaciones de políticas de intercalación intentadas en este nodo que fueron exitosas.

local_node

El número de veces que un proceso en este nodo asignó memoria correctamente en este nodo.

other_node

El número de veces que un proceso en este nodo asigna memoria correctamente en este nodo.

Al suplir cualquiera de las siguientes opciones cambian las unidades desplegadas a MB de memoria (redondeando a dos decimales), y otras conductas **numastat** específicas como se describe a continuación.

-c

De forma horizontal condensa la tabla de información desplegada. Es útil en sistemas con un gran número de nodos NUMA, pero la anchura de columnas y el espacio entre columnas son, de alguna manera, predecibles. Cuando se utiliza esta opción, la cantidad de memoria se redondea al megabyte más cercano.

-m

Muestra la información de uso de memoria en todo el sistema en una base por nodo, similar a la información que se encuentra en **/proc/meminfo**.

-n

Muestra la misma información que la original de **numastat** (numa_hit, numa_miss, numa_foreign, interleave_hit, local_node, and other_node), con un formato actualizado, mediante MB como la unidad de medida

-p *patrón*

Muestra información de memoria por nodo para un patrón específico. Si el valor para el *patrón* consta de dígitos, **numastat** supone que es un identificador de procesos numérico. De lo contrario, **numastat** busca líneas de comando de procesos para el patrón especificado.

Se asume que los argumentos de línea de comandos ingresados después del valor de la opción **-p** sean patrones adicionales para ser filtrados. Los patrones adicionales expanden el filtro, en lugar de estrecharlo.

-s

Clasifica los datos desplegados en orden descendente para que los que consumen más memoria (según la columna **total**) estén en la lista de primeros.

También puede especificar un *nodo*, y la tabla será organizada según la columna de *nodo*. Al usar esta opción, el valor de *nodo* debe ir acompañado de la opción **-s** inmediatamente, como se muestra aquí:

```
numastat -s2
```

No incluya el espacio en blanco entre la opción y su valor.

-v

Muestra información más detallada. Es decir, información del proceso para múltiples procesos desplegará información detallada para cada proceso.

-V

Despliega información sobre la versión de **numastat**.

-z

Omite las filas y columnas de tabla con valores de cero únicamente. Observe que los valores cercanos a cero se redondean a cero para propósitos de despliegue y no serán omitidos de la salida desplegada.

4.1.4. Daemon de administración de afinidad NUMA (numad)

numad es un daemon de administración de afinidad NUMA. Monitoriza la topología de NUMA y el uso de recursos dentro de un sistema con el fin de mejorar de forma dinámica la asignación de recursos y administración de NUMA (y por ende, el rendimiento del sistema).

Según la carga de trabajo del sistema, **numad** puede proporcionar un punto de referencia de mejoras de rendimiento hasta de 50%. Para obtener estas ganancias, **numad** accede de forma periódica a la información desde el sistema de archivos **/proc** para monitorizar los recursos del sistema disponibles por nodo. En daemon intenta entonces colocar los procesos importantes en nodos de NUMA que tienen alineada suficiente memoria y recursos de CPU para óptimo rendimiento de NUMA. El umbral actual para administración de procesos es de por lo menos 50% de una CPU y al menos 300 MB de memoria. **numad** intenta mantener un nivel de utilización de recursos y re-equilibrar las asignaciones cuando sea necesario al trasladar los procesos entre nodos de NUMA.

numad también proporciona un servicio de consejería de pre-colocación que puede ser solicitado por varios sistemas de administración de trabajo para asistir en la vinculación inicial de CPU y recursos de memoria para sus procesos. Este servicio de pre-colocación está disponible independiente de si **numad** se esta ejecutando o no, como un daemon en el sistema. Para mayor información sobre el uso de la opción **-w** para consejo de pre-colocación, consulte la página de manual: **man numad**.

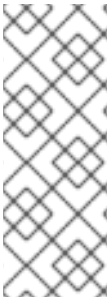
4.1.4.1. Se beneficia de numad

numad se beneficia principalmente de sistemas con procesos de larga ejecución que consumen cantidades importantes de recursos, en particular, cuando dichos procesos hacen parte de un subconjunto del total de recursos del sistema.

numad también puede beneficiarse de las aplicaciones que consumen valor de recursos de múltiples nodos de NUMA. Sin embargo, los beneficios que **numad** proporciona decrecen a medida que decrece el porcentaje de recursos consumidos en un sistema.

numad es poco probable mejorar el rendimiento de procesos cuando los procesos se ejecutan por unos pocos minutos o cuando no consumen muchos recursos.También es poco probable que los sistemas con patrones de acceso de memoria continua, tales como base de datos en-memoria, se beneficien del uso de **numad**.

4.1.4.2. Modos de operación



NOTA

Las estadísticas de conteo de memoria de Kernel pueden contradecirse después de grandes cantidades de fusión. Como tal, **numad** se puede confundir cuando el daemon KSM fusiona grandes cantidades de memoria. El daemon KSM será más consciente de NUMA en futuros lanzamientos. Sin embargo, actualmente, si su sistema tiene una gran cantidad de memoria libre, podrá realizar un rendimiento superior al apagar y desactivar el daemon KSM.

numad sirve de dos formas:

- como un servicio
- como un ejecutable

4.1.4.2.1. Uso de numad como un servicio

Mientras el servicio de **numad** se ejecuta, intentará ajustar de forma dinámica el sistema basado en su carga de trabajo.

Para iniciar el servicio, ejecute:

```
# service numad start
```

Para hacer que el servicio sea persistente a través de reinicios, ejecute:

```
# chkconfig numad on
```

4.1.4.2.2. Uso de numad como un ejecutable

Para usar **numad** como un ejecutable, ejecute:

```
# numad
```

numad se ejecutará hasta que se detenga. Mientras se ejecuta, sus actividades se registran en **/var/log/numad.log**.

Para restringir la administración de **numad** a un proceso específico, inícielo con las siguientes opciones.

```
# numad -S 0 -p pid
```

-p pid

Añade el *pid* especificado a una lista de inclusión explícita. El proceso especificado no será administrado, sino hasta cuando alcance el umbral de importancia del proceso de **numad**.

-S modo

El parámetro ***-S*** especifica el tipo de proceso que escanea. Al establecerlo a **0** limita la administración de **numad** a proceso incluidos de forma explícita.

Para detener **numad**, ejecute:

```
# numad -i 0
```

La detención de **numad** no retira los cambios que ha hecho para mejorar afinidad de NUMA. Si el sistema usa cambios de forma significativa, al ejecutar **numad** otra vez, ajustará la afinidad para mejorar rendimiento bajo las nuevas condiciones.

Para obtener mayor información sobre opciones disponibles de **numad**, consulte la página de manual de **numad**: **man numad**.

4.2. PROGRAMACIÓN DE CPU

El *programador* es responsable de mantener las CPU en el sistema ocupado. El programador de Linux implementa un número de *políticas de programación*, las cuales determinan cuándo y por cuánto tiempo se ejecuta un hilo en un núcleo de CPU particular.

Las políticas de programación se dividen en dos categorías principales:

1. Políticas de Realtime
   - SCHED_FIFO
   - SCHED_RR
2. Políticas normales
   - SCHED_OTHER
   - SCHED_BATCH
   - SCHED_IDLE

4.2.1. Políticas de programación Realtime

Los hilos de Realtime se programan primero y luego los hilos normales después de que todos los hilos Realtime hayan sido programados.

Las políticas de *realtime* sirven para tareas críticas de tiempo que deben completarse sin interrupciones.

SCHED_FIFO

Esta política también se conoce como *programación de prioridad estática*, porque define una prioridad fijada (entre 1 y 99) para cada hilo. El programador escanea una lista de hilos SCHED_FIFO en orden de prioridad y programa el hilo con prioridad más alta listo para ejecutarse. Este hilo se ejecuta hasta que se bloquea, sale o es prevaciado por una prioridad mayor que está lista para ejecutarse.

La prioridad inferior del hilo de tiempo real será programada por antes de cualquier hilo con una política de no-realtime; si únicamente existe un hilo de tiempo real, el valor de prioridad **SCHED_FIFO** no es problema.

SCHED_RR

Una variante round-robin de la política **SCHED_FIFO**. Los hilos de **SCHED_RR** también reciben una prioridad fijada entre 1 y 99. Sin embargo, los hilos con la misma prioridad se reprograman al estilo round-robin dentro de un cierto quantum, o porción de tiempo. La llamada de sistema

sched_rr_get_interval(2) retorna el valor de la porción de tiempo, pero la duración de la porción del tiempo no puede ser establecida por usuario. Esta política es útil si se necesita múltiples hilos con la misma prioridad.

Para obtener mayor información sobre la semántica definida de políticas de programación de tiempo real, consulte *IEEE 1003.1 POSIX standard* en Interfaces del sistema — Realtime, disponible en http://pubs.opengroup.org/onlinepubs/009695399/functions/xsh_chap02_08.html.

La mejor práctica para definir prioridad de hilos es iniciar a una prioridad baja y aumentarla únicamente cuando se identifique latencia legítima. Los hilos de tiempo real no se dividen como los hilos normales. Los hilos **SCHED_FIFO** se ejecutan hasta que se bloqueen, salgan, o sean prevaciadas por un hilo con una prioridad superior. Por lo tanto, establecer una prioridad de 99 no se recomienda, ya que esto sitúa su proceso al mismo nivel de prioridad como hilos de migración y vigilancia. Si estos hilos se bloquean debido a que su hilo va a un bucle de computación, no podrán ejecutarse. Los sistemas de un procesador, terminarán bloqueándose en esta situación.

En el kernel de Linux, la política **SCHED_FIFO** incluye un mecanismo de capa de banda ancha. Esto protege a los programadores de aplicaciones de tiempo real de las tareas de tiempo real que podrían monopolizar la CPU. Este mecanismo puede ajustarse mediante los siguientes parámetros del sistema de archivos **/proc**:

/proc/sys/kernel/sched_rt_period_us

Define el periodo de tiempo para considerarse cien por ciento de ancho de banda de CPU, en microsegundos ('us' siendo el equivalente más cercano a 'µs' en texto plano). El valor predeterminado es 1000000µs, o 1 segundo.

/proc/sys/kernel/sched_rt_runtime_us

Define el periodo de tiempo dedicado a la ejecución de hilos en tiempo real, en microsegundos (siendo 'us' la forma más cercana en texto plano a 'µs'). El valor predeterminado es 950000µs, o 0.95 segundos.

4.2.2. Políticas de programación normales

Hay tres políticas de programación normal: **SCHED_OTHER**, **SCHED_BATCH** y **SCHED_IDLE**. Sin embargo, las políticas **SCHED_BATCH** y **SCHED_IDLE** son para trabajos de una prioridad muy baja, y como tal son de interés limitado en una guía de ajuste de rendimiento.

SCHED_OTHER, o SCHED_NORMAL

La política de programación predeterminada. Esta política usa el Programador de reparto justo (CFS) para proporcionar periodos de acceso justo a todos los hilos por medio de esta política. CFS establece una lista dinámica de prioridades en parte basada en el valor de ***niceness*** de cada hilo de proceso. (Consulte la *Guía de implementación* para obtener mayor información sobre este parámetro y el sistema de archivos **/proc**.) Esto le otorga a los usuarios un nivel de control indirecto sobre la prioridad de procesos, pero la lista de prioridad dinámica solamente puede ser cambiada por el CFS.

4.2.3. Selección de políticas

Seleccionar la política de programador correcta para hilos de una aplicación no siempre es una tarea sencilla. En general, las políticas de tiempo real deben utilizarse para tareas de tiempo crítico o importantes que necesitan reprogramarse rápidamente y no ejecutarse por un largos periodos de

tiempo. Las políticas normales generalmente producen mejores resultados de rendimiento que las políticas de tiempo real debido a que permiten al programador ejecutar hilos de una forma más eficiente.

Si está administrando grandes cantidades de hilos y si está interesado principalmente en el rendimiento de procesamiento de datos (paquetes de redes por segundo, escrituras a disco, etc.) entonces use **SCHED_OTHER** y permita que el sistema administre el uso de CPU por usted.

Si está interesado en tiempo de respuesta de eventos (latencia) entonces use **SCHED_FIFO**. Si tiene una pequeña cantidad de hilos, considere aislar un socket y trasladarlo a núcleos de socket para que no haya hilos compitiendo entre sí por tiempo en los núcleos.

4.3. INTERRUPCIONES Y AJUSTE DE IRQ

Una solicitud de interrupción de cola (IRQ) es una solicitud de un servicio, enviado en el nivel de hardware. Las interrupciones pueden ser enviadas por una línea de hardware dedicado o a través de un bus de hardware como un paquete de información (una Interrupción de mensajes señalada o MSI).

Cuando las interrupciones están habilitadas, la recepción de una IRQ le pide al interruptor, detener el contexto. El código de envío de interrupciones de Kernel, recupera el número de IRQ y su lista asociada de Rutinas de servicio de interrupciones registradas (ISR), y a su vez, llama a cada ISR. La ISR reconoce las interrupciones e ignora las interrupciones redundantes de la misma IRQ, luego pone en cola un indicador diferido para terminar de procesar la interrupción e impedir que la ISR ignore las interrupciones futuras.

El archivo **/proc/interrupts** lista el número de interrupciones por CPU por dispositivo de E/S. Muestra el número de IRQ, el número de dicha interrupción manejada por cada núcleo de CPU, el tipo de interrupción y la lista delimitada por comas de controladores registrados para recibir esa interrupción. (Para obtener mayor información, consulte la página de manual: **man 5 proc**)

Las IRQ tienen la propiedad de "afinidad", ***smp_affinity***, que define los núcleos de CPU permitidos para ejecutar la ISR de dicha IRQ. Esta propiedad puede servir para mejorar el rendimiento de aplicaciones al asignar tanto afinidad de interrupciones como afinidad de hilos de aplicaciones a uno o más núcleos de CPU específicos. Así permite compartir la línea de cache entre aplicaciones de interrupciones e hilos.

El valor de afinidad de interrupciones para un número IRQ específico, es almacenado en el archivo **/proc/irq/*NÚMERO_IRQ*/smp_affinity** asociado, el cual se puede ver y modificar mediante el usuario de root. El valor almacenado en estee archivo es una máscara de bits hexadecimal que representa todos los núcleos de CPU en el sistema.

A manera de ejemplo, para establecer la afinidad de interrupciones para el controlador de Ethernet en un servidor con cuatro núcleos de CPU, primero determine el número IRQ asociado con el controlador de Ethernet:

```
# grep eth0 /proc/interrupts
32:   0     140      45       850264      PCI-MSI-edge      eth0
```

Use el número de IRQ para localizar el archivo apropiado de ***smp_affinity***:

```
# cat /proc/irq/32/smp_affinity
f
```

El valor predeterminado para smp_affinity es **f**, lo que significa que la IRQ puede servirse de las CPU en el sistema. Si configura este valor a **1**, como se muestra a continuación, significa que solamente la CPU 0 puede servir esta interrupción:

```
# echo 1 >/proc/irq/32/smp_affinity
# cat /proc/irq/32/smp_affinity
1
```

Las comas pueden servir para delimitar los valores de ***smp_affinity*** para grupos de 32 bits. Esto se requiere en sistemas con más de 32 núcleos. Por ejemplo, el siguiente ejemplo muestra que la IRQ 40 se sirve en todos los núcleos de un sistema de núcleos de 64:

```
# cat /proc/irq/40/smp_affinity
ffffffff,ffffffff
```

Para servicio IRQ 40 solamente en los núcleos superiores a 32 de un sistema de núcleos de 64, debe hacer lo siguiente:

```
# echo 0xffffffff,00000000 > /proc/irq/40/smp_affinity
# cat /proc/irq/40/smp_affinity
ffffffff,00000000
```

NOTA

En sistemas que soportan *direccionamiento de interrupciones*, al modificar ***smp_affinity*** de una IRQ se configura el hardware para que la decisión de servir una interrupción con una CPU determinada se haga en el nivel del hardware, sin ninguna intervención del kernel.

4.4. MEJORAS A NUMA EN RED HAT ENTERPRISE LINUX 6

Red Hat Enterprise Linux 6 incluye una serie de mejoras para capitalizar en todo el potencial actual del hardware escalable. Esta sección entrega una visión general de las mejoras de rendimiento más importantes relacionadas con NUMA proporcionadas por Red Hat Enterprise Linux 6.

4.4.1. Optimización de escalabilidad y en vacío

4.4.1.1. Mejoras en reconocimiento de topología

Las siguientes mejoras permiten a Red Hat Enterprise detectar información de arquitectura y hardware de bajo nivel, mejorando así su capacidad para optimizar de forma automática el procesamiento en su sistema.

Detección de topología mejorada

Permite al sistema operativo detectar información de hardware de bajo nivel (tal como CPU, hiper procesos, núcleos, conectores, nodos de NUMA y tiempos de acceso en su sistema.

Programador de reparto justo

Este nuevo modo de programación garantiza que el tiempo de ejecución sea compartido por igual entre los procesos elegibles. Al combinarlo con esta detección de topología permite la programación de los procesos en las CPU dentro del mismo conector para evitar el costoso acceso de memoria remota, y garantizar que el contenido de cache sea preservado siempre que sea posible.

`malloc`

malloc ahora se ha optimizado para garantizar que las regiones de memoria asignadas al proceso sean tan físicas como sea posible. Incluido el aumento de velocidad de acceso de memoria.

asignación de búfer de E/S skbuff

Al igual que **malloc**, ha sido optimizado para usar memoria que está cerca físicamente a las operaciones de manejo de E/S tales como la interrupciones.

Afinidad de interrupciones de dispositivo

La información registrada por los controladores de dispositivos acerca de cuál CPU maneja cada una de las interrupciones, puede utilizarse para restringir el manejo de interrupciones a las CPU dentro del mismo conector físico, preservando así, la afinidad de cache y limitando la comunicación entre conectores de alto volumen.

4.4.1.2. Mejoras en sincronización de multiprocesador

La coordinación de tareas entre múltiples procesadores requiere operaciones de consumo de tiempo frecuentes para garantizar que los procesos que se ejecutan en paralelo no comprometan la integridad de los datos. Red Hat Enterprise Linux incluye las siguientes mejoras de rendimiento en esta área.

Cerrojos de Leer-Copiar-Actualizar (RCU)

El 90% de cerrojos suele adquirirse para propósitos de lectura. El cerramiento de RCU retira la necesidad para obtener un cerrojo de acceso exclusivo cuando los datos que se acceden no sean modificados. Este modo de cerramiento ahora se utiliza para asignar o desasignar operaciones.

algoritmos por CPU y por socket

Muchos algoritmos han sido actualizados para realizar coordinación de cerrojos entre las CPU que cooperan en el mismo socket para permitir un cerramiento más específico. Numerosos Spinlocks han sido sustituidos por métodos de cerramiento por socket, y zonas de asignador de memoria actualizadas y listas de páginas de memoria relacionadas con la lógica de asignación de memoria para atravesar un subconjunto de estructuras de datos de operaciones de asignación o desasignación de memoria.

4.4.2. Optimización de virtualización

Debido a que KVM utiliza la funcionalidad de kernel, los huéspedes virtualizados de KVM se benefician inmediatamente de la optimización en vacío. Red Hat Enterprise Linux también incluye una serie de mejoras que permiten a los huéspedes virtualizados enfocar el nivel de rendimiento del sistema en vacío. Estas mejoras se enfocan en la ruta de E/S en el acceso de almacenamiento y redes, lo cual permite incluso a las cargas de trabajo intensivas tales como base de datos y servicio de archivos, hacer uso de la implementación virtualizada. Entre las mejoras de rendimiento de sistemas virtualizados están:

Enclavar la CPU

Los huéspedes virtuales pueden ser vinculados para que se ejecuten en un conector específico, con el fin de optimizar el uso de la cache local y retirar la necesidad de costosas comunicaciones interconectadas y acceso de memoria remota.

Páginas gigantes transparentes (THP)

Con las THP habilitadas, el sistema realiza automáticamente las solicitudes de asignación de memoria de reconocimiento de NUMA para grandes cantidades de memoria contigua, reduciendo

así, la contención del cerrojo y el número de operaciones requeridas de Translation Lookaside Buffer (TLB) y generando un aumento de rendimiento de más de 20% en huéspedes virtuales.

Implementación de E/S basada en Kernel

El subsistema de E/S del huésped virtual ahora se implementa en el kernel, reduciendo así, el costo de la comunicación internodal y el acceso de memoria al evitar una cantidad significativa de cambios de contexto, y sincronización y gasto de comunicaciones.

CAPÍTULO 5. MEMORIA

Este capítulo presenta una visión general de las funcionalidades de administración de memoria disponibles en Red Hat Enterprise Linux, y cómo utilizarlas para optimizar el uso de memoria en su sistema.

5.1. HUGE TRANSLATION LOOKASIDE BUFFER (HUGETLB)

Las direcciones de memoria física se traducen en direcciones de memoria virtual como parte de administración de memoria. La relación asignada de direcciones físicas a virtuales se almacena en una estructura de datos conocida como la tabla de páginas. Puesto que la tabla de páginas para cada asignación de direcciones significaría un consumo de tiempo y recursos costoso, existe una memoria cache para direcciones utilizadas recientemente. Esta cache se denomina Translation Lookaside Buffer (TLB).

Sin embargo, TLB puede únicamente guardar en cache muchas asignaciones de mapas. Si la asignación de direcciones solicitada no está en TLB, la tabla de página debe aún ser leída para determinar la asignación de la dirección física a la virtual. Esto se conoce como una pérdida de TLB o "TLB miss". Es más probable que las aplicaciones con grandes requerimientos de memoria se afecten más por pérdidas de TLB que las aplicaciones con requerimientos de memoria mínima, debido a la relación entre sus requisitos de memoria y el tamaño de las páginas de tabla utilizadas para guardar en cache las asignaciones de direcciones en TLB. Ya que cada pérdida implica la lectura de tabla de página, es importante evitar pérdidas en lo que sea posible.

Huge Translation Lookaside Buffer (HugeTLB) permite que la memoria sea administrada en grandes segmentos para que más asignaciones de direcciones puedan ser guardadas en cache al mismo tiempo. De esta manera se reduce la probabilidad de 'TLB miss', lo cual a su vez mejora el rendimiento en aplicaciones con grandes requerimientos de memoria.

Para obtener mayor información sobre cómo configurar HugeTLB, consulte la documentación de kernel: **`/usr/share/doc/kernel-doc-`*`version`*`/Documentation/vm/hugetlbpage.txt`**

5.2. PÁGINAS GIGANTES Y PÁGINAS GIGANTES TRANSPARENTES

La memoria es administrada en bloques conocidos como *páginas*. Una página es de 4096 bytes. 1MB de memoria es igual a 256 páginas; 1GB de memoria es igual a 256.000 páginas, etc. Las CPU tienen una *unidad de administración de memoria* incorporada que contiene una lista de dichas páginas y cada página se referencia a través de una *entrada de tabla de páginas*.

Hay dos formas de habilitar el sistema para que administre grandes cantidades de memoria:

- Aumentar el número de entradas de tabla de páginas en la unidad de administración de memoria
- Aumentar el tamaño de página

El primer método es costoso, puesto que la unidad de administración de hardware en un procesador moderno únicamente soporta cientos o miles de entradas de tabla de página. Además, la administración de algoritmos de hardware y memoria que funcionan bien con miles de páginas (MB de memoria) puede tener dificultad al funcionar con millones (o incluso miles de millones) de páginas. Esto se traduce en problemas de rendimiento: cuando una aplicación necesita usar más páginas de memoria que las que soporta la unidad de administración de memoria, el sistema se conmuta a una administración de memoria basada en software más lento, lo cual hace que todo el sistema se ejecute más lentamente.

Red Hat Enterprise Linux 6 implementa el segundo método a través de las *páginas gigantes*.

Las páginas gigantes (o Hugepages) son bloques de memoria que vienen en tamaños de 2MB y 1GB. Las tablas de páginas utilizadas por las páginas de 2MB son apropiadas para administrar múltiples GB de memoria y las tablas de páginas de 1GB son las mejores para escalar a TB de memoria.

Las páginas gigantes deben asignarse en el momento de arranque. También son difíciles de administrar y suelen requerir cambios significativos al código para utilizar de forma efectiva. Como tal, Red Hat Enterprise Linux 6 también implementó el uso de la *página gigante transparente* (THP). THP es una capa de abstracción que automatiza la mayoría de los aspectos de creación, manejo y uso de páginas gigantes.

THP oculta parte de su complejidad del uso de páginas gigantes a los administradores y desarrolladores del sistema. Puesto que la meta de THP es mejorar el rendimiento, sus desarrolladores (tanto de la comunidad como de Red Hat) han probado y optimizado THP a través de un amplio rango de sistemas, configuraciones, aplicaciones y cargas de trabajo. Esto ha permitido a los parámetros de THP mejorar el rendimiento de la mayoría de configuraciones del sistema.

Observe que la THP puede únicamente asignar regiones de memoria anónima como espacio de montículo y de pila.

5.3. CÓMO UTILIZAR VALGRIND PARA PERFILAR EL USO DE MEMORIA

Valgrind es un marco de trabajo que proporciona instrumentación para binarios de espacio de usuario. Se distribuye con un número de herramientas que sirven para perfilar y analizar el rendimiento del programa. Las herramientas descritas en esta sección proporcionan un análisis que puede ayudar en la detección de errores de memoria tales como el uso de memoria no inicializada y la asignación o desasignación incorrecta de memoria. Todas se incluyen en el paquete valgrind, y pueden ejecutarse con el siguiente comando:

```
valgrind --tool=nombre de herramienta programa
```

Remplace *nombre de herramienta* por el nombre de la herramienta que usted desea usar (para perfilar la memoria, **memcheck**, **massif**, o **cachegrind**), y *programa* por el programa que desea perfilar con Valgrind. Tenga en cuenta que la instrumentación de Valgrind hará que su programa se ejecute más lentamente que lo normal.

Para obtener una visión general de las capacidades de Valgrind vaya a la Sección 3.5.3, “Valgrind”. Información adicional, incluida la disponible sobre conectores para Eclipse, se encuentra en la *Guía del desarrollador*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/. También puede ver información en la página de manual con el comando **man valgrind** cuando el paquete valgrind esté instalado o se encuentre en los siguientes sitios:

- **/usr/share/doc/valgrind-*versión*/valgrind_manual.pdf**, y
- **/usr/share/doc/valgrind-*version*/html/index.html**.

5.3.1. Perfilar uso de memoria con Memcheck

Memcheck es la herramienta predeterminada de Valgrind, y puede ejecutarse con **valgrind *programa***, sin especificar **--tool=memcheck**. Detecta y reporta una serie de errores que pueden dificultar la detección y el diagnóstico, tal como el acceso a la memoria, lo cual no debería ocurrir, el uso de valores indefinidos o no inicializados, memoria de montículo liberada incorrectamente, punteros superpuestos y escapes de memoria. Los programas se ejecutan diez o treinta veces más lentamente con Memcheck que cuando se ejecutan normalmente.

Memcheck retorna errores específicos según el tipo de problema que detecte. Estos errores se describen en la documentación de Valgrind, la cual se incluye en **/usr/share/doc/valgrind-*versión*/valgrind_manual.pdf**.

Observe que Memcheck solamente reporta dichos errores— no evita que ocurran. Si su programa accede a memoria de tal forma que resulte en una falla de segmentación, la falla de segmentación aún ocurrirá. Sin embargo, Memcheck registrará un mensaje de errores inmediatamente antes de la falla.

Memcheck proporciona opciones de línea de comandos que pueden servir para enfocar el proceso de verificación. Algunas de las opciones disponibles son:

--leak-check

Si Memcheck está habilitada, buscará escapes de memoria cuando el programa de cliente termine. El valor predeterminado es **summary**, el cual emite el número de escapes que encuentra. Otros valores posibles son **yes** y **full**, ambos informan sobre cada escape, y el valor **no**, desactiva la verificación de escape de memoria.

--undef-value-errors

Si Memcheck está habilitada (**yes**), reportará errores cuando se utilicen valores indefinidos. Si Memcheck está inhabilitada (**no**), los valores de errores indefinidos no se reportarán. Este valor es el predeterminado. Al inhabilitarlo se agiliza un poco Memcheck.

--ignore-ranges

Permite al usuario especificar uno más rangos que Memcheck debe ignorar al verificar el direccionamiento. Los múltiples rangos están delimitados por comas, por ejemplo, **--ignore-ranges=0xPP-0xQQ,0xRR-0xSS**.

Para obtener una lista completa de las opciones, consulte la documentación incluida en **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

5.3.2. Perfilar uso de cache con Cachegrind

Cachegrand simula su interacción de programa con la jerarquía de cache de máquina y el predictor de saltos (opcionalmente). Este rastrea el uso de la instrucción simulada del primer nivel y las caches de datos para detectar interacción de código pobre con este nivel de cache; y la cache de último nivel, ya sea de segundo o tercer nivel, para rastrear el acceso a la memoria principal. Así los programas que se ejecutan con Cachegrind, se ejecutan de una forma de veinte a cien veces más lenta que cuando se ejecutan normalmente.

Para ejecutar Cachegrind, ejecute el siguiente comando, remplace *programa* por el programa que desea perfilar con Cachegrind:

```
# valgrind --tool=cachegrind programa
```

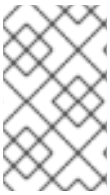
Cachegrind puede reunir las siguientes estadísticas para el programa completo, y para cada una de sus funciones:

- lecturas de cache de instrucciones de primer nivel (o instrucciones ejecutadas) y pérdidas de lecturas, y pérdidas de lectura de instrucción cache de último nivel;
- lecturas de cache de datos (o lecturas de memoria), pérdidas de lectura y pérdidas de lectura de datos cache de último nivel;

- escrituras de cache de datos (o escrituras de memoria), pérdidas de escritura y pérdidas de escritura de cache de último nivel.
- saltos condicionales ejecutados y predichos erróneamente; y
- saltos indirectos ejecutados y predichos erróneamente

Cachegrind imprime en la consola información de resumen sobre estas estadísticas y escribe más información de perfil en un archivo (**cachegrind.out.*pid*** predeterminado, donde *pid* es el número de proceso en el que se ejecutó Cachegrind). Este archivo puede ser procesado por la herramienta de acompañamiento **cg_annotate** , de esta manera:

```
# cg_annotate cachegrind.out.pid
```



NOTA

cg_annotate puede emitir líneas de más de 120 caracteres, según la longitud de la ruta. Para obtener una salida más clara y fácil de leer, se recomienda que su ventana de terminal tenga al menos este ancho antes de ejecutar el comando antes mencionado.

También puede comparar los archivos de perfil creados por Cachegrind para simplificar el rendimiento del programa de tabla antes y después de un cambio. Para ello, use el comando **cg_diff**, remplace *primero* por el primer perfil de salida y *segundo* por el archivo de salida de siguiente perfil:

```
# cg_diff primero segundo
```

Este comando produce un archivo de salida combinado, el cual puede verse en mayor detalle con **cg_annotate**.

Cachegrind soporta una cantidad de opciones para enfocar su salida. Algunas de las opciones disponibles son:

--I1

Especifica el tamaño, la capacidad de asociación y el tamaño de línea de la cache de instrucción de primer nivel, separados por comas: **--I1=*tamaño*,*asociatividad*,*tamaño de línea***.

--D1

Especifica el tamaño, capacidad de asociación y tamaño de línea de cache de datos de primer nivel, separados por comas: **--D1=*tamaño*,*asociatividad*,*tamaño de línea***.

--LL

Especifica el tamaño, la capacidad de asociación y el tamaño de línea de la cache de instrucción de último nivel, separados por comas: **--LL=*tamaño*,*asociatividad*,*tamaño de línea***.

--cache-sim

Habilita o inhabilita la recolección de acceso de datos y conteos de pérdidas. El valor predeterminado es **yes** (habilitado).

Observe que al inhabilitar ambos saltos y **--branch-sim** no le dejará a Cachegrind ninguna información para recolectar.

--branch-sim

Habilita o inhabilita la instrucción de saltos y conteos predichos erróneamente. Se establece a **no** (inhabilitado) de forma predeterminada, ya que ralentiza a Cachegrind en un 25 por ciento.

Observe que al inhabilitar ambos saltos y **--cache-sim** dejará a Cachegrind sin ninguna información para recolectar.

Para obtener una lista completa de las opciones, consulte la documentación incluida en **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

5.3.3. Cómo perfilar montículo y espacio de montículo con Massif

Massif mide el espacio de montículo usado por un determinado programa; tanto el espacio útil como cualquier espacio adicional asignado para propósitos de contabilidad y alineación. Massif puede ayudar a reducir la cantidad de memoria utilizada por su programa,lo cual puede aumentar la velocidad del programa y reducir la posibilidad de que el programa agote el espacio de la máquina en la cual se ejecuta. Massif también proporciona información sobre las partes del programa responsables de asignar la memoria de montículo. Los programas que se ejecutan con Massif son aproximadamente veinte veces más lentas que la velocidad de ejecución normal.

Para perfilar el uso de montículo de un programa, especifique **massif** como la herramienta de Valgrind que desea utilizar:

```
# valgrind --tool=massif program
```

El perfilado de datos reunidos por Massif se escribe a un archivo, el cual se denomina de forma predeterminada **massif.out.*pid***, donde *pid* es el ID del proceso del *programa* especificado.

El perfilado de datos pueden graficarse con el comando **ms_print**, así:

```
# ms_print massif.out.pid
```

Produce una gráfica que muestra el consumo de memoria en la ejecución del programa, y la información detallada sobre los sitios responsables de asignar en varios puntos en el programa, incluido en el punto de asignación de memoria máxima.

Massif proporciona una cantidad de opciones de línea de comando que sirven para dirigir la salida de la herramienta. Algunas de las opciones disponibles son:

--heap

Especifica si realiza o no el perfilado de montículo. El valor predeterminado es **yes**. El perfilado de montículo puede ser desactivado al establecer esta opción a **no**.

--heap-admin

Especifica el número de bytes por bloque a usar para administrar el perfilado de montículo. El valor predeterminado es de **8** bytes por bloque.

--stacks

Especifica si realiza o no el perfilado de pila. El valor predeterminado es **no** (desactivado). Para habilitar el perfilado de pila, establezca esta opción a **yes**, pero tenga en cuenta que al hacerlo ralentizará ampliamente a Massif. Observe también que Massif supone que la pila principal tiene un

tamaño de cero al inicio para indicar el tamaño de porción de pila sobre el cual el ente perfilado tiene control.

`--time-unit`

Especifica la unidad de tiempo utilizada para el perfilado. Hay tres valores válidos para esta opción: instrucciones ejecutadas (**`i`**), el valor predeterminado, el cual es útil en la mayoría de los casos; tiempo real (**`ms`**, en milisegundos), el cual puede ser útil en algunos casos; y bytes asignados o desasignados en el montículo y/o en la pila (**`B`**), el cual sirve para la mayoría de programas de ejecución corta y para pruebas, porque es el más reproducible a través de diferentes máquinas. Esta opción sirve para graficar salida de Massif con **`ms_print`**.

Para obtener una lista completa de las opciones, consulte la documentación incluida en **`/usr/share/doc/valgrind-`*`version`*`/valgrind_manual.pdf`**.

5.4. CAPACIDAD DE AJUSTE

Por favor lea esta sección para obtener un resumen de memoria, kernel y capacidad de sistema de archivos, los parámetros relacionados con cada uno y las concesiones al ajustar estos parámetros.

Para establecer estos parámetros de forma temporal durante el ajuste, 'echo' el valor deseado al archivo apropiado en el sistema de archivos proc. Por ejemplo, para establecer de forma temporal ***`overcommit_memory`*** a **`1`**, ejecute:

```
# echo 1 > /proc/sys/vm/overcommit_memory
```

Observe que la ruta al parámetro en el sistema de archivos proc, depende del sistema afectado por el cambio.

Para establecer de forma persistente estos valores, necesitará usar el comando **`sysctl`**. Para obtener mayor información, consulte la *Guía de implementación*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Ajustables de memoria relacionados con la capacidad

Cada uno de los siguientes parámetros se localiza en el sistema de archivos proc, **`/proc/sys/vm/`** .

`overcommit_memory`

Define las condiciones que determinan si se acepta o niega una solicitud de memoria grande. Hay tres valores posibles para este parámetro:

- **`0`** — el parámetro predeterminado. El kernel realiza sobre-envío de memoria heurística al estimar la cantidad de memoria disponible y fallar solicitudes que son evidentemente inválidas. Infortunadamente, como la memoria se asigna mediante una heurística en lugar de un algoritmo preciso, este parámetro, algunas veces autoriza la sobrecarga de memoria disponible en el sistema.
- **`1`** — el kernel realiza el manejo de sobre-envío de no memoria. Bajo este parámetro, la memoria potencial para sobrecarga aumenta, pero también el rendimiento para tareas intensivas de rendimiento.

- **2** — el kernel niega solicitudes para memoria igual o mayor que swap y el porcentaje de RAM físico especificado en ***overcommit_ratio***. Este parámetro es mejor si desea un menor riesgo de sobreasignación de memoria.

NOTA

Este parámetro únicamente se recomienda para sistemas con áreas de swap más grandes que su memoria física.

overcommit_ratio

Especifica el porcentaje de memoria RAM física que se tiene en cuenta cuando ***overcommit_memory*** se establece a **2**. El valor predeterminado es **50**.

max_map_count

Define el número máximo de áreas de mapas que un proceso puede usar . En la mayoría de los casos, el valor predeterminado de **65530** es el apropiado. Aumente este valor si su aplicación necesita asignar más de este número de archivos.

nr_hugepages

Define el número de páginas gigantes configuradas en el kernel. El valor predeterminado es 0. Solamente se pueden asignar (o desasignar) páginas gigantes si hay las suficientes páginas libres físicas contiguas. Las páginas reservadas por este parámetro no se pueden usar para otros propósitos. Puede obtener mayor información de la documentación instalada en: **/usr/share/doc/kernel-doc-*kernel_version*/Documentation/vm/hugetlbpage.txt**

Ajustables de kernel relacionados con la capacidad

Cada uno de los siguientes parámetros se localiza en el sistema de archivos proc, **/proc/sys/kernel/**.

msgmax

Define el tamaño máximo permitido en bytes de un mensaje individual en bytes de cualquier mensaje en la cola de mensajes. Este valor no debe exceder el tamaño de la cola (***msgmnb***). El valor predeterminado es **65536**.

msgmnb

Define el tamaño máximo en bytes de una cola de mensajes. El valor predeterminado es **65536**.

msgmni

Define el número máximo de identificadores de cola de mensajes (y por lo tanto, el número máximo de colas). El valor predeterminado en máquinas de arquitectura de 64 bits es **1985**; para arquitectura de 32 bits, el valor predeterminado es **1736**.

shmall

Define la cantidad total de memoria compartida en bytes que puede utilizarse en el sistema al mismo tiempo. El valor predeterminado en máquinas de arquitectura de 64 bits es **4294967296**; para arquitectura de 32 bits, el valor predeterminado es **268435456**.

shmmax

Define el máximo segmento de memoria compartida por el kernel, en bytes. El valor predeterminado en máquinas de arquitectura de 64 bits es **68719476736**; para arquitectura de 32 bits, el valor predeterminado es **4294967295**. Sin embargo, el kernel soporta valores mucho más grandes.

shmmni

Define el número máximo de todo el sistema de segmentos de memoria compartida. El valor predeterminado es **4096** tanto en arquitectura de 64 bits como en la de 32 bits.

threads-max

Define el número máximo de hilos (tareas) en todo el sistema que van a ser utilizados por el kernel al mismo tiempo. El valor predeterminado es igual al valor de kernel ***max_threads***. La fórmula en uso es:

```
max_threads = mempages / (8 * THREAD_SIZE / PAGE_SIZE )
```

El valor mínimo de ***threads-max*** es **20**.

Ajustables de sistema de archivos relacionados con la capacidad

Cada uno de los siguientes parámetros se localiza en el sistema de archivos proc, **/proc/sys/fs/**

aio-max-nr

Define el máximo número de eventos permitidos en todos los contextos asíncronos de E/S. El valor predeterminado es **65536**. Observe que al cambiar este valor no se preasigna o redimensiona ninguna estructura de datos de kernel.

file-max

Lista el número máximo de identificadores de archivos asignados por el kernel. El valor predeterminado coincide con el valor de ***files_stat.max_files*** en el kernel, el cual se establece al valor más grande, ya sea de **(mempages * (PAGE_SIZE / 1024)) / 10**, o **NR_FILE** (8192 en Red Hat Enterprise Linux). El aumento de este valor puede corregir errores ocasionados por la falta de identificadores de archivos disponibles.

Ajustables para matar un proceso en falta de memoria

Falta de memoria o una OOM, se refiere a un estado en computación donde toda la memoria disponible, incluido el espacio swap, ha sido asignada. Esta situación hace que el sistema entre en pánico y deje de funcionar. Sin embargo, la configuración del parámetro ***/proc/sys/vm/panic_on_oom*** a **0** instruye al kernel para que llame a la función **oom_killer** cuando se presente una OOM. **oom_killer** puede matar procesos no autorizados y el sistema sobrevive.

El parámetro a continuación puede establecerse por proceso, lo cual le otorga control sobre los procesos que la función **oom_killer** puede matar. Dicha función se localiza en el sistema de archivos proc, **/proc/*pid*/**, donde *pid* es el número de ID de proceso.

oom_adj

Define un valor de **-16** a **15** que ayuda a determinar el ***oom_score*** de un proceso. Entre más alto sea el valor de ***oom_score***, más probabilidad habrá de que **oom_killer** mate el proceso. Si establece ***oom_adj*** a un valor de **-17** se desactivará el **oom_killer** para ese proceso.

IMPORTANTE

Cualquier proceso generado por el proceso ajustado heredará ese ***oom_score*** de proceso. Por ejemplo, si un proceso **sshd** está protegido de la función **oom_killer**, todos los procesos iniciados por dicha sesión SSH también se protegerán. Esto puede afectar la capacidad de la función **oom_killer** para rescatar el sistema si se presenta una OOM.

5.5. AJUSTE DE MEMORIA VIRTUAL

La memoria virtual suele ser consumida por procesos, cache de sistema de archivos y el kernel. El uso de la memoria virtual depende de un número de factores que pueden afectarse mediante los siguientes parámetros:

swappiness

Un valor de 0 a 100 que controla el punto en el que cambia el sistema. Un valor alto da prioridad al rendimiento del sistema, al intercambiar de forma agresiva los procesos de memoria física cuando no están activos. Un valor bajo da prioridad a la interacción y evita el intercambio de procesos de memoria física por el tiempo que sea posible, lo cual decrece la latencia de respuesta. El valor predeterminado es **60**.

min_free_kbytes

El número mínimo de kilobytes a mantener libres a través del sistema. Este valor sirve para computar un valor de marca de agua para cada zona de memoria baja, a la cual se le asigna un número de páginas libres reservadas, proporcional a su tamaño.

AVISO

Sea cauteloso al establecer este parámetro, puesto que si los valores son demasiado altos o demasiado bajos pueden ocasionar daños.

Si establece a ***min_free_kbytes*** demasiado lento, evitará que el sistema reclame memoria. Esto hará que el sistema se cuelgue y ocasione procesos múltiples de OOM-killing.

Sin embargo, si establece este parámetro a un valor demasiado alto (5-10 % de la memoria total del sistema) hará que la memoria de su sistema se agote inmediatamente. Linux está diseñado para usar todos los datos del sistema de archivos cache de RAM disponible. Si establece un valor de ***min_free_kbytes*** hará que el sistema consuma mucho tiempo reclamando memoria.

dirty_ratio

Define un valor de porcentaje. La escritura de datos sucios comienza (a través de **pdflush**) cuando los datos sucios comprenden este porcentaje del total de memoria del sistema. El valor predeterminado es **20**.

dirty_background_ratio

Define un valor de porcentaje. La escritura de datos sucios comienza en el segundo plano (mediante **pdflush**) cuando los datos sucios comprimen este porcentaje de memoria total. El valor predeterminado es **10**.

drop_caches

Si establece este valor a **1**, **2**, o **3** hará que el kernel envíe varias combinaciones de cache de página y de cache de plancha.

1

El sistema invalida y libera toda la memoria de cache de página.

2

El sistema libera toda la cache de plancha de memoria no utilizada

3

El sistema libera toda la cache de página y la memoria cache de plancha.

Esta es una operación no destructiva. Puesto que los objetos sucios no pueden ser liberados, se recomienda la ejecución de **sync** antes de establecer el valor de parámetro.

IMPORTANTE

El uso de ***drop_caches*** para liberar memoria no se recomienda en un entorno de producción.

Para establecer estos valores de forma temporal durante el ajuste, ejecute con el comando 'echo' el valor deseado para el archivo apropiado en el sistema de archivos proc. Por ejemplo, para establecer temporalmente ***swappiness*** a **50**, ejecute:

```
# echo 50 > /proc/sys/vm/swappiness
```

Si desea establecer de forma persistente este valor, use el comando **sysctl**. Para obtener mayor información, consulte la *Guía de implementación*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

CAPÍTULO 6. ENTRADA/SALIDA

6.1. FUNCIONALIDADES

Red Hat Enterprise Linux 6 introduce un número de mejoras de rendimiento en la pila de E/S:

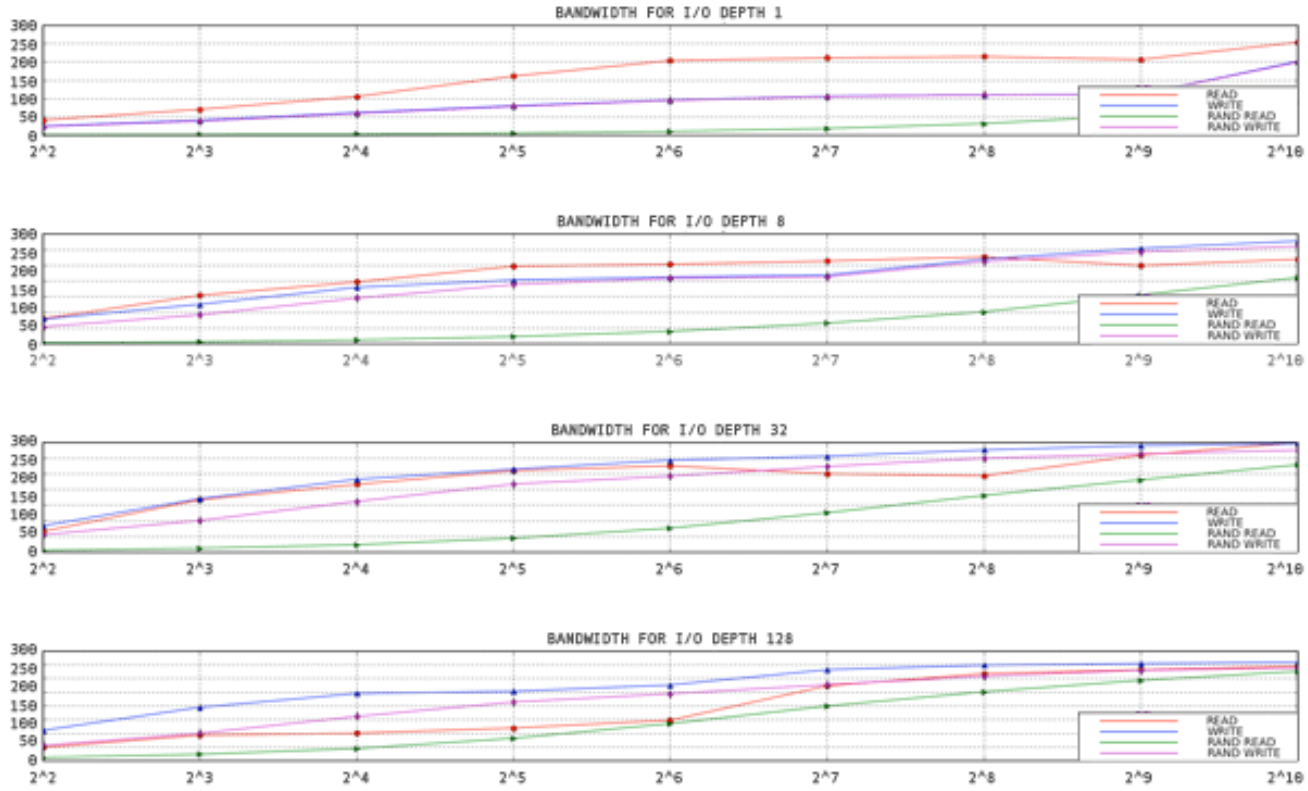
- Los discos de estado sólido (SSD) ahora se reconocen de forma automática, y el rendimiento del programador de E/S se ajusta para aprovechar el alto porcentaje de E/S por segundo (IOPS) que estos dispositivos pueden realizar.
- Se ha adicionado el soporte de descarte al kernel para reportar los rangos de bloques al almacenaje subyacente. Esto ayuda a los SSD con sus algoritmos de nivelación de uso. También ayuda a soportar el aprovisionamiento de bloques lógico (una clase de espacio de dirección virtual para almacenamiento) al mantener etiquetas más cerca en la cantidad de almacenamiento en uso.
- La implementación de la barrera del sistema de archivos ha sido revisada en Red Hat Enterprise Linux 6.1 para hacerla más funcional.
- **`pdflush`** ha sido remplazado por hilos de vaciador de dispositivos por respaldo, el cual mejora la escalabilidad del sistema en configuraciones con grandes cuentas de LUN.

6.2. ANÁLISIS

El ajuste correcto del rendimiento de pila de almacenamiento requiere un entendimiento de cómo fluyen los datos a través del sistema, como también un conocimiento profundo del almacenamiento subyacente y de cómo funciona en cargas de trabajo. También requiere un entendimiento de la carga real que se está ajustando.

Cada vez que despliega un nuevo sistema, es una buena idea perfilar el almacenaje desde la base. Inicie con los LUN o discos crudos, y evalúe su rendimiento mediante E/S directa (E/S que evita la cache de página del kernel). Esta es la prueba más básica que usted puede realizar, y será estándar por lo que mide el rendimiento de E/S en la pila. Inicie con un generador de carga de trabajo (tal como **aio-stress**) que produce lecturas secuenciales y aleatorias a través de una variedad de tamaños de E/S y profundidades de cola.

La siguiente es una gráfica de la serie de ejecuciones de **aio-stress**, cada una de las cuales realiza cuatro etapas: escritura secuencial, lectura secuencial, escritura aleatoria, y lectura aleatoria. En este ejemplo, la herramienta está configurada para ejecutarse a través de un rango de tamaños de registros (el eje x) y profundidades de cola (una por gráfica). La profundidad de cola representa el número total de operaciones de E/S en curso en un tiempo determinado.



El eje 'y' muestra el ancho de banda en MB pro segundo. El eje 'x' muestra el tamaño de E/S en kilobytes.

Figura 6.1. salida aio-stress para hilo 1, archivo 1

Observe cómo la línea de rendimiento tiende a dirigirse hacia la esquina izquierda inferior. Note también que para un determinado tamaño de registro, se puede obtener más rendimiento de almacenamiento al aumentar el número de E/S en curso.

Al ejecutar estas cargas de trabajo sencillas con su almacenamiento, entenderá cómo se realiza su almacenaje bajo carga. Retenga los datos generados por estas pruebas para comparar durante el análisis de cargas de trabajo complejas.

Si va a utilizar el mapeador de dispositivo o MD, añada esa capa a continuación y repita sus pruebas. Si hay una pérdida grande de rendimiento, asegúrese de que sea esperado o que puede explicarse. Por ejemplo, se puede esperar una caída de rendimiento si se ha añadido una capa de RAID de suma de verificación a la pila. Las caídas inesperadas de rendimiento pueden deberse a operaciones de E/S desalineadas . Red Hat Enterprise Linux alinea de forma predeterminada las particiones y metadatos de mapeador de dispositivos de forma óptima. Sin embargo, no todos los tipos de almacenaje reportan su alineación óptima y por lo tanto, pueden requerir un ajuste manual.

Después de añadir el mapeador de dispositivos o capa MD, adicione un sistema de archivos sobre el dispositivo de bloques y pruébelo con él, utilizando aún E/S directa . Compare otra vez los resultados con las pruebas anteriores y verifique si entiende bien las discrepancias. La E/S de escritura directa suele preasignar archivos, por lo tanto verifique si ha preasignado sus archivos antes de probar su rendimiento.

Entre los generadores de carga de trabajo sintética que pueden ser útiles están:

- **aio-stress**
- **iozone**
- **fio**

6.3. HERRAMIENTAS

Hay una serie de herramientas disponibles para diagnosticar los problemas de rendimiento en el subsistema de E/S. **vmstat** proporciona una visión general del rendimiento del sistemas. Las siguientes columnas son las más relevantes para E/S: ***si*** (swap in), ***so*** (swap out), ***bi*** (block in), ***bo*** (block out), y ***wa*** (I/O wait time). ***si*** y ***so*** sirven cuando su espacio swap está en el mismo dispositivo de su partición de datos y como un indicador de presión de memoria general. ***si*** y ***bi*** son operaciones de lectura, mientras que ***so*** y ***bo*** son operaciones de escritura. Cada una de estas categorías se reporta en kilobytes. ***wa*** es el tiempo inactivo; indica qué porción de la cola de ejecución se bloquea al esperar a que la E/S termine.

El análisis de su sistema con **vmstat** le dará una idea de si el subsistema de E/S es responsable o no de los problemas de rendimiento. También las columnas ***free***, ***buff***, y ***cache*** son importantes. El valor de la memoria ***cache*** que aumenta junto al valor ***bo***, seguido de una caída de la ***cache*** y un aumento en ***free*** indica que el sistema está realizando escritura diferida e invalidación de memoria cache de página.

Observe que los números de E/S reportados por **vmstat** son adiciones a todos los dispositivos de E/S. Una vez que hayan determinado que puede haber una brecha de rendimiento en el subsistema de E/S, puede examinar el problema de cerca con **iostat**, el cual dividirá el reporte de E/S por dispositivo. También puede recuperar un información más detallada, tal como promedio de tamaño de solicitud, el número de lectura y escrituras por segundo y la cantidad de E/S en fusión que en el momento.

Al usar el tamaño de petición y cola promedios (**avgqu-sz**), podrá estimar cómo realizar el almacenaje mediante gráficas que genera y al caracterizar el rendimiento de su almacenaje. Algunas generalizaciones aplican: por ejemplo, si el tamaño de solicitud promedio es de 4KB y el tamaño de cola es 1, será poco probable que el rendimiento sea muy efectivo.

Si los números de rendimiento no coinciden con el rendimiento esperado, realice un análisis minucioso con **blktrace**. El paquete de herramientas **blktrace** ofrece información detallada sobre el tiempo que gasta en el subsistema de E/S. La salida de **blktrace** es una serie de archivos de trazado binarios que pueden ser posprocesados por otras herramientas tales como **blkparse**.

blkparse es la herramienta compañera de **blktrace**. Lee la salida cruda del trazado y produce una versión textual abreviada.

El siguiente es un ejemplo de salida de **blktrace**:

```
8,64   3        1     0.000000000  4162  Q  RM 73992 + 8 [fs_mark]
8,64   3        0     0.000012707     0  m   N cfq4162S / alloced
8,64   3        2     0.000013433  4162  G  RM 73992 + 8 [fs_mark]
8,64   3        3     0.000015813  4162  P   N [fs_mark]
8,64   3        4     0.000017347  4162  I   R 73992 + 8 [fs_mark]
8,64   3        0     0.000018632     0  m   N cfq4162S / insert_request
8,64   3        0     0.000019655     0  m   N cfq4162S / add_to_rr
8,64   3        0     0.000021945     0  m   N cfq4162S / idle=0
8,64   3        5     0.000023460  4162  U   N [fs_mark] 1
8,64   3        0     0.000025761     0  m   N cfq workload slice:300
8,64   3        0     0.000027137     0  m   N cfq4162S / set_active
wl_prio:0 wl_type:2
8,64   3        0     0.000028588     0  m   N cfq4162S / fifo=(null)
8,64   3        0     0.000029468     0  m   N cfq4162S / dispatch_insert
8,64   3        0     0.000031359     0  m   N cfq4162S / dispatched a
request
8,64   3        0     0.000032306     0  m   N cfq4162S / activate rq,
```

```
drv=1
8,64   3        6     0.000032735  4162  D   R 73992 + 8 [fs_mark]
8,64   1        1     0.004276637     0  C   R 73992 + 8 [0]
```

Como puede ver, la salida es densa y difícil de leer. Puede decir qué procesos son responsables de emitir E/S a su dispositivo, lo cual es útil, pero **blkparse** puede proporcionar información adicional en un formato fácil de entender en su resumen. La información de resumen **blkparse** se imprime al final de la salida:

```
Total (sde):
Reads Queued:          19,       76KiB  Writes Queued:     142,183,
568,732KiB
Read Dispatches:       19,       76KiB  Write Dispatches:   25,440,
568,732KiB
Reads Requeued:         0               Writes Requeued:       125
Reads Completed:       19,       76KiB  Writes Completed:   25,315,
568,732KiB
Read Merges:            0,        0KiB  Write Merges:      116,868,
467,472KiB
IO unplugs:        20,087               Timer unplugs:           0
```

El resumen muestra las tasas de E/S promedio, la actividad de fusión y compara la carga de lectura con la carga de trabajo de escritura. Sin embargo, para la mayoría, la salida de **blkparse** es demasiado voluminosa. Afortunadamente, hay varias herramientas que ayudan a visualizar los datos.

btt proporciona un análisis de la cantidad de tiempo de E/S utilizado en las diferentes áreas de la pila de E/S. Dichas áreas son:

- Q — Una E/S de bloque está en cola
- G — Obtener solicitud

  Una nueva E/S de bloque en cola no era candidata para fusionar con ninguna solicitud existente, por lo tanto se asigna una nueva petición de capa de bloque.
- M — Una E/S de bloque se fusiona con una solicitud existente.
- I — Una solicitud se inserta dentro de la cola de dispositivo.
- D — Una solicitud se expide al Dispositivo.
- C — Una solicitud es completada por el controlador.
- P — La cola de dispositivo se tapona para permitir que las solicitudes se acumulen.
- U — La cola de dispositivo se destapona para permitir que las solicitudes acumuladas se envíen al dispositivo.

btt no solamente divide el tiempo utilizado en cada una de las áreas, sino también el tiempo de transición utilizado entre ellas, al igual que:

- Q2Q — tiempo entre solicitudes enviadas a la capa de bloques
- Q2G — tiempo que se tarda desde el momento que una E/S de bloque es puesta en cola hasta el tiempo que obtiene una solicitud asignada.

- G2I — tiempo que se tarda desde que se asigna una solicitud al tiempo que se inserta en la cola del dispositivo.
- Q2M — tiempo que se tarda desde que E/S de un bloque es puesta en cola al tiempo que se fusiona con la solicitud existente.
- I2D — tiempo que se tarda desde que se inserta una solicitud en un dispositivo al tiempo que se emite al dispositivo.
- M2D — tiempo que se tarda desde la fusión de una E/S de bloque con la solicitud de salida hasta la emisión de la solicitud al dispositivo
- D2C — tiempo de servicio de la solicitud por dispositivo
- Q2C — tiempo total utilizado en la capa de bloques para una solicitud

Puede reducir bastante una carga de trabajo de la tabla anterior. Por ejemplo, si Q2Q es mucho mayor que Q2C, significará que la aplicación no está emitiendo una E/S en una rápida sucesión. De esta manera, cualquier problema de rendimiento que se presente no estará relacionado de ninguna manera con el subsistema de E/S. Si D2C es muy alto, entonces significará que el dispositivo se está tardando mucho para servir las solicitudes. Esto puede indicar que el dispositivo esta sobrecargado (lo cual puede ser debido a un recurso compartido), o puede ser debido a que la carga enviada al dispositivo es subóptima. Si Q2G es muy alto, significa que hay muchas solicitudes en cola al mismo tiempo. Esto puede indicar que el almacenamiento no puede sostener la carga de E/S.

Por último, **Seekwatcher** consume los datos binarios de **blktrace** y genera un grupo de gráficos, incluidos Dirección de bloques lógicos (LBA), rendimiento, búsquedas por segundo, y Operaciones de E/S por segundo (IOPS).

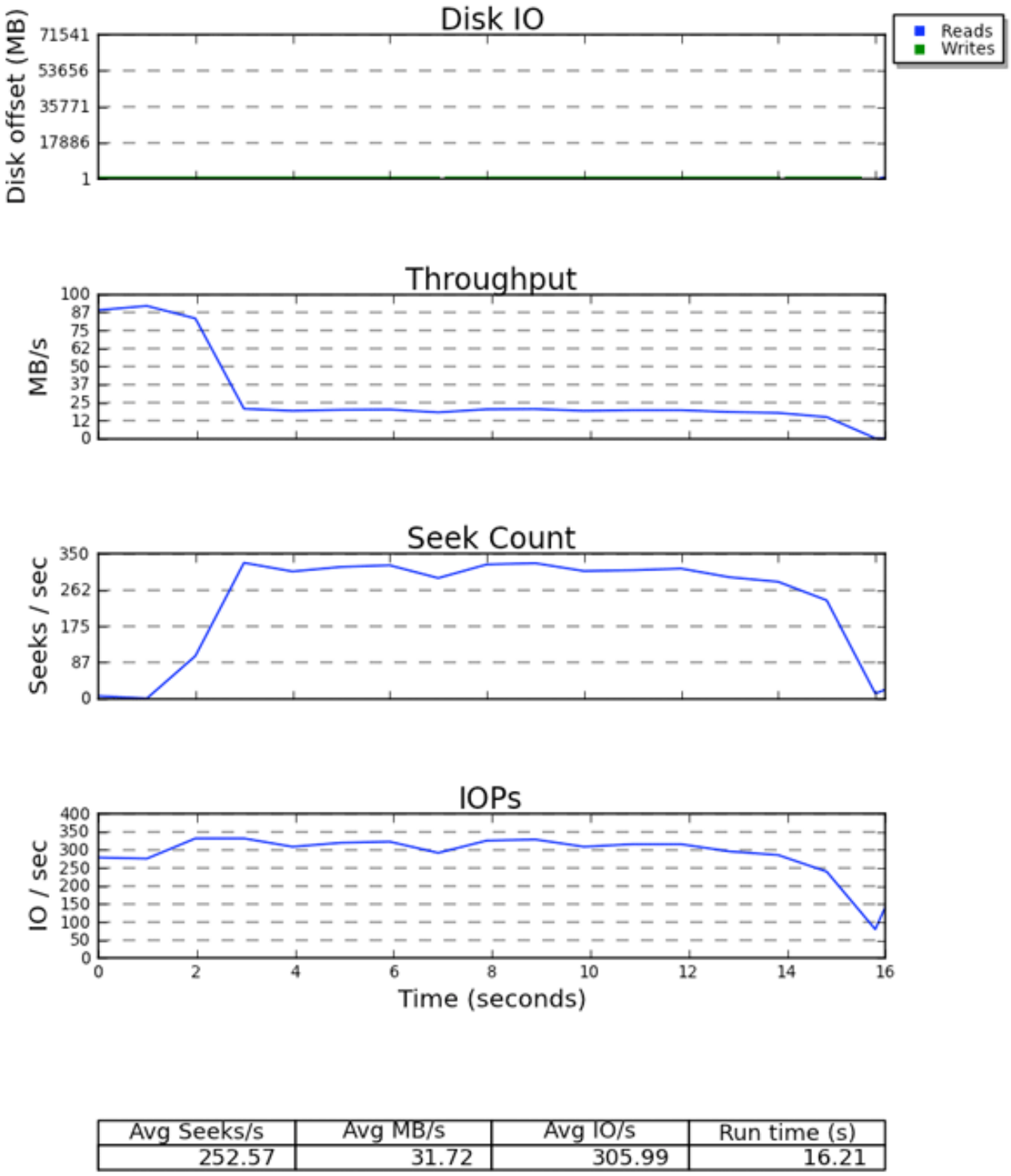


Figura 6.2. Ejemplo de salida de Seekwatcher

Todos los gráficos usan el tiempo como eje X. El gráfico LBA muestra y escribe en diferentes colores. Es interesante observar la relación entre el rendimiento y las gráficas de búsqueda por segundo. Para almacenamiento que es sensible a búsquedas, hay una relación inversa entre los dos gráficos. El gráfico IOPS es útil si por ejemplo, no se está obteniendo el rendimiento esperado de un dispositivo, pero está llegando a sus limitaciones de IOPS.

6.4. CONFIGURACIÓN

Una de las primeras decisiones que deberá tomar, será decidir qué programador de E/S va a utilizar. Esta sección proporciona una visión general de cada uno de los programadores principales para ayudarle a decidir cuál es el mejor para su carga de trabajo.

6.4.1. Cola de reparto justo (CFQ)

CFQ intenta ser algo justa al programar las decisiones de E/S con base en el proceso que inició la E/S. Hay tres clases diferentes de programación provistas: tiempo-real (RT), mejor-esfuerzo (BE), e inactiva. Una clase de programación puede ser asignada de forma manual a un proceso con el comando **`ionice`** o asignada de forma programática con la llamada del sistema **`ioprio_set`**. Los procesos se sitúan de forma predeterminada en la clase de programación de mejor-esfuerzo. Las clases de programación de tiempo-real y el mejor-esfuerzo se subdividen en ocho prioridades de E/S dentro de cada clase, la prioridad superior es 0 y la prioridad inferior es 7. Los procesos en la clase de programación del tiempo se programan de una forma más agresiva que los procesos de mejor-esfuerzo e inactivos, por lo tanto cualquier E/S en tiempo-real programada siempre se realizará antes de E/S de mejor-esfuerzo o inactiva. Esto significa que la prioridad de E/S de tiempo-real no otorgará recursos a las clases de mejor-esfuerzo e inactivas. El mejor-esfuerzo de programación es la clase de programación predeterminada y 4 es la prioridad predeterminada dentro de esta clase. Los estados en clase de programación inactiva solo se sirven cuando no hay otra E/S pendiente en el sistema. Por lo tanto, es muy importante establecer a inactiva únicamente la clase de programación de E/S de un proceso, si E/S del proceso no se requiere de ninguna manera para continuar.

CFQ es justa al asignar una porción del tiempo a cada uno de los procesos de E/S. Durante su porción de tiempo, un proceso puede tener (de forma predeterminada) hasta 8 solicitudes al aire. El programador trata de anticipar, con base en los datos históricos, si una aplicación entregará más E/S en un futuro cercano. Si se espera que un proceso entregue más E/S, entonces CFQ estará inactiva, esperando dicha E/S, incluso si hay otros procesos de E/S que esperan ser entregados.

Debido a la inactivación realizada por CFQ, no suele ser un buen ajuste para hardware que no sufre una multa de larga búsqueda, tal como matrices de almacenamiento externo rápidas o discos en estado sólido. Si se requiere usar CFQ en dicho almacenamiento (por ejemplo, si desea usar el programador de E/S de peso proporcional de cgroup), deberá ajustar algunos parámetros para mejorar el rendimiento de CFQ. Establezca los siguientes parámetros en los archivos del mismo nombre localizados en **`/sys/block/`*`dispositivo`*`/queue/iosched/`**:

```
slice_idle = 0
quantum = 64
group_idle = 1
```

Cuando se establece ***`group_idle`*** a 1, aún existe el potencial para paradas de E/S (donde el almacenamiento de back-end no esté ocupado debido a inactividad). Sin embargo, estas paradas serán menos frecuentes que inactivas en cada cola en el sistema.

CFQ no es un programador de E/S de conservación de no-trabajo, lo cual significa que puede estar inactivo incluso cuando hay solicitudes pendientes. El apilamiento de programadores de conservación de no-trabajo puede introducir varias latencias de ruta de E/S. Un ejemplo de cada pila es el uso de CFQ por encima de un RAID de hardware de host. El controlador de RAID puede implementar su propio controlador de conservación de no-trabajo, causando así demoras en dos niveles de la pila. Los programadores de conservación de no-trabajo operan mejor cuando tienen tantos datos como sea posible para fundamentar sus decisiones. En el caso de apilamiento de dichos algoritmos de programación, el programador que está más al fondo, solamente verá lo que el programador superior envía abajo. Por lo tanto, la capa inferior verá un patrón de E/S que no representa del todo la carga de trabajo real.

Ajustables

back_seek_max

Las búsquedas en retroceso no suelen tener un buen rendimiento, ya que incurren en mayor demora que las búsquedas anticipadas al reubicar las cabezas. Sin embargo, CFQ las realizará, si son lo suficientemente pequeñas. Este ajustable controla la distancia máxima en KB que el programador de E/S permitirá en búsquedas en retroceso. La predeterminada es **16** KB.

back_seek_penalty

Debido a la ineficacia de búsquedas en retroceso, la multa se asocia con cada una de ellas. La multa es un multiplicador; por ejemplo, considere una cabeza de disco en una posición de 1024KB. Asuma que hay dos solicitudes en la cola, una a 1008KB y otra a 1040KB. Las dos solicitudes son equidistantes a la posición de la cabeza actual. Sin embargo, después de aplicar la multa de búsqueda en retroceso (predeterminada: 2), la solicitud a una posición posterior en disco estará dos veces más cerca de la solicitud anterior. Por lo tanto, la cabeza avanzará.

fifo_expire_async

Este ajustable controla el tiempo que lleva una solicitud async sin servicio. Después del tiempo de expiración, (en milisegundos), una solicitud async sin servicio será trasladada a la lista de envío. El predeterminado es **250** ms.

fifo_expire_sync

Es el mismo ajustable fifo_expire_async, para solicitudes en sincronía (lectura y escritura O_DIRECT). El predeterminado es **125** ms.

group_idle

Si está configurado, CFQ se inactiva en el último proceso entregando E/S en un cgroup. Se debe configurar a **1** cuando se usan cgroups de E/S de peso proporcional y se establece ***slice_idle*** a **0** (típicamente se realiza en almacenamiento rápido).

group_isolation

Si el aislamiento de grupo se activa (establece a **1**), proporciona un aislamiento mayor entre grupos en gasto de rendimiento. Por lo general, si el aislamiento del grupo se desactiva, se hace justicia para las cargas de trabajo secuenciales únicamente. Al habilitar el aislamiento de grupos se hace justicia tanto a las cargas de trabajo secuenciales como a las aleatorias. El valor predeterminado es **0** (inhabilitado). Consulte **Documentation/cgroups/blkio-controller.txt** para obtener mayor información.

low_latency

Cuando la latencia baja está habilitada (**1**), CFQ intenta proporcionar un tiempo máximo de espera de 300 ms para cada proceso que emite E/S en un dispositivo. Esto favorece la justicia en el rendimiento. Al inhabilitar la latencia baja (**0**) se ignora la latencia de destino, lo cual permite que cada proceso en el sistema obtenga una porción de tiempo completo. La latencia baja es la predeterminada.

quantum

El quantum controla la cantidad de E/S que CFQ enviará al almacenaje al mismo tiempo, esencialmente la profundidad de cola de dispositivo que se predetermina a **8**. El almacenamiento puede soportar mucha más profundidad de cola, pero el aumento de ***quantum*** también tendrá un impacto negativo en la latencia, sobre todo en la presencia de grandes cargas de trabajo de escritura secuenciales.

slice_async

Este ajustable controla el tiempo asignado a cada proceso que se realiza E/S asíncrona (escritura en búfer). Por defecto, se predetermina a **40** ms.

slice_idle

Especifica el tiempo que CFQ debe estar inactivo mientras espera las siguientes solicitudes. El valor predeterminado en Red Hat Enterprise Linux 6.1 y anterior es **8** ms. En Red Hat Enterprise Linux 6.2 y posterior, el valor predeterminado es **0**. El valor de cero mejora el rendimiento de almacenaje de RAID externo al retirar todo lo inactivo a nivel de cola y árbol de servicios. Sin embargo, un valor de cero puede degradar el rendimiento en almacenamiento interno non-RAID, puesto que aumenta el número total de búsquedas. Para almacenaje non-RAID, recomendamos que el valor de ***slice_idle*** sea mayor que 0.

slice_sync

Este ajustable dicta la porción del tiempo asignado al proceso que se emite E/S en sincronía (lectura o escritura directa). La predeterminada es **100** ms.

6.4.2. Programador de tiempo límite de E/S

Programador de tiempo límite de E/S intenta proporcionar latencia garantizada para solicitudes. Es importante anotar que la medida de latencia únicamente inicia cuando las solicitudes llegan al programador de E/S (es una diferencia importante , ya que una aplicación puede poner a dormir o a esperar a los descriptores de solicitudes que van a liberarse). Por defecto, las lecturas tienen prioridad sobre las escrituras, puesto que es más probable que las aplicaciones se bloqueen en la lectura de E/S

El tiempo límite envía E/S en lotes. Un lote es una secuencia de E/S de lectura o escritura que aumenta el orden de LBA. Después de procesar cada lote, el programador de E/S verifica si las solicitudes de escritura han dejado de recibir los servicios por demasiado tiempo y luego decide si inicia o no un nuevo lote de lecturas y escrituras. La lista de solicitudes FIFO (Primero en entrar, primero en salir), únicamente es revisada en el inicio de cada lote y luego, en la dirección de ese lote. Por lo tanto, si se selecciona un lote de escritura y hay una solicitud de lectura que ya ha expirado, esa solicitud de lectura no será servida sino hasta que el lote termine.

Ajustables

fifo_batch

Determina el número de lectura y escritura a emitir en un lote sencillo. El predeterminado es **16**. Un valor más alto, puede producir un mejor rendimiento, pero también un aumento en la latencia.

front_merges

Puede establecer este ajustable a **0** si sabe que su carga de trabajo nunca generará fusiones frontales. A menos que haya medido la sobrecarga de esta revisión, es aconsejable dejar el parámetro predeterminado: **1**.

read_expire

Este ajustable le permite establecer el número en milisegundos, en el cual una solicitud de lectura debe ser servida. El predeterminado es **500** ms (medio segundo).

write_expire

Este ajustable le permite establecer el número en milisegundos, en el cual una solicitud de lectura debe ser servida. El predeterminado es **5000** ms (cinco segundos).

writes_starved

Este ajustable controla cuántos lotes de lectura pueden ser procesados antes de procesar un lote de escritura individual. Entre más alto se establezca, más preferencia se dará a las lecturas.

6.4.3. Noop

Programador Noop de E/S implementa un algoritmo de programación simple 'primero en entrar, primero en salir' (FIFO). La fusión de solicitudes sucede en la capa de bloque genérica, pero es una cache sencilla de 'último-golpe'. Si un sistema está vinculado a una CPU y el almacenaje es rápido, puede ser el mejor programador de E/S a utilizar.

A continuación se presentan los ajustables disponibles para la capa de bloques.

Ajustables /sys/block/sdX/queue

add_random

En algunos casos, los eventos de sobrecarga de E/S que contribuyen al grupo entrópico para **/dev/random** es medible. Algunas veces puede establecerse a un valor de 0.

max_sectors_kb

El tamaño de solicitud máximo predeterminado enviado a disco es **512** KB. Este ajustable sirve para aumentar o disminuir dicho valor. El valor mínimo está limitado por el tamaño de bloque lógico y el valor máximo está limitado por ***max_hw_sectors_kb***. Hay algunos SSD que funcionan peor cuando los tamaños de E/S exceden el tamaño de bloque de borrado interno. En estos casos, se recomienda ajustar ***max_hw_sectors_kb*** al tamaño de bloque de borrado. Puede probarlo con un generador de E/S tal como **iozone** o **aio-stress**, variando el tamaño de registro de **512** bytes a **1** MB, por ejemplo.

nomerges

Este ajustable es principalmente una ayuda de depuración. La mayoría de cargas de trabajo se benefician de la fusión de solicitudes (incluso en un almacenamiento más rápido tal como SSD). En algunos casos, sin embargo, es deseable desactivar la fusión, como cuando se desee ver cuántos IOPS puede procesar un back-end de almacenamiento sin desactivar la lectura anticipada o realizar E/S aleatoria.

nr_requests

Cada cola de solicitud tiene un límite en el total de solicitudes de descriptores que pueden asignarse para cada E/S de lectura y escritura . El número predeterminado es **128**, es decir 128 lecturas y 128 escrituras pueden ser puestas en cola a la vez antes de poner a dormir el proceso. El proceso que fue puesto a dormir es el siguiente que intenta asignar una solicitud, no necesariamente el haya asignado todas las solicitudes disponibles.

Si tiene una aplicación sensible de latencia, debe considerar disminuir el valor de ***nr_requests*** en su cola de solicitudes y limitar la profundidad de cola de comandos en el almacenaje a un número bajo (incluso tan bajo como **1**), así, la retro-escritura de E/S no puede asignar todos los descriptores de solicitudes disponibles y llenar la cola de dispositivo con E/S de escritura. Una vez se haya asignado ***nr_requests***, todos los procesos que intentan realizar E/S serán puestos a dormir para esperar que las solicitudes estén disponibles. Esto hace que las cosas sean justas, ya que las

solicitudes se distribuyen en round-robin (en lugar de permitir a un solo proceso consumirlos todos en una rápida sucesión). Observe que solamente es un problema cuando se usan los programadores de tiempo límite o Noop, ya que el CFQ predeterminado protege contra esta situación.

optimal_io_size

En algunas circunstancias, el almacenamiento subyacente reportará un tamaño de E/S óptimo. Esto es más común en RAID de hardware y software, donde el tamaño de E/S óptimo es el tamaño de banda. Si este valor se reporta, las aplicaciones deberán emitir E/S alineada y en múltiples del tamaño óptimo de E/S siempre y cuando sea posible.

read_ahead_kb

El sistema operativo puede detectar cuándo una aplicación está leyendo datos en forma secuencial desde un archivo o un disco. En dichos casos, realiza un algoritmo de lectura anticipada inteligente, donde se leen más datos en el disco que los solicitados por el usuario. Así, cuando el usuario intenta leer un bloque de datos, ya habrá puesto en memoria cache de página del sistema operativo. La desventaja en potencia es que el sistema operativo puede leer más datos que los necesarios, lo cual ocupa espacio en la cache de página hasta que es desalojada debido a una presión de alta memoria. Al tener múltiples procesos haciendo una lectura anticipada, aumentaría la presión de memoria en estas circunstancias.

Para dispositivos de mapeador de dispositivos, suele ser una buena idea aumentar el valor de ***read_ahead_kb*** a un número más grande, tal como **8192**. La razón es que ese dispositivo de mapeador de dispositivos suele componerse de varias dispositivos subyacentes. Establecer este parámetro al predeterminado (**128** KB) multiplicado por el número de dispositivos que usted está asignando es un buen comienzo para el ajuste.

rotational

Los discos duros tradicionales han sido rotatorios (compuestos por bandejas rotatorias). No obstante, los SSD, no lo son. La mayoría de SSD lo publicará adecuadamente. Sin embargo, si se encuentra un dispositivo que no publique este indicador adecuadamente, deberá establecer, de forma manual, el rotatorio a **0**; cuando el rotatorio esté desactivado, el elevador de E/S no emplea la lógica que se espera para reducir búsquedas, puesto que hay una pequeña multa para las operaciones de búsqueda en medios que no son rotatorios.

rq_affinity

La afinación de E/S puede ser procesada en una CPU diferente a la que expidió la E/S. La configuración de ***rq_affinity*** a **1** hace que el kernel entregue ajustes a la CPU en la cual se emitió E/S. Esto puede mejorar la efectividad de puesta en cache de datos de CPU.

CAPÍTULO 7. SISTEMAS DE ARCHIVOS

Por favor lea este capítulo que muestra los sistemas de archivos que tienen soporte al ser utilizados con Red Hat Enterprise Linux, y cómo optimizar su rendimiento.

7.1. CONSIDERACIONES DE AJUSTE PARA SISTEMAS DE ARCHIVOS

Hay varias consideraciones de ajuste comunes a todos los sistemas de archivos: opciones de formato y montaje seleccionadas en su sistema, y acciones disponibles para aplicaciones que pueden mejorar su rendimiento en un sistema determinado.

7.1.1. Opciones de formateo

Tamaño de bloques de sistema de archivos

El tamaño de bloque puede seleccionarse en tiempo de **mkfs**. El rango de dimensiones válidas depende del sistema: el límite superior es la dimensión de página máxima del sistema de host, mientras que el límite inferior depende del sistema de archivos utilizado. El tamaño de bloque predeterminado en el apropiado en la mayoría de los casos.

Si espera crear varios archivos más pequeños que el bloque predeterminado, establezca un tamaño de bloque más pequeño para minimizar la cantidad de espacio desperdiciado en el disco. Observe, sin embargo, que la configuración de un tamaño de bloque más pequeño puede ocasionar sobrecarga adicional de tiempo de ejecución, en particular para archivos más grandes que el tamaño de bloque seleccionado.

Geometría de sistemas de archivos

Si su sistema usa un almacenaje en banda tal como RAID5, puede mejorar el rendimiento al alinear los datos y metadatos con la geometría de almacenamiento subyacente en tiempo **mkfs**. Para RAID por software (LVM o MD) y algún almacenaje de hardware empresarial, esta información se solicita y establece automáticamente, pero en muchos casos el administrador debe especificar de forma manual dicha geometría con **mkfs** en la línea de comandos.

Para obtener mayor información sobre cómo crear y mantener estos sistemas de archivos, consulte la *Guía de administración de almacenamiento*.

Diarios externos

Las cargas de trabajo de metadatos intensivos significan que la sección de registro de un sistema de archivos de diario (tal como un ext4 y XFS) se actualiza con mucha frecuencia. Para minimizar el tiempo de búsqueda de un sistema de archivos a un diario, coloque el diario en un almacenaje dedicado. No obstante, observe que al colocar el diario en almacenaje externo que sea más lento que en el sistema de archivos primario, anulará cualquier ventaja potencial asociada al uso de almacenaje externo.

AVISO

Verifique si su diario externo es confiable. La pérdida de un dispositivo de diario externo dañará el sistema de archivos.

Los diarios externos se crean en tiempo **mkfs** con dispositivos de diario que se especifican en tiempo de montaje. Consulte las páginas de manual **mke2fs(8)**, **mkfs.xfs(8)**, y **mount(8)** para obtener mayor información.

7.1.2. Opciones de montaje

Barreras

Una barrera de escritura es un mecanismo de kernel usado para verificar si los metadatos del sistema de archivos están escritos correctamente y ordenados en almacenaje persistente, incluso cuando los dispositivos de almacenaje con memorias caches volátiles pierden energía. Los sistemas de archivos con barreras habilitadas también se aseguran de que los datos transmitidos a través de **fsync()** persistan a través de pérdidas de energía. Red Hat Enterprise Linux habilita barreras predeterminadas en todo el hardware que las soportan.

Sin embargo, al habilitar barreras de escritura algunas aplicaciones se retrasan demasiado, en especial, las aplicaciones que usan bastante **fsync()** o crean y borran muchos archivos pequeños. Para almacenaje sin cache de escritura no volátil o en un caso raro en el que las inconsistencias de sistemas de archivos y pérdida de datos después de una pérdida de energía sea aceptable, se pueden desactivar las barreras con la opción de montaje ***nobarrier***. Para obtener mayor información, consulte la *Guía de administración de almacenamiento*.

Hora de acceso (noatime)

Anteriormente, cuando el archivo era leído, el tiempo de acceso (***atime***) para ese archivo debía ser actualizado en los metadatos del inodo, lo cual implica E/S de escritura adicional. Si no se requieren los metadatos exactos de ***atime***, monte el sistema de archivos con la opción ***noatime*** para eliminar estas actualizaciones de metadatos. En la mayoría de los casos, sin embargo,***atime*** no es una gasto debido al atime predeterminado relativo (o la conducta ***relatime***) en el kernel de Red Hat Enterprise Linux 6. La conducta de ***relatime*** solamente actualiza ***atime*** si el ***atime*** anterior es mayor que el tiempo de modificación (***mtime***) o el tiempo de cambio de estatus (***ctime***).

NOTA

Al habilitar la opción ***noatime*** también se habilita la conducta ***nodiratime***; no hay necesidad de establecer ***noatime*** y ***nodiratime***.

Incremento en soporte de lectura anticipada

El acceso al archivo de lectura anticipada se agiliza al pre-obtener datos y cargarlos en la memoria cache de la página para que puedan estar disponibles más temprano en memoria en lugar de en disco. Algunas cargas de trabajo, tales como las que implican flujos pesados de E/S, se benefician de los valores de lectura anticipada.

La herramienta **tuned** y el uso de las franjas de LVM elevan el valor de lectura- anticipada, sin embargo esto no siempre es suficiente para algunas cargas de trabajo. Además, Red Hat Enterprise Linux no siempre puede establecer un valor de lectura anticipada sobre lo que puede detectar de sus sistema de archivos. Por ejemplo, si una matriz de almacenamiento se presenta a Red Hat Enterprise Linux como un LUN poderoso, el sistema operativo no siempre la tratará como una matriz de LUN poderosa, por lo tanto, no hará usará de forma predeterminada todas las ventajas de lectura anticipada disponibles para el almacenamiento .

Use el comando **blockdev** para ver y modificar el valor de lectura anticipada. Para ver el valor de lectura anticipada actual para un dispositivo de bloques particular, ejecute:

```
# blockdev -getra dispositivo
```

Para modificar el valor de lectura anticipada para dicho dispositivo de bloque, ejecute el siguiente comando. *N* representa el número de sectores de 512-bytes.

```
# blockdev -setra N dispositivo
```

Observe que el valor seleccionado con el comando **blockdev** no persistirá entre inicios. Le recomendamos crear un nivel de ejecución de script **init.d** para establecer este valor durante el inicio.

7.1.3. Mantenimiento de sistema de archivos

Descartar bloques no utilizados

El descarte de lotes y las operaciones de descarte son funcionalidades de sistemas de archivos montados que descartan los bloques que el sistema de archivos no está utilizando. Estas operaciones sirven tanto para unidades de estado sólido como de almacenamiento finamente-aprovisionado.

Las *operaciones de descarte de lote* son ejecutadas de forma explícita por el usuario mediante el comando **fstrim**. Este comando descarta todos los bloques no utilizados en un sistema de archivos coincidente con los criterios del usuario. Ambos tipos de operaciones están soportados para usar con los sistemas de archivos XFS y ext4 en Red Hat Enterprise Linux 6.2 y posteriores junto con el dispositivo de bloque subyacente soporta las operaciones de descarte físico. Las operaciones de descarte físico están soportadas si el valor de **/sys/block/*device*/queue/discard_max_bytes** no es cero.

Las *operaciones de descarte en línea* se especifican en el momento de montaje con la opción **-o discard** (ya sea en **/etc/fstab** o como parte del comando **mount**), y se ejecutan en tiempo real sin intervención del usuario. Las operaciones de descarte en línea solamente descartan bloques que están en transición de usados a libres. Las operaciones de descarte están soportadas en sistemas de archivos ext4 en Red Hat Enterprise Linux 6.2 y posteriores, y en sistemas de archivos XFS en Red Hat Enterprise Linux 6.4 y posteriores.

Red Hat recomienda las operaciones de descarte a menos que la carga de trabajo del sistema sea tal que el descarte de lote no sea factible o que las operaciones de descarte en línea se necesiten para mantener el rendimiento.

7.1.4. Consideraciones de aplicaciones

Pre-asignación

Los sistemas de archivos ext4, XFS, y GFS2 soportan la pre-asignación de espacio eficiente mediante llamada glib de **fallocate(2)**. En los casos en que los archivos sean fragmentados erróneamente debido a patrones de escritura, que conducen a un pobre rendimiento de lectura, la pre-asignación de espacio puede ser una técnica útil. La pre-asignación de espacio marca el espacio de disco como si hubiese sido asignado a un archivo, sin escribir ningún dato dentro de dicho espacio. Mientras que no se haya escrito nada en un bloque pre-asignado, las operaciones de lectura retornarán ceros.

7.2. PERFILES PARA RENDIMIENTO DE SISTEMA DE ARCHIVOS

La herramienta **tuned-adm** permite a los usuarios el fácil intercambio de un número de perfiles que han sido diseñados para mejorar el rendimiento en casos específicos. Los perfiles que son en particular útiles para mejorar el rendimiento son:

latency-performance

Un perfil de servidores para ajuste de rendimiento de latencia típico. Desactiva los mecanismos de ahorro de energía de **tuned** y **ktune**. El modo de **cpuspeed** cambia a **rendimiento**. El elevador de E/S cambia a **fecha límite** para cada dispositivo. El parámetro ***cpu_dma_latency*** se registra con un valor de **0** (la latencia más baja posible) para que el servicio de calidad de administración de energía limite la latencia cuando sea posible.

throughput-performance

Un perfil de servidor para ajuste de rendimiento típico. Este perfil se recomienda si el sistema no tiene almacenamiento de clase empresarial. Es igual a **latency-performance**, excepto:

- ***kernel.sched_min_granularity_ns*** (programador granularidad de preferencia mínima) se establece a **10** milisegundos,
- ***kernel.sched_wakeup_granularity_ns*** (programador de granularidad de despertador) se establece a **15** milisegundos,
- ***vm.dirty_ratio*** (relación sucia de máquina virtual) se establece a 40%, y
- las páginas gigantes transparentes se activan.

enterprise-storage

Este perfil se recomienda para configuraciones de servidor de tamaño empresarial con almacenamiento de clase empresarial, que incluye protección de cache y administración de controlador de batería de respaldo de cache en disco. Es igual que el perfil de **throughput-performance**, excepto:

- el valor ***readahead*** se establece a **4x**, y
- los sistemas de archivos no root/boot se remontan con **barrier=0**.

Para obtener mayor información sobre **tuned-adm** consulte la página de manual (**man tuned-adm**), o la *Guía de administración de energía* disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

7.3. SISTEMAS DE ARCHIVOS

7.3.1. El sistema de archivos Ext4

El sistema de archivos ext4 es una extensión escalable del sistema de archivos predeterminado ext3 disponible en Red Hat Enterprise Linux 5. Ext4 es ahora el sistema de archivos predeterminado para Red Hat Enterprise Linux 6, y está soportado para un tamaño máximo de sistema de archivos de 16 TB y un tamaño máximo de un archivo individual de 16TB. También retira el límite de subdirectorio de 32000 presente en ext3.

NOTA

Para sistemas de archivos superiores a 16TB, recomendamos usar un sistema de archivos de alta capacidad escalable tales como XFS. Para obtener mayor información, por favor consulte la Sección 7.3.2, “Sistema de archivos XFS”.

Los predeterminados del sistema de archivos ext4 son óptimos para la mayoría de las cargas de trabajo, pero si el análisis de rendimiento muestra que la conducta del sistema de archivos está impactando el rendimiento, hay varias opciones de ajuste disponibles:

Inicialización de tabla de inodo

Para sistemas de archivos muy grandes, el proceso de **mkfs.ext4** puede tardarse en inicializar todas las tablas de inodo en el sistema de archivos. Este proceso puede diferirse con la opción ***-E lazy_itable_init=1***. Al utilizar esta opción, los procesos de kernel continuarán inicializando el sistema de archivos después de que es montado. La tasa en la cual se presenta esta inicialización, puede controlarse con la opción ***-o init_itable=n*** para el comando **mount**, donde la cantidad de tiempo utilizado realizando esta inicialización de fondo es aproximadamente 1/n. El valor predeterminado para **n** es **10**.

Conducta de Auto-fsync

Debido a que algunas aplicaciones no siempre se sincronizan correctamente con **fsync()** después de renombrar un archivo existente o truncar y rescribir, ext4 se predetermina para sincronizar automáticamente los archivos después de las operaciones 'replace-via-rename' y 'replace-via-truncate'. Esta conducta es bastante consistente con la conducta anterior del sistema de archivos ext3. Sin embargo, las operaciones **fsync()** pueden consumir bastante tiempo, por lo tanto si esta conducta automática no se requiere, use la opción ***-o noauto_da_alloc*** con el comando **mount** para desactivarla. Esto significará que la aplicación debe usar explícitamente **fsync()** para garantizar la persistencia de datos.

Prioridad de E/S de diario

El envío de datos de E/S de diario predeterminados recibe una leve prioridad que ***journal_ioprio=n*** del comando **mount**. El valor predeterminado es **3**. El rango de valores válidos va de 0 a 7, siendo 0 la E/S de prioridad más alta.

Para obtener otras opciones de **mkfs** y ajuste, por favor consulte las páginas de manual **mkfs.ext4(8)** y **mount(8)**, como también el archivo **Documentation/filesystems/ext4.txt** en el paquete kernel-doc.

7.3.2. Sistema de archivos XFS

XFS es un sistema de archivos de diario robusto y de alta escalabilidad de 64 bits. Este sistemas se basa totalmente en la extensión, por lo tanto soporta grandes archivos y sistemas de archivos muy grandes. El número de archivos que pueden contener un sistema XFS está limitado únicamente por el espacio disponible en el sistema de archivos.

XFS soporta diarios de metadatos, lo cual facilita la recuperación rápida de caídas. Los sistemas de archivos XFS también pueden desfragmentarse y expandirse mientras estén montados y activos. Además, Red Hat Enterprise Linux 6 soporta copia de seguridad y restaura herramientas específicas para XFS.

XFS usa asignación de extensión y presenta un números de esquemas de asignación tales como asignación y pre-asignación explícita. La asignación explícita proporciona un método más compacto y método eficiente para rastrear el espacio utilizado en un sistema de archivos, y mejora el rendimiento de grandes archivos al reducir la fragmentación y el espacio consumido por metadatos. La asignación retardada aumenta la posibilidad de que un archivo sea escrito en un grupo contiguo de bloques, que reduce la fragmentación y mejora el rendimiento. La pre-asignación sirve para evitar la fragmentación totalmente en los casos en que la aplicación conoce la cantidad necesaria para escribir en tiempo anticipado.

XFS proporciona excelente estabilidad de escalabilidad de E/S mediante árboles-b para indexar todos

los datos y metadatos de usuarios. El conteo de objetos aumenta a medida que las operaciones heredan las características de escalabilidad logarítmica de árboles-b subyacentes. Algunas de las opciones de ajuste que XFS proporciona en tiempo **mkfs** varían la anchura de los árboles-b, lo cual cambia las características de escalabilidad de diferentes subsistemas.

7.3.2.1. Ajuste básico para XFS

En general, las opciones de formato XFS y de montaje son óptimas para la mayoría de las cargas de trabajo; Red Hat recomienda utilizar los valores predeterminados, a menos que se hagan cambios de configuración específicos para beneficiar la carga de trabajo del sistema de archivos. Si el RAID de software está en uso, el comando **mkfs.xfs** se configurará automáticamente con la unidad de banda correcta para alinearse con el hardware. Este proceso debe ser configurado de forma manual si el RAID de hardware está en uso.

La opción de montaje **inode64** se recomienda altamente para sistemas de archivos, excepto cuando el sistema de archivos es exportado mediante NFS y los clientes NFS de legado de 32 bits requieren acceder al sistema de archivos.

La opción de montaje **logbsize** es la recomendad para los sistemas de archivos que son modificados con frecuencia o en ráfagas. El valor predeterminado es **MAX** (32 KB, unidad de bandas de registro), y el tamaño máximo es 256 KB. Para sistemas de archivos que sufren modificaciones profundas, se recomienda un valor de 256 KB .

7.3.2.2. Ajuste avanzado para XFS

Antes de cambiar parámetros XFS, es necesario entender por qué los parámetros XFS predeterminados están causando problemas de rendimiento. Esto implica entender lo que su aplicación está haciendo y la forma como está reaccionando a esas operaciones.

Los problemas de rendimiento observables que se pueden corregir o reducir mediante ajustes, suelen ser causados por la fragmentación de archivos o la contención de recursos en el sistema de archivos. Existen varias formas de solucionar estos problemas y en algunos casos la corrección del problemas requerirá modificar la aplicación en lugar de la configuración del sistema de archivos.

Si usted no ha pasado por el proceso anteriormente, se recomienda que contacte a su ingeniero de soporte local de Red Hat para obtener asistencia.

Optimización para una gran cantidad de archivos

XFS impone un límite arbitrario del número de archivos que puede guardar un sistema de archivos. En general, este límite es lo suficientemente alto que nunca será alcanzado. Si usted sabe con anterioridad que el límite predeterminado es insuficiente, puede aumentar el porcentaje del espacio del sistema de archivos permitidos por inodos con el comando **mkfs.xfs**. Si se da cuenta del límite de archivos después de crear el sistema de archivos (que suele indicarse mediante errores ENOSPC cuando se intente crear un archivo o directorio aunque el espacio libre esté disponible), puede ajustar el límite con el comando **xfs_growfs**.

Cómo optimizar un gran número de archivos en un directorio individual

El tamaño de bloque de directorio se fija para el archivo del sistema de archivos, y no puede cambiarse excepto durante el formato inicial con **mkfs**. El bloque de directorio mínimo es el tamaño de bloque del sistema de archivos, el cual se predetermina a**MAX** (4 KB, tamaño de bloque de archivos). En general, no hay razón para reducir el tamaño de bloque de directorio.

Puesto que la estructura de directorio se basa en árbol-b, el cambio de bloque afecta la cantidad de información del directorio que puede recuperarse o modificarse por E/S física. Entre más grande sea el directorio, más operación de cada E/S se requerirá en un tamaño de bloque determinado.

Sin embargo, al usar un tamaño más grande de bloque de directorio, se consume más CPU por cada operación de modificación en comparación con la misma operación en un sistema de archivos de tamaño de bloque de directorio más pequeño. Es decir que para directorios pequeños, los bloques de directorios grandes se traducirán en modificaciones más bajas de rendimiento. Cuando el directorio alcanza un tamaño en el que la E/S es el factor limitante de rendimiento, los bloques de directorio producen mejor rendimiento.

La configuración predeterminada de un bloque de sistema de archivos de 4 KB y un directorio de bloque de directorio de 4 KB es mejor para directorios hasta de 1-2 millón de entradas con una longitud de nombre de 20-40 bytes por entrada. Si su sistema de archivos requiere más entradas, los tamaños superiores de bloque de directorio tienden a rendir mejor - un bloque de 16 KB es mejor que un sistema de archivos de con 1-10 millones de entradas de directorio, y un tamaño de bloque de 64 KB es mejor para sistemas de archivos con más de 10 millones de entradas de directorio.

Si la carga de trabajo usa búsquedas de directorios aleatorios más que modificaciones (es decir, las lecturas de directorios son más comunes o importantes que las escrituras de directorio), entonces los umbrales anteriores para aumentar el tamaño de bloque son aproximadamente un orden de magnitud inferior.

Optimización para concurrencia

A diferencia de otros sistemas de archivos, XFS puede realizar al mismo tiempo varios tipos de operaciones de asignación o desasignación, siempre y cuando las operaciones se presenten en objetos no compartidos. La operaciones de asignación o desasignación de extensiones pueden presentarse de manera concurrente, siempre y cuando las operaciones concurrentes se presenten en diferentes grupos de asignación. Igualmente, los inodos de asignación o desasignación pueden ocurrir al mismo tiempo, siempre y cuando las operaciones simultáneas afecten diferentes grupos de asignación.

La cantidad de grupos de asignación es importante cuando se utilizan máquinas con alto conteo de CPU y aplicaciones multihilos que intentan realizar operaciones simultáneas. Si hay solo cuatro grupos de asignación, entonces las operaciones de metadatos sostenidos y paralelos únicamente escalarán hasta cuatro CPU (límite de concurrencia provisto por el sistema). Para sistemas de archivos pequeños, asegúrese de que el número de grupos de asignación esté soportado por la concurrencia provista por el sistema. Para sistemas de archivos grandes (decenas de TB y superiores) las opciones de formato predeterminado suelen crear suficientes grupos de asignación para evitar limitación de concurrencia.

Las aplicaciones deben tener en cuenta los puntos de contención para usar el paralelismo inherente en la estructura del sistema de archivos XFS. No es posible modificar de forma concurrente un directorio, por lo tanto las aplicaciones que crean y retiran grandes cantidades de archivos deben evitar almacenar todos los archivos en un solo directorio. Cada directorio creado es colocado en un grupo de asignación diferente, de esta manera los archivos con técnicas de dispersión en múltiples subdirectorios proporcionan un patrón de almacenamiento más escalable en comparación con el uso de un directorio individual grande.

Optimización para aplicaciones que utilizan atributos extendidos

XFS puede almacenar directamente atributos en el inodo si hay espacio disponible en el inodo. Si el atributo se ajusta dentro del inodo, entonces puede ser recuperado y modificado sin requerir E/S adicional para recuperar bloques de atributos independientes. La diferencia de rendimiento entre los atributos en-línea y fuera de línea puede fácilmente ser una orden de magnitud más lenta para atributos fuera de línea.

Para el tamaño predeterminado de inodo de 256 bytes, hay disponibles 100 bytes de espacio de atributo aproximadamente, dependiendo del número de punteros de extensión de datos también almacenados

en el inodo. El tamaño de inodo predeterminado es realmente útil para almacenar un número escaso de pequeños atributos.

El aumento del tamaño del inodo en tiempo mkfs puede también incrementar la cantidad de espacio disponible para atributos de almacenamiento en línea. Un inodo de 512 bytes aumenta el espacio disponible para atributos a350 aproximadamente; un inodo de 2 KB tiene 1900 bytes disponibles aproximadamente.

Sin embargo, existe un límite de tamaño de los atributos individuales que pueden almacenarse en línea - un límite máximo de 254 bytes tanto para el nombre del atributo como para el valor (es decir, un atributo con una longitud de nombre de 254 bytes estará en línea). Si se excede de estos tamaños límites obligará a los atributos a estar fuera de línea incluso si han tenido suficiente espacio para almacenar todos los atributos en el inodo.

Optimización para modificaciones de metadatos sostenidos

El tamaño de registro es el factor principal para determinar el nivel alcanzable de modificación de metadatos sostenidos. El dispositivo de registro es circular, por lo tanto antes de la cola pueden sobrescribirse todas las modificaciones en el registro en disco. Esto puede implicar una cantidad significativa de búsqueda para restaurar todos los datos sucios de metadatos. La configuración predeterminada escala el tamaño de registro en relación con el tamaño de sistema en general, por lo tanto en la mayoría de los casos el tamaño de registro no requerirá ajuste.

Un dispositivo pequeño de registro se traduce en escritura diferida - el registro empujará constantemente su cola para liberar el espacio y por lo tanto los metadatos que suelen ser modificados se escribirán en disco, retrasando así las operaciones.

El aumento del tamaño del registro aumenta el periodo de tiempo entre los eventos de envío de cola. Lo cual permite la mejor agregación de metadatos sucios, y a su vez, produce mejores patrones de retro-escritura de metadatos y menos retro-escritura de metadatos modificados con frecuencia. La desventaja es que los registros más grandes requieren más memoria para rastrear todos los cambios importantes en memoria.

Si usted tiene una máquina con una memoria limitada, entonces los registros grandes no son benéficos, ya que las limitaciones de memoria harán que los metadatos se restauren mucho antes que los beneficios de un registro grande pueda realizarse. En estos casos, los registros más pequeños proveerán mejor rendimiento que los grandes, puesto que los metadatos del registro que le falta espacio es más eficiente que la retro-escritura controlada por el reclamo de memoria.

Siempre debe tratar de alinear el registro con la banda subyacente del dispositivo que contiene el sistema de archivos. **`mkfs`** lo hace de forma predeterminada para dispositivos MD y DM, pero debe especificarse para RAID de hardware. Al configurarlo correctamente se evita toda posibilidad de que registro de E/S cause E/S no alineada y posteriores operaciones de 'lectura-modificar-escritura' al escribir los cambios al disco.

La operación de registro puede mejorarse al modificar las opciones. El aumento de tamaño de búferes de registro en memoria (**`logbsize`**) aumenta la velocidad de los cambios que pueden escribirse al registro. El tamaño de registro predeterminado es **`MAX`** (32 KB, unidad de banda de registro), y el tamaño máximo es 256 KB. En general, un valor superior produce un rendimiento más rápido. Sin embargo, en cargas de trabajo pesadas de fsync, los búferes de registro pequeño pueden ser notablemente más rápidos que los búferes grandes con una alineación de unidad de banda grande.

La opción de montaje **`delaylog`** también mejora el rendimiento de modificaciones de metadatos sostenidos mediante la reducción del número de cambios al registro. Esto se produce al agregar cambios individuales en memoria antes de escribirlos al registro: en lugar de escribir los metadatos frecuentes cada vez se modifican, se escriben al registro de forma periódica . Esta opción aumenta el uso de memoria de rastreo de metadatos sucios y aumenta las operaciones de pérdida potenciales

cuando se presenta una falla, pero puede mejorar la velocidad de modificación de metadatos y la escalabilidad mediante un orden de magnitud o más. El uso de esta opción no reduce la integridad de los datos o metadatos cuando **`fsync`**, **`fdatasync`** o **`sync`** sean utilizados para garantizar que los datos y metadatos sean escritos a disco.

7.4. AGRUPAMIENTO

El almacenamiento en clústeres proporciona una imagen de un sistema de archivos consistente a través de todos los servidores en un clúster, lo cual permite que los servidores lean y escriban a un sistema de archivos único y compartido. Este procedimiento simplifica la administración de almacenamiento al limitar tareas tales como instalar o corregir aplicaciones para un sistema de archivos. Un sistema de archivos amplio en clúster también elimina la necesidad de copias redundantes de datos de aplicaciones, al mismo tiempo que simplifica la copia de seguridad y la recuperación de desastres.

La adición de alta disponibilidad de Red Hat proporciona almacenamiento en clúster junto con el Sistema de archivos global 2 de Red Hat (parte de adición de almacenamiento resistente).

7.4.1. Sistema de archivos global 2

El Sistema de archivos global 2 (GFS2) es un sistema de archivos que interactúa directamente con el sistema de archivos de kernel de Linux. Le permite a varios computadores (nodos) compartir de forma simultánea en un clúster. El sistema de archivos GFS2 es ampliamente auto-ajustable, pero también se puede ajustar de forma manual. Esta sección presenta algunas consideraciones sobre el ajuste manual.

Red Hat Enterprise Linux 6.4 introduce mejoras para administrar la fragmentación de archivos en GFS2. Los archivos creados por Red Hat Enterprise Linux 6.3 o anteriores eran propensos a fragmentarse si los archivos múltiples eran escritos al mismo tiempo por más de un proceso. Esta fragmentación retrasaba la ejecución, sobre todo en las cargas de trabajo que implicaban grandes archivos. Con Red Hat Enterprise Linux 6.4, la escritura simultánea se traduce en menor fragmentación de archivos y por lo tanto, en mejor rendimiento para dichas cargas de trabajo.

Aunque no hay una herramienta de desfragmentación para GFS2 en Red Hat Enterprise Linux, usted puede desfragmentar archivos individuales al identificarlos con la herramienta **filefrag**, copiarlos en archivos temporales y renombrarlos para remplazar los originales. (Este procedimiento también puede realizarse en versiones anteriores a 6.4 siempre y cuando la lectura se haga de forma secuencial.)

Puesto que GFS2 usa un mecanismo de cerramiento global, el cual requiere comunicación entre nodos de un clúster, el mejor rendimiento se realizará cuando su sistema esté diseñado para evitar la contención de archivo y directorio entre estos nodos. Algunos métodos para evitar contención son los siguientes:

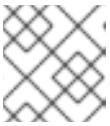
- Pre-asignar archivos y directorios con **`fallocate`** cuando sea posible para optimizar el proceso de asignación y evitar así tener que bloquear las páginas de origen.

- Minimizar las áreas del sistema de archivos que se comparten entre múltiples nodos para minimizar la invalidación de cache a través de nodos y mejorar el rendimiento. Por ejemplo, si varios nodos montan el mismo sistema de archivos, es muy probable que obtenga un mejor rendimiento si traslada un subdirectorio a un sistema de archivos independiente.

- Seleccionar un tamaño y número de grupo de recursos óptimo. Esto depende del tamaño de archivos y del espacio libre disponible en el sistema, y afecta la probabilidad de que múltiples nodos intenten usar un grupo de recursos de forma simultánea. Demasiados grupos de recursos pueden demorar la asignación de bloques cuando se localiza el espacio de asignación. Suele ser mejor probar las configuraciones para determinar cuál es mejor para su carga de trabajo.

Sin embargo, la contención no es el único problema que afecta el rendimiento del sistema de archivos GFS2. Otras prácticas para mejorar el rendimiento total, son:

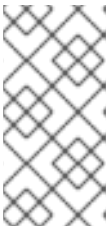
- Seleccione su hardware de almacenamiento según los patrones de E/S esperados de los nodos de clúster y los requerimientos de rendimiento del sistema de archivos.
- Use el almacenamiento de estado sólido en donde sea posible disminuir el tiempo de búsqueda.
- Cree un sistema de archivos con el tamaño apropiado para su carga de trabajo y asegúrese de que el sistema de archivos no esté nunca a una capacidad mayor que 80%. Los sistemas de archivos más pequeños tendrán en forma proporcional, tiempos más cortos de copia de seguridad, pero estarán sujetos a una alta fragmentación si son demasiado pequeños para la carga de trabajo.
- Establezca tamaños de diarios para cargas de trabajo intensivas de metadatos o cuando los datos puestos en diario estén en uso. Aunque se usa más memoria, mejora el rendimiento, ya que el espacio de diario está disponible para almacenar datos antes de que una escritura sea necesaria.
- Asegúrese de que los relojes en nodos GFS2 estén sincronizados para evitar problemas con aplicaciones en red. Le recomendamos el uso de NTP (Protocolo de tiempo de red).
- A menos que los tiempos de acceso de directorio o archivos sean críticos para la operación de su aplicación, monte el sistema de archivos con las opciones de montaje de **`noatime`** y **`nodiratime`**.



NOTA

Red Hat recomienda el uso de la opción **`noatime`** con GFS2.

- Si necesita utilizar cuotas, trate de reducir la frecuencia de transacciones de sincronización de cuotas o utilice la sincronización de cuota difusa para evitar problemas de rendimiento que surgen de constantes actualizaciones de archivos



NOTA

Las cuotas de conteo difuso permiten a los usuarios y grupos excederse un poco del límite de cuota. Para minimizar este problema, GFS2 reduce de forma dinámica el periodo de sincronización cuando el usuario o grupo se acerca a la cuota límite.

Para obtener mayor información sobre cada aspecto del ajuste de rendimiento de GFS2, consulte la Guía del *Sistema de archivos global 2*, disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

CAPÍTULO 8. REDES

Con el tiempo, la pila de redes de Red Hat Enterprise Linux ha sido mejorada con varias funcionalidades de optimización automatizadas. Para la mayoría de las cargas de trabajo, los parámetros de redes autoconfiguradas proporcionan rendimiento optimizado.

En la mayoría de los casos, los problemas de rendimiento de redes son causados por el mal funcionamiento del hardware o por una estructura defectuosa. Dichos casos van más allá de este documento; los problemas de rendimiento y las soluciones que se discuten en este capítulo son útiles para optimizar los sistemas funcionales.

Redes es un subsistema delicado, que consta de diferentes partes con conexiones sensitivas. Esta es la razón por la cual la comunidad de código abierto y Red Hat invierten mucho trabajo en implementar automáticamente formas para optimizar el rendimiento de redes. Como tal, dada la mayoría de cargas de trabajo, usted nunca necesitará reconfigurar las redes para rendimiento.

8.1. MEJORAS DE RENDIMIENTO DE REDES

Red Hat Enterprise Linux 6.1 proporciona las siguientes mejoras de rendimiento de redes:

Direccionamiento de paquetes recibidos (RPS)

RPS habilita una sola cola de NIC **rx** que tiene su carga de recepción **softirq** distribuida entre varias CPU. Esto ayuda a evitar el embotellamiento de tráfico de redes de en una sola cola de hardware de NIC.

Para activar RPS, especifique los nombres de destino de CPU en **/sys/class/net/*ethX*/queues/*rx-N*/rps_cpus**, remplace ***ethX*** por el nombre de dispositivo correspondiente de NIC (por ejemplo, **eth1**, **eth2**) y ***rx-N*** con la cola de recepción del NIC especificado. Así, permitirá a las CPU especificadas en el archivo procesar datos de la cola ***rx-N*** en ***ethX***. Al especificar las CPU, considere la *afinidad de cache*[4].

Direccionamiento de flujo recibido

RFS es una extensión de RPS que permite al administrador configurar una tabla hash generada automáticamente cuando las aplicaciones reciben datos y son interrogadas por la pila de redes. Esto determina cuáles aplicaciones están recibiendo cada parte de datos de redes.

Mediante esta información, la pila de redes puede programar la CPU óptima para recibir cada paquete. Para configurar RFS, use los siguientes ajustables:

/proc/sys/net/core/rps_sock_flow_entries

Este ajustable controla el número máximo de sockets/flujos que el kernel puede dirigir hacia cualquier CPU especificada. Se trata de un sistema amplio y de límite compartido.

/sys/class/net/*ethX*/colas/*rx-N*/rps_flow_cnt

Este ajustable controla el máximo número de sockets/flujos que el kernel puede dirigir para una cola de recibo especificada (***rx-N***) en un NIC (***ethX***). Observe que la suma de todos los valores por cola para este ajustable en todos los NIC deben ser igual o menor que los **/proc/sys/net/core/rps_sock_flow_entries**.

A diferencia de RPS, RFS le permite tanto a la cola de recibo como a la aplicación compartir la misma CPU al procesar flujos de paquetes. Esto mejora el rendimiento en algunos casos. Sin embargo, dichas mejoras dependen de factores tales como jerarquía de cache, carga de aplicación y similares.

Soporte getsockopt para corrientes finas TCP

Corriente fina o Thin-stream es un término utilizado para caracterizar protocolos de transporte donde las aplicaciones envían datos a una tasa tan baja que los mecanismos de retransmisión de protocolo no se saturan totalmente. Las aplicaciones que usan protocolos de corriente fina suelen transportarse a través de protocolos confiables tales como TCP; en la mayoría de los casos, dichas aplicaciones ofrecen servicios muy sensibles al tiempo (por ejemplo, comercio de acciones, juegos en línea y sistemas de control).

Para los servicios de tiempo sensible, la pérdida de paquetes puede ser devastadora en la calidad de servicios. Para evitar dicha pérdida, la llamada **`getsockopt`** ha sido mejorada para soportar dos opciones adicionales:

TCP_THIN_DUPACK

Este booleano habilita la activación dinámica de retransmisiones después de un dupACK para flujos finos.

TCP_THIN_LINEAR_TIMEOUTS

Este booleano habilita la activación dinámica de tiempos de espera para flujos finos.

Ambas opciones son activadas de forma específica por la aplicación. Para obtener mayor información, consulte **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/ip-sysctl.txt`**. Para obtener mayor información sobre flujos finos, consulte **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/tcp-thin.txt`**.

Soporte de proxy transparente (TProxy)

Ahora el kernel puede manipular sockets IPv4 TCP y UDP no vinculados localmente para soportar proxis transparentes. Para habilitarlo, deberá configurar iptables como corresponde. También deberá activar y configurar la política de ruta correctamente.

Para obtener mayor información sobre proxis transparentes, consulte **`file:///usr/share/doc/kernel-doc-`*`versión`*`/Documentation/networking/tproxy.txt`**.

8.2. PARÁMETROS DE REDES OPTIMIZADAS

El ajuste de rendimiento suele realizarse en un modo de prevaciado. A menudo, ajustamos variables conocidas antes de ejecutar la aplicación o implementar un sistema. Si los sondeos de ajuste no son efectivos, tratamos de ajustar otras variables. La lógica detrás de este pensamiento es que de *forma predeterminada*, el sistema no está funcionando en un nivel óptimo de rendimiento; como tal, *pensamos* que debemos ajustar el sistema como corresponde. En algunos casos lo hacemos mediante conjeturas.

Como se mencionó anteriormente, la pila de redes es principalmente auto-optimizadora. Además, para un ajuste de red efectivo se requiere un entendimiento profundo no solamente de la forma como funciona la red, sino también de los requerimientos de recursos de red. La configuración de rendimiento de red incorrecta, puede conducir a un rendimiento degradado.

Por ejemplo considere el *Problema bufferfloat* . Al aumentar el valor de cola del búfer, las conexiones

TCP que tienen ventanas de congestión más grandes que el enlace se permitirían (debido al alto valor de búfer). Sin embargo, dichas conexiones también tienen amplios valores RTT, puesto que los marcos consumen mucho tiempo en cola. A su vez, produce una salida subóptima, ya que sería imposible detectar la congestión.

Cuando se trata del rendimiento de redes, es aconsejable mantener los parámetros predeterminados *a menos que* se haga evidente un problema de rendimiento particular. Problemas como tal, incluyen pérdida de marco, reducción significativa de rendimiento y similares. Aún así, la mejor solución suele ser la que resulta de un estudio meticuloso del problema, en lugar de simplemente ajustar los parámetros a lo alto (aumentando las longitudes de cola y búfer y reduciendo latencia de interrupciones, etc).

Las siguientes herramientas le servirán para diagnosticar correctamente el problema de rendimiento de red:

netstat

Una herramienta de línea de comandos que imprime conexiones de redes, tablas de rutas, estadísticas de interfaz, conexiones de máscara y membresías multidifusión. Recupera la información sobre el subsistema de redes del sistema de archivos **`/proc/net/`**. Dichos archivos incluyen:

- **`/proc/net/dev`** (información de dispositivos)
- **`/proc/net/tcp`** (Información de socket TCP)
- **`/proc/net/unix`** (Informaciónn de sockete de dominio Unix)

Para obtener mayor información sobre **`netstat`** y sus archivos mencionados en **`/proc/net/`**, consulte la página de manual **`netstat`**: **`man netstat`**.

dropwatch

Una herramienta que monitoriza los paquetes eliminados por el kernel. Para mayor información, consulte la página de manual **`dropwatch`**: **`man dropwatch`**.

ip

Una herramienta para administrar y monitorizar rutas, dispositivos, y políticas túneles y rutas de políticas . Para obtener mayor información, consulte la página de manual de **`ip`**: **`man ip`**.

ethtool

Una herramienta para desplegar y cambiar los parámetros NIC. Para obtener mayor información, consulte la página de manual **`ethtool`**: **`man ethtool`**.

/proc/net/snmp

Un archivo que muestra datos ASCII necesarios para las bases de información de administración de IP, ICMP, TCP, y UDP para un agente **`snmp`**. También despliega estadísticas de UDP-lite en tiempo real

La *Guía para principiantes de SystemTap* contiene varias muestras de scripts que puede utilizar para perfilar y monitorizar el rendimiento de redes. Esta guía está disponible en http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Después de recolectar los datos relevantes sobre el problema de rendimiento de redes, podrá formular

una teoría — y, posiblemente, una solución. [5] Por ejemplo, un aumento en errores de entrada UDP en **/proc/net/snmp** indica que uno o más conectores de colas de recepción se llena cuando la pila de redes intenta poner en cola nuevos marcos dentro de un socket de aplicación.

Esto indica que los paquetes están trancados en *por lo menos* una cola de socket, es decir que la cola del socket drena muy lentamente los paquetes o que el volumen de paquetes es demasiado grande para dicha cola de socket. Si se trata de esta última, entonces verifique la pérdida de datos en los registros de aplicación intensiva de redes, para resolverlo, deberá optimizar o reconfigurar la aplicación infractora.

Tamaño de búfer de recepción de socket

Los tamaños de búfer de recepción y envío de socket se ajustan de forma dinámica, por lo tanto, muy pocas veces necesitará modificarlos de forma manual. Si se requiere un mayor análisis, tal como el presentado en el ejemplo de red SystemTap, **sk_stream_wait_memory.stp** sugiere que la tasa de drenaje de cola de socket sea demasiado baja, para que luego usted pueda aumentar la profundidad de cola del socket de la aplicación. Para hacerlo, aumente el tamaño de los búferes utilizados por conectores, mediante la configuración de alguno de los siguientes valores:

rmem_default

Un parámetro de kernel que controla el tamaño *predeterminado* de búferes de recepción utilizado por conectores. Para configurarlo, ejecute el siguiente comando:

```
sysctl -w net.core.rmem_default=N
```

Remplace ***N*** por el tamaño en bytes del búfer deseado. Para determinar el valor para este parámetro de kernel, vea **/proc/sys/net/core/rmem_default**. Tenga en cuenta que el valor de **rmem_default** no debería ser mayor que **rmem_max** (**/proc/sys/net/core/rmem_max**); en caso de serlo, aumente el valor de **rmem_max**.

SO_RCVBUF

Una opción de conexión que controla el tamaño *máximo* de un búfer de recepción de socket, en bytes. Para obtener mayor información sobre **SO_RCVBUF**, consulte la página de manual: **man 7 socket**.

Para configurar **SO_RCVBUF**, use la herramienta **setsockopt**. Puede recuperar el valor actual de **SO_RCVBUF** con **getsockopt**. Para obtener mayor información sobre el uso de estas dos herramientas, consulte la página de manual de **setsockopt**: **man setsockopt**.

8.3. VISIÓN GENERAL DE RECEPCIÓN DE PAQUETES

Para analizar mejor los cuellos de botella y los problemas de rendimiento, es necesario entender cómo funciona la recepción de paquetes. La recepción de paquetes es esencial en el ajuste de rendimiento de redes. Los marcos que se pierden en la ruta de recepción pueden impactar de forma significativa el rendimiento de red.

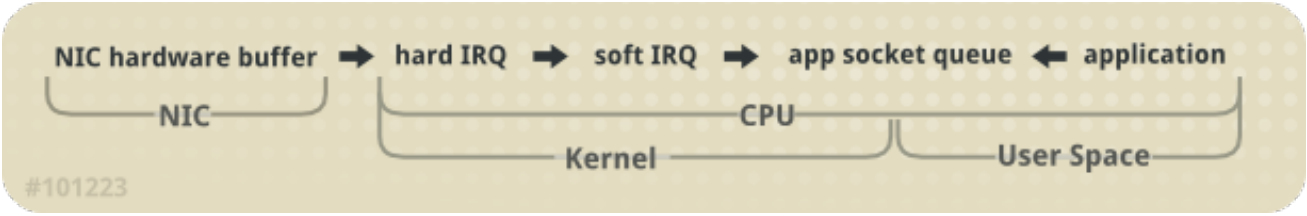


Figura 8.1. Diagrama de rutas de recepción de redes

El kernel de Linux recibe cada marco y lo sujeta a un proceso de cuatro etapas:

1. *Recepción de hardware*: la *tarjeta de interfaz de red* (NIC) recibe el marco en el alambre. Según su configuración de controlador, el NIC transfiere el marco ya sea a una memoria de hardware interna o a un búfer de anillo determinado.

2. *IRQ dura*: el NIC impone la presencia de un marco de red al interrumpir la CPU. Así, el controlador reconoce la interrupción y programa la *operación de IRQ blanda*.

3. La *IRQ blanda*: esta etapa implementa el proceso de recepción de marcos y se ejecuta en contexto **`softirq`**. Es decir que la etapa prevacía todas las aplicaciones que se ejecutan en la CPU especificada, pero aún permite que los IRQ duros sean impuestos.

   En este contexto (ejecutándose en la misma CPU como IRQ dura, minimizando así la sobrecarga de cerramiento), el kernel retira el marco de los búferes de hardware de NIC y lo procesa mediante la pila de red. Desde ese punto, el marco puede ser reenviado, descartado o trasladado a un destino de socket de escucha.

   Cuando pasa a un socket, el marco se añade a la aplicación que posee el socket. Este proceso se realiza de forma repetitiva hasta que se agoten los marcos del búfer de hardware de NIC o hasta que el *peso de dispositivo* (**`dev_weight`**). Para obtener mayor información sobre peso de dispositivos, consulte la Sección 8.4.1, “Búfer de hardware de NIC”

4. *Recibos de aplicación* : la aplicación recibe el marco y saca de la cola los sockets que posea vía las llamadas POSIX (**`read`**, **`recv`**, **`recvfrom`**). En este punto, los datos recibidos en la red ya no existen en la pila de red.

Afinidad CPU/cache

Para mantener alto rendimiento en la ruta de recibo, se recomienda mantener la cache de L2 *caliente*. Como se describió anteriormente, los búferes de red se reciben en la misma CPU como la IRQ que señaló su presencia. Es decir, que los datos de búferes no estará en la cache L2 de esa CPU de recepción.

Para sacar provecho de esto, sitúe la afinidad del proceso en aplicaciones que esperan recibir la mayoría de los datos en el NIC que comparte el mismo núcleo de la cache L2. Este procedimiento maximizará las oportunidades de tocar la cache y así, mejorar el rendimiento.

8.4. SOLUCIÓN DE PROBLEMAS COMUNES DE COLAS Y PÉRDIDA DE MARCOS

Hasta el momento, la razón más común por pérdida de marcos es el *exceso de cola*. El kernel establece un límite a una longitud de una cola y en algunos casos la cola se llena más rápido que lo que drena. Cuando esto se presenta por demasiado tiempo, los marcos comienzan a perderse.

Como se ilustra en la Figura 8.1, “Diagrama de rutas de recepción de redes”, hay dos colas principales en la ruta de recibo: el búfer de hardware de NIC y la cola de socket. Ambas colas necesitan configurarse como corresponde para protegerse de los excesos de cola.

8.4.1. Búfer de hardware de NIC

El NIC llena su búfer de hardware con marcos; luego el búfer es drenado por **softirq**, el cual NIC afirma a través de una interrupción. Para interrogar el estatus de esta cola, use el siguiente comando:

```
ethtool -S ethX
```

Remplace ***ethX*** por el nombre de dispositivo del NIC correspondiente. Así desplegará el número de marcos que han sido eliminados dentro de ***ethX***. Una pérdida suele ocurrir debido a que a la cola se le acaba el espacio de búfer para almacenar marcos.

Hay dos formas de solucionar este problema:

Tráfico de entrada

Puede evitar excesos de cola si disminuye el paso del tráfico de entrada. Esto se logra por filtraje, reduciendo el número de grupos de multidifusión y disminuyendo el tráfico de difusión, entre otros.

Longitud de cola

También, puede aumentar la longitud de cola. Esto implica el aumento del número de búferes en una cola especificada para cualquier máximo que permita el controlador. Para hacerlo, modifique los parámetros de anillo **rx**/**tx** de ***ethX*** mediante:

```
ethtool --set-ring ethX
```

Añada los valores **rx** or **tx** apropiados al comando antes mencionado. Para obtener mayor información, consulte **man ethtool**.

Peso de dispositivo

También puede aumentar la tasa a la que se drena una cola. Para hacerlo, ajuste el *peso del dispositivo* como corresponde. Este atributo se refiere al número máximo de marcos que el NIC puede recibir antes que el contexto **softirq** tenga que entregar la CPU y reprogramarse. Se controla mediante la variable **/proc/sys/net/core/dev_weight**.

La mayoría de administradores tienden a escoger la tercera opción. Sin embargo, tenga presente que habrá consecuencias al hacerlo. El aumento del número de marcos que pueden recibirse desde el NIC en una repetición, implica ciclos de CPU adicionales durante los cuales no se pueden programar aplicaciones en esa CPU.

8.4.2. Cola de socket

A diferencia de la cola de hardware de NIC, la cola de socket es llenada por la pila de red desde el contexto **softirq**. Luego, las aplicaciones drenan las colas de los sockets correspondientes mediante llamadas a **read**, **recvfrom**, y similares.

Para monitorizar el estatus de esta cola, use la herramienta **netstat**; la columna **Recv-Q** muestra el tamaño de la cola. En general, la exceso de ejecución en la cola del socket se administra en la misma forma que los excesos de ejecución de búfer de hardware de NIC (e.j. Sección 8.4.1, “Búfer de hardware de NIC”):

Tráfico de entrada

La primera opción es disminuir el tráfico de entrada al configurar la tasa en la cual se llena la cola. Para hacerlo, puede filtrar marcos o prevaciarlos. También puede retardar el tráfico de entrada al disminuir el peso de dispositivos[6].

Profundidad de cola

También puede evitar que la cola se exceda al aumentar la profundidad de cola. Para hacerlo, aumente el valor del parámetro de kernel **`rmem_default`** o la opción de socket **`SO_RCVBUF`**. Para mayor información, consulte la Sección 8.2, “Parámetros de redes optimizadas”.

Frecuencia de llamada de aplicación

Cuando sea posible, optimice la aplicación para realizar llamadas de una forma más frecuente. Esto implica modificar o reconfigurar la aplicación de redes para realizar más llamadas POSIX (tales como **`recv`**, **`read`**). A su vez, esto permite a una aplicación drenar la cola de una forma más rápida.

Para muchos administradores, el aumento de la profundidad de cola es la solución preferida. Aunque es la solución más fácil, no siempre es la que funciona a largo plazo. Como las tecnologías de redes se tornan más rápidas, las colas de sockets se llenarán de una forma más rápida. Con el tiempo, esto significa tener que reajustar la profundidad de la cola como corresponda.

La mejor solución es mejorar o configurar la aplicación para drenar los datos del kernel de una forma más rápida, incluso si esto significa poner en cola los datos en espacio de aplicaciones. Así se pueden almacenar datos de una forma más flexible, ya que se puede intercambiar y revertir cuando sea necesario.

8.5. CONSIDERACIONES DE MULTIDIFUSIÓN

Cuando se escuchan múltiples aplicaciones para un grupo de multidifusión, el código de kernel que administra marcos de multidifusión es requerido por diseño para duplicar los datos de red de cada socket individual. Esta duplicación consume tiempo y se presenta en el contexto de **`softirq`**.

La adición de múltiples oyentes en un grupo de multidifusión individual tiene un impacto directo en el tiempo ejecución del contexto **`softirq`**. La adición de un oyente a un grupo de multidifusión, implica que el kernel debe crear una copia adicional de cada marco recibido para dicho grupo.

El efecto es mínimo en volumen de tráfico bajo y pequeña cantidad de oyentes. Sin embargo, cuando los sockets múltiples escuchan a un grupo multidifusión en un tráfico alto, el tiempo de ejecución aumentado del contexto **`softirq`** puede conducir a pérdidas de marcos tanto en tarjetas de red como en colas de sockets. El aumento de tiempos de ejecución de **`softirq`** se traduce en una oportunidad reducida para que las aplicaciones se ejecuten en sistemas de carga pesada, entonces la tasa en la que los marcos de multidifusión se pierden aumenta a medida que aumenta el número de aplicaciones que escuchan al grupo de multidifusión de alto volumen.

Puede resolver esta pérdida de marco al optimizar sus colas de conectores y búferes de hardware de NIC, como se describe en la Sección 8.4.2, “Cola de socket” o en la Sección 8.4.1, “Búfer de hardware de NIC”. Igualmente, puede optimizar el uso de un socket de aplicación; para hacerlo, configure la aplicación para controlar un socket individual y diseminar rápidamente los datos de red recibidos a otros procesos de espacio de usuario.

---

[4] de la cola. Asegurar afinidad de cache entre una CPU y un NIC significa configurarlos para compartir la misma cache L2. Para obtener mayor información, consulte Sección 8.3, “Visión general de recepción de paquetes”.

[5] Sección 8.3, “Visión general de recepción de paquetes” contiene una visión general de viaje de paquetes, que ayudarán a localizar y trazar áreas propensas a cuellos de botellas.

[6] El peso de dispositivos se controla mediante **`/proc/sys/net/core/dev_weight`**. Para obtener mayor información sobre el peso de dispositivos y las implicaciones de su ajuste, consulte la Sección 8.4.1, “Búfer de hardware de NIC”.

APÉNDICE A. HISTORIAL DE REVISIONES

Revisión 4.0-22.2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

Revisión 4.0-22.2 **Thu Jun 27 2013** **Gladys Guerrero-Lozano**
Revisión completa

Revisión 4.0-22.1 **Thu Apr 18 2013** **Chester Cheng**
Los archivos de traducción sincronizados con fuentes XML 4.0-22

Revisión 4.0-22 **Fri Feb 15 2013** **Laura Bailey**
Publicación para Red Hat Enterprise Linux 6.4.

Revisión 4.0-19 **Wed Jan 16 2013** **Laura Bailey**
Correcciones menores de consistencia (BZ#868404).

Revisión 4.0-18 **Tue Nov 27 2012** **Laura Bailey**
Publicación para Red Hat Enterprise Linux 6.4 Beta.

Revisión 4.0-17 **Mon Nov 19 2012** **Laura Bailey**
Se añadieron comentarios SME en la sección re. numad (BZ#868404).

Revisión 4.0-16 **Thu Nov 08 2012** **Laura Bailey**
Se añadió el borrador de la sección sobre numad (BZ#868404).

Revisión 4.0-15 **Wed Oct 17 2012** **Laura Bailey**
Aplicación de comentarios SME a discusión de descarte de bloques y se trasladó la sección a Opciones de montaje (BZ#852990).
Se actualizaron las descripciones de perfil de rendimiento (BZ#858220).

Revisión 4.0-13 **Wed Oct 17 2012** **Laura Bailey**
Se actualizaron las descripciones de perfil de rendimiento (BZ#858220).

Revisión 4.0-12 **Tue Oct 16 2012** **Laura Bailey**
Se mejoró la navegación de libro (BZ#854082).
Se corrigió la definición de ***file-max*** (BZ#854094).
Se corrigió la definición de ***threads-max*** (BZ#856861).

Revisión 4.0-9 **Tue Oct 9 2012** **Laura Bailey**
Se añadió la recomendación FSTRIM para el capítulo de Sistemas de archivos (BZ#852990).
Se actualizó la descripción del parámetro ***threads-max*** según los comentarios de los usuarios (BZ#856861).
Se actualizó la nota sobre mejoras de administración de fragmentación de GFS2 BZ#857782).

Revisión 4.0-6 **Thu Oct 4 2012** **Laura Bailey**
Se añadió nueva sección sobre la herramienta numastat (BZ#853274).

Revisión 4.0-3 **Tue Sep 18 2012** **Laura Bailey**
Se añadió nota de funcionalidades re. new perf (BZ#854082).
Se corrigió la descripción del parámetro file-max (BZ#854094).

Revisión 4.0-2 **Mon Sep 10 2012** **Laura Bailey**
Se añadió la sección BTRFS y la introducción básica al sistema da archivos (BZ#852978).
Se anotó la integración Valgrind con GDB (BZ#853279).

Revisión 3.0-15 **Thursday March 22 2012** **Laura Bailey**

Se añadieron y actualizaron descripciones de perfiles de adm ajustados (BZ#803552).

Revisión 3.0-10 **Friday March 02 2012** **Laura Bailey**

Se actualizaron las descripciones de parámetros de file-max y threads-max (BZ#752825).

Se actualizó el valor predeterminado del parámetro slice_idle (BZ#785054).

Revisión 3.0-8 **Thursday February 02 2012** **Laura Bailey**

Se reestructuró y adicionó información sobre taskset y vinculación de CPU y asignación de memoria con numactl para Sección 4.1.2, “Ajuste de rendimiento de la CPU” (BZ#639784).

Se corrigió el uso de enlaces internos (BZ#786099).

Revisión 3.0-5 **Tuesday January 17 2012** **Laura Bailey**

Se hicieron correcciones menores a Sección 5.3, “Cómo utilizar Valgrind para perfilar el uso de memoria” (BZ#639793).

Revisión 3.0-3 **Wednesday January 11 2012** **Laura Bailey**

Se aseguró consistencia entre hiperenlaces internos y externos (BZ#752796).

Se adicionó Sección 5.3, “Cómo utilizar Valgrind para perfilar el uso de memoria” (BZ#639793).

Se adicionó Sección 4.1.2, “Ajuste de rendimiento de la CPU” y reestructuró Capítulo 4, *CPU* (BZ#639784).

Revisión 1.0-0 **Friday December 02 2011** **Laura Bailey**

Lanzamiento para GA de Red Hat Enterprise Linux 6.2.