Red Hat Enterprise Linux 6

Guide de r&#xF;glage des performances

Optimiser le débit des sous-systèmes dans Red Hat Enterprise Linux 6

Édition 4.0

Last Updated: 2017-10-13

Red Hat Enterprise Linux 6 Guide de r�glage des performances

Optimiser le débit des sous-systèmes dans Red Hat Enterprise Linux 6
Édition 4.0

Experts de domaine fonctionnel de Red Hat

Publié par

Don Domingo

Laura Bailey

Notice légale

Résumé

Le Guide de réglage des performances décrit comment optimiser les performances d'un système exécutant Red Hat Enterprise Linux 6. Il documente aussi les mises à niveau concernant la performance sur Red Hat Enterprise Linux 6. Même si ce guide contient des procédures testées sur le terrain et prouvées, Red Hat vous recommande de tester correctement toutes les configurations planifiées dans un environnement de test avant de les appliquer dans un environnement de production. Vous devriez aussi effectuer des copies de sauvegardes de toutes vos données et configurations pré-optimisation.

Table des matières

CHAPITRE 1. APERÇU **4**
1.1. AUDIENCE 4
1.2. ÉVOLUTIVITÉ HORIZONTALE 5
1.2.1. Informatique parallèle 6
1.3. SYSTÈMES DISTRIBUÉS 6
1.3.1. Communication 7
1.3.2. Stockage 8
1.3.3. Réseaux convergents 9

CHAPITRE 2. CARACTÉRISTIQUES DES PERFORMANCES DE RED HAT ENTERPRISE LINUX 6 . . . . . **11**
2.1. PRISE EN CHARGE 64 BITS 11
2.2. TICKET SPINLOCKS 12
2.3. STRUCTURE DE LISTE DYNAMIQUE 12
2.4. NOYAU TICKLESS 12
2.5. GROUPES DE CONTRÔLE 13
2.6. AMÉLIORATIONS DU STOCKAGE ET DES SYSTÈMES DE FICHIERS 14

CHAPITRE 3. CONTRÔLE ET ANALYSE DES PERFORMANCES DU SYSTÈME **17**
3.1. LE SYSTÈME DE FICHIERS PROC 17
3.2. MONITEURS SYSTÈME GNOME ET KDE 17
3.3. OUTILS DE CONTRÔLE INTÉGRÉS SUR LIGNE DE COMMANDE 18
3.4. TUNED ET KTUNE 19
3.5. PROFILEURS D'APPLICATION 20
3.5.1. SystemTap 21
3.5.2. OProfile 21
3.5.3. Valgrind 22
3.5.4. Perf 22
3.6. RED HAT ENTERPRISE MRG 23

CHAPITRE 4. CPU **25**
TOPOLOGIE 25
THREADS 25
INTERRUPTIONS 25
4.1. TOPOLOGIE DE CPU 26
4.1.1. Topologie des CPU et de NUMA 26
4.1.2. Régler les performances CPU 27
4.1.2.1. Définir les affinités de CPU avec taskset 29
4.1.2.2. Contrôler la stratégie NUMA avec numactl 29
4.1.3. numastat 31
4.1.4. numad (« NUMA Affinity Management Daemon ») 33
4.1.4.1. Bénéfices de numad 33
4.1.4.2. Modes d'opération 34
4.1.4.2.1. Utiliser numad en tant que service 34
4.1.4.2.2. Utiliser numad en tant qu'exécutable 34
4.2. ORDONNANCEMENT CPU 35
4.2.1. Stratégies d'ordonnancement en temps réel 35
4.2.2. Stratégies d'ordonnancement normal 36
4.2.3. Sélection de la stratégie 37
4.3. RÉGLAGES DES INTERRUPTIONS ET DES IRQ 37
4.4. AMÉLIORATIONS DE NUMA RED HAT ENTERPRISE LINUX 6 38
4.4.1. Optimisation de l'évolutivité et du bare-metal 38
4.4.1.1. Améliorations de la conscience de la topologie 39

4.4.1.2. Améliorations de la synchronisation de multiples processeurs 39
4.4.2. Optimisation de la virtualisation 40

CHAPITRE 5. MÉMOIRE 41
5.1. HUGETLB (« HUGE TRANSLATION LOOKASIDE BUFFER ») 41
5.2. « HUGE PAGES » ET « TRANSPARENT HUGE PAGES » 41
5.3. UTILISER VALGRIND POUR ÉTABLIR UN PROFIL DE L'UTILISATION DE MÉMOIRE 42
5.3.1. Profiler l'utilisation de mémoire avec Memcheck 42
5.3.2. Profiler l'utilisation du cache avec Cachegrind 43
5.3.3. Profiler les espaces du tas et de pile avec Massif 45
5.4. RÉGLAGE DES CAPACITÉS 46
5.5. RÉGLAGES DE LA MÉMOIRE VIRTUELLE 49

CHAPITRE 6. ENTRÉES/SORTIES 52
6.1. FONCTIONNALITÉS 52
6.2. ANALYSE 52
6.3. OUTILS 54
6.4. CONFIGURATION 57
6.4.1. CFQ (« Completely Fair Queuing ») 58
6.4.2. Programmateur d'E/S « Deadline » 60
6.4.3. Noop 61

CHAPITRE 7. SYSTÈMES DE FICHIERS 63
7.1. CONSIDÉRATIONS POUR OPTIMISER DES SYSTÈMES DE FICHIERS 63
7.1.1. Options de formatage 63
7.1.2. Options de montage 64
7.1.3. Maintenance de systèmes de fichiers 65
7.1.4. Considérations pour l'application 65
7.2. PROFILS POUR LES PERFORMANCES DE SYSTÈMES DE FICHIERS 65
7.3. SYSTÈMES DE FICHIERS 66
7.3.1. Le système de fichiers Ext4 66
7.3.2. Le système de fichiers XFS 67
7.3.2.1. Réglages de base pour XFS 68
7.3.2.2. Réglages avancés pour XFS 68
7.4. CLUSTERING 71
7.4.1. Global File System 2 71

CHAPITRE 8. « NETWORKING » (RÉSEAU) 74
8.1. AMÉLIORATIONS DES PERFORMANCES RÉSEAU 74
RPS (« Receive Packet Steering ») 74
RFS (« Receive Flow Steering ») 74
Prise en charge de getsockopt pour les flux de type « thin-stream » TCP 75
Prise en charge des TProxy (proxys transparents) 75
8.2. PARAMÈTRES RÉSEAU OPTIMISÉS 75
Taille du tampon de réception du socket 77
8.3. APERÇU DES RÉCEPTIONS DE PAQUETS 77
CPU/cache affinity 78
8.4. RÉSOUDRE DES PROBLÈMES COMMUNS DE PERTE DE FILE OU DE TRAME 78
8.4.1. Mémoire tampon matérielle de la carte d'interface réseau (NIC) 79
8.4.2. File de sockets 79
8.5. CONSIDÉRATIONS SUR LA MULTIDIFFUSION 80

ANNEXE A. HISTORIQUE DES VERSIONS 82

CHAPITRE 1. APERÇU

Le *Guide de réglage des performances* est une référence complète sur la configuration et l'optimisation de Red Hat Enterprise Linux. Même si cette version contient aussi des informations sur les capacités des performances de Red Hat Enterprise Linux 5, toutes les instructions présentées ici sont spécifiques à Red Hat Enterprise Linux 6.

Ce livre est divisé en chapitres discutant de sous-systèmes spécifiques dans Red Hat Enterprise Linux. Le *Guide de réglage des performances* se concentre sur trois thèmes majeurs par sous-système :

Fonctionnalités

Chaque chapitre de sous-système décrit des fonctionnalités de performances uniques à (ou implémentées différemment dans) Red Hat Enterprise Linux 6. Ces chapitres traitent aussi des mises à jour de Red Hat Enterprise Linux 6 ayant amélioré les performances de sous-systèmes spécifiques sur Red Hat Enterprise Linux 5.

Analyse

Cet ouvrage énumère aussi les indicateurs de performance de chaque sous-système spécifique. Les valeurs typiques de ces indicateurs sont décrites dans le contexte de services spécifiques, vous aidant à mieux comprendre leurs signification dans des systèmes de production réels.

En outre, le *Guide de réglage des performances* montre aussi différentes manières de récupérer les données des performances (c'est-à-dire le profilage) d'un sous-système. Remarquez que certains outils de profilage présentés ici sont documentés ailleurs avec plus de détails.

Configuration

Les informations les plus importantes dans cet ouvrage sont probablement les instructions sur la manière d'ajuster les performances d'un sous-système spécifique dans Red Hat Enterprise Linux 6. Le *Guide de réglage des performances* explique comment régler précisément un sous-système de Red Hat Enterprise Linux 6 pour des services spécifiques.

N'oubliez pas que le peaufinage des performances d'un sous-système spécifique peut affecter les performances d'un autre sous-système, quelquefois de manière défavorable. La configuration par défaut de Red Hat Enterprise Linux 6 est optimale pour la *plupart* des services exécutés sous des charges *modérées*.

Les procédures énumérées dans le *Guide de réglage des performances* ont été considérablement testées par des ingénieurs de Red Hat dans des laboratoires mais aussi sur le terrain. Cependant, Red Hat recommande que vous testiez correctement toutes les configurations planifiées dans un environnement de test sécurisé avant de les appliquer à vos serveurs de production. Vous devriez aussi effectuer des copies de sauvegarde de toutes vos données et de toutes les informations de votre configuration avant de commencer à régler votre système.

1.1. AUDIENCE

Cet ouvrage conviendra à deux types de lecteurs :

Analystes informatiques ou commerciaux

Cet ouvrage énumère et explique les fonctionnalités des performances de Red Hat Enterprise Linux 6 à un haut niveau, offrant de nombreuses informations sur la performance des sous-systèmes avec des charges de travail spécifiques (par défaut et lorsque optimisé). Le niveau de détail utilisé dans les

descriptions des fonctionnalités des performances de Red Hat Enterprise Linux 6 aide les clients potentiels et ingénieurs de vente à mieux comprendre le caractère approprié de cette plate-forme et son offre de services exigeants en ressources à un niveau acceptable.

Lorsque possible, le *Guide de réglage des performances* fournit aussi des liens vers des documents plus détaillés sur chaque fonctionnalité. À ce niveau de détails, les lecteurs pourront suffisamment comprendre les fonctionnalités de performance pour former une stratégie de haut niveau dans le déploiement et l'optimisation de Red Hat Enterprise Linux 6. Ceci permet aux lecteurs de développer *et* d'évaluer des propositions d'infrastructure.

Ce niveau de documentation, concentré sur les fonctionnalités, convient aux lecteurs possédant une bonne compréhension des sous-systèmes Linux et des réseaux de niveau entreprise.

Administrateur système

Les procédures énumérées dans cet ouvrage conviendront aux administrateurs système possédant un niveau de compétences RHCE [1] (ou son équivalent, de 3 à 5 ans d'expérience dans le déploiement et la gestion de Linux). Le *Guide de réglage des performances* vise à fournir autant de détails que possible sur les effets de chaque configuration ; autrement dit, la description de toute contre-performance pouvant survenir.

Les performances sous-jacentes lors du réglage des performances ne dépend pas de savoir comment analyser et régler un sous-système. Au contraire, un administrateur système adepte du réglage des performance saura comment équilibrer et optimiser un système Red Hat Enterprise Linux 6 *dans un but précis*. Cela implique *aussi* de savoir quelles pénalités et contre-performances sont acceptables lors de la tentative d'implémentation d'une configuration conçue pour stimuler les performances d'un sous-système spécifique.

1.2. ÉVOLUTIVITÉ HORIZONTALE

Les efforts de Red Hat pout améliorer les performances de Red Hat Enterprise Linux 6 se concentrent sur l'*évolutivité*. Les fonctionnalités d'amélioration des performances sont évaluées en se basant principalement sur la manière par laquelle ils affectent les performances de la plate-forme dans différentes zones du spectre de la charge de travail ; c'est-à-dire, à partir du serveur web solitaire et ce jusqu'à l'ordinateur central de la grappe de serveurs.

Mettre l'accent sur l'évolutivité permet à Red Hat Enterprise Linux de maintenir sa versatilité pour différents types de charges de travail et différentes optiques. Cela signifie aussi qu'au fur et à mesure que votre entreprise s'agrandit et que vos charges de travail s'étendent, la reconfiguration de l'environnement de votre serveur sera moins prohibitive (en termes de coût et d'heures de travail) et plus intuitive.

Red Hat a apporté des améliorations à Red Hat Enterprise Linux pour l'*évolutivité horizontale* et l'*évolutivité verticale* ; cependant, l'évolutivité horizontale est généralement le cas le plus applicable. L'idée figurant derrière une évolutivité horizontale consiste à fournir de multiples *ordinateurs standards* pour distribuer de lourdes charges de travail afin d'améliorer les performances et la fiabilité.

Dans une grappe de serveur typique, ces ordinateurs standards se trouvent sous la forme de serveurs montés sur racks 1U et de serveurs blade. Chaque ordinateur standard peut être aussi petit qu'un simple système à deux sockets, même si certaines grappes de serveurs utilisent de grands systèmes avec plus de sockets. Certains réseaux de niveau entreprise possèdent un mélange de systèmes de grande et petite taille. Dans de tels cas, les systèmes de grande taille sont des serveurs de performance (par exemple, des serveurs de bases de données) et ceux de petite taille sont des serveurs d'applications (par exemple, des serveurs web ou des serveurs mail).

Ce type d'évolutivité simplifie la croissance de votre infrastructure informatique : une entreprise de taille moyenne avec une charge de travail correspondante ne nécessitera probablement que deux serveurs « pizza box » pour répondre à tous leurs besoins. Au fur et à mesure que l'entreprise engage plus de personnel, étend ses opérations, augmente le volume de ses ventes, et ainsi de suite, ses besoins en informatique augmenteront en volume et en complexité. Une évolutivité horizontale permettra au service informatique de simplement déployer des machines supplémentaires avec des configurations (principalement) identiques à leurs prédécesseurs.

Pour résumer, l'évolutivité horizontale ajoute une couche d'abstraction qui simplifie l'administration du matériel du système. Grâce au développement de la plate-forme Red Hat Enterprise Linux pour qu'elle s'étende horizontalement, l'augmentation des capacités et des performances des services informatiques peut être aussi simple qu'ajouter de nouvelles machines facilement configurables.

1.2.1. Informatique parallèle

Les utilisateurs ne bénéficient pas uniquement de l'évolutivité horizontale de Red Hat Enterprise Linux parce qu'elle simplifie l'administration du matériel du système, mais aussi parce que l'évolutivité horizontale est une philosophie de développement convenable donné les tendances actuelles dans le développement de matériel.

Prenez ceci en considération : les applications d'entreprise les plus complexes possèdent des milliers de tâches devant être effectuées de manière simultanée, avec différentes méthodes de coordination entre les tâches. Alors que les premiers ordinateurs possédaient un processeur à cœur unique pour jongler entre différentes tâches, presque tous les processeurs disponibles aujourd'hui possèdent de multiples cœurs. En effet, les ordinateurs modernes mettent de multiples cœurs dans un seul socket, faisant ainsi des systèmes à multiples processeurs à partir d'ordinateurs de bureau ou d'ordinateurs portables ne possédant qu'un seul socket.

Depuis 2010, les processeurs standard AMD et Intel sont disponibles avec deux à seize cœurs. Ce type de processeurs sont prévalents dans les serveurs blade ou « pizza box », qui peuvent maintenant contenir jusqu'à 40 cœurs. Ces systèmes à bas coût et à haute performance font entrer les capacités et caractéristiques des systèmes de grande taille dans la vie de tous les jours.

Pour réaliser les meilleures performances et la meilleure utilisation d'un système, chaque cœur doit rester occupé. Cela signifie que 32 différentes tâches doivent être en cours d'exécution pour tirer profit d'un serveur blade de 32 cœurs. Si un châssis blade contient dix de ces 32 cœurs blade, alors l'installation toute entière peut traiter un minimum de 320 tâches simultanément. Si ces tâches font partie d'un travail unique, elles devront être coordonnées.

Red Hat Enterprise Linux a été développé de manière à bien pouvoir s'adapter aux tendances de développement du matériel et pour s'assurer que les entreprises puissent pleinement bénéficier de celles-ci. La Section 1.3, « Systèmes distribués » explore les technologies permettant l'évolutivité horizontale de Red Hat Enterprise Linux avec plus de détails.

1.3. SYSTÈMES DISTRIBUÉS

Pour effectuer une évolutivité horizontale complète, Red Hat Enterprise Linux utilise de nombreux composants d'une *informatique distribuée*. Les technologies composant l'informatique distribuée peuvent être divisées en trois couches :

Communication

Une évolutivité horizontale requiert que de nombreuses tâches soient effectuées simultanément (en parallèle). Ainsi, ces tâches doivent pouvoir effectuer des *communications inter-processus* afin de coordonner leur travail. Par ailleurs, une plate-forme avec une évolutivité horizontale devrait être en mesure de partager des tâches sur de multiples systèmes.

Stockage

Le stockage via des disques locaux n'est pas suffisant pour répondre aux conditions nécessaires à une évolutivité horizontale. Une certaine forme de stockage distribué ou partagé est requise, avec une couche d'abstraction permettant à la capacité d'un volume de stockage unique d'augmenter de manière transparente avec l'ajout de nouveau matériel de stockage.

Gestion

La responsabilité la plus importante dans l'informatique distribuée est la couche de *gestion*. Cette couche de gestion coordonne tous les composants matériels et logiciels, gérant efficacement les communications et le stockage, ainsi que l'utilisation des ressources partagées.

Les sections suivantes décrivent les technologies dans chaque couche avec plus de détails.

1.3.1. Communication

La couche de communication assure le transport des données, elle est composée de deux parties :

- Matériel
- Logiciel

La plus simple (et plus rapide) manière de communiquer pour de multiples systèmes est à travers une *mémoire partagée*. Ceci implique l'utilisation d'opérations de lecture ou écriture familières ; une mémoire partagée possède une bande passante élevée, une latence basse et un faible en-tête d'opérations ordinaires de lecture ou écriture de mémoire.

Ethernet

Pour les ordinateurs, la manière la plus commune de communiquer entre eux est sur Ethernet. De nos jours, GbE (« *Gigabit Ethernet* ») est fourni par défaut sur les systèmes et la plupart des serveurs incluent 2 à 4 ports Gigabit Ethernet. GbE offre une bonne bande passante et latence. Ceci est la fondation des systèmes distribués les plus utilisés de nos jours. Même lorsque des systèmes incluent un matériel réseau plus rapide, il reste commun d'utiliser GbE pour une interface de gestion dédiée.

10GbE

10 Gigabit Ethernet (10GbE) est de plus en plus accepté pour les serveurs haut de gamme et de milieu de gamme. 10GbE fournit dix fois la bande passante de GbE. L'un de ses principaux avantages réside dans ses processeurs multi-cœurs modernes, où est restauré l'équilibre entre les communications et l'informatique. Vous pouvez comparer un système à seul cœur utilisant GbE avec un système à huit cœurs utilisant 10GbE. Utilisé de cette manière, 10GbE est particulièrement utile pour la maintenance des performances générales du système pour éviter la congestion des communications.

Malheureusement, 10GbE est coûteux. Malgré la baisse du coût des cartes d'interfaces réseau NIC 10GbE, le prix de l'interconnexion (particulièrement la fibre optique) reste élevé et les commutateurs réseau 10GbE sont extrêmement coûteux. Nous pouvons supposer que ces prix vont décliner au cours du temps, mais 10GbE est aujourd'hui le moyen le plus utilisé dans les salles de serveur et applications dont les performances sont critiques.

Infiniband

Infiniband offre des performances encore plus élevées que 10GbE. En plus des connexions TCP/IP et UDP utilisées avec Ethernet, Infiniband prend aussi en charge les communications de mémoire partagée. Cela permet à Infiniband de fonctionner entre des systèmes via RDMA (de l'anglais, « *remote direct memory access* »).

L'utilisation de RDMA permet à Infiniband de directement déplacer les données entre les systèmes sans l'en-tête de TCP/IP ou les connexions des sockets. Cela réduit la latence, qui est critique à certaines applications.

Infiniband est le plus souvent utilisé dans les applications HPTC (de l'anglais *High Performance Technical Computing*, ou « Informatique technique de haute performance ») qui nécessitent une bande passante élevée, une faible latence et un faible en-tête. De nombreuses applications de super-ordinateurs bénéficient de cela, au point où le meilleur moyen d'augmenter les performances est d'investir dans Infiniband plutôt que dans des processeurs plus rapides ou dans davantage de mémoire.

RoCCE

RoCCE (« *RDMA over Ethernet* ») implémente des communications dans le style d'Infiniband (y compris RDMA) sur une infrastructure 10GbE. Vu le coût des améliorations associées au volume grandissant des produits 10GbE, il est raisonnable de s'attendre à une augmentation de l'utilisation de RDMA et RoCCE sur un plus large éventail de systèmes et d'applications.

Chacune de ces méthodes de communication est entièrement prise en charge par Red Hat pour une utilisation avec Red Hat Enterprise Linux 6.

1.3.2. Stockage

Un environnement utilisant une informatique distribuée utilise de multiples instances de stockage partagé. Cela peut signifier l'une de deux choses :

- De multiples systèmes stockent des données dans un emplacement unique
- Une unité de stockage (par exemple, un volume) composée de multiples appareils de stockage

L'exemple de stockage le plus familier est le lecteur de disque local monté sur un système. Il est approprié pour les opérations informatiques dans lesquelles toutes les applications sont hébergées sur un seul hôte, ou même sur un petit nombre d'hôtes. Cependant, lorsque l'infrastructure s'étend sur des douzaines ou même sur des centaines de systèmes, la gestion des disques de stockages locaux devient difficile et compliquée.

Le stockage distribué ajoute une couche pour faciliter et automatiser l'administration du matériel de stockage en tant que balances commerciales. Posséder de multiples systèmes partageant une poignée d'instances de stockage réduit le nombre de périphériques que l'administrateur doit gérer.

La consolidation des capacités de stockage de multiples appareils de stockage en un seul volume aide les utilisateurs et les administrateurs. Ce type de stockage distribué offre une couche d'abstraction aux pools de stockage : les utilisateurs voient une seule unité de stockage, ce qu'un administrateur peut facilement augmenter en ajoutant du matériel. Certaines technologies activant le stockage distribué offrent aussi des bénéfices supplémentaires, comme le basculement (« failover ») et le « multipathing ».

NFS

NFS (« *Network File System* ») permet à de multiples serveurs ou utilisateurs de monter et utiliser la même instance de stockage distant via TCP ou UDP. NFS est communément utilisé pour conserver les données partagées par de multiples applications. NFS est aussi pratique pour le stockage en vrac de grandes quantités de données.

SAN

Les réseaux de stockage SAN (« *Storage Area Networks* ») utilisent le protocole Fibre Channel ou iSCSI afin d'offrir un accès distant au stockage. L'infrastructure Fibre Channel (comme les adaptateurs de bus hôte Fibre Channel, commutateurs et matrices de stockage) combinent hautes performances, une

bande passante importante et un stockage massif. Les réseaux de stockage SAN séparent le stockage du traitement, offrant ainsi une importante flexibilité dans la conception du système.

L'autre principal avantage des réseaux de stockage SAN réside dans le fait qu'ils fournissent un environnement de gestion pour effectuer des tâches d'administration du matériel de stockage majeures. Ces tâches incluent :

- Contrôle de l'accès au stockage
- Gestion de grandes quantités de données
- Approvisionnement des systèmes
- Sauvegarde et réplication de données
- Prise d'instantanés
- Prise en charge du basculement (« failover ») de systèmes
- Assurance de l'intégrité des données
- Migration des données

GFS2

Le système de fichiers Red Hat *Global File System 2* (GFS2) offre plusieurs capacités spécialisées. La fonction de base de GFS2 est d'offrir un système de fichiers unique, y compris l'accès simultané lecture/écriture, partagé sur de multiples membres d'un cluster. Cela signifie que chaque membre du cluster voit exactement les mêmes données « sur disque » dans le système de fichiers GFS2.

GFS2 permet à tous les systèmes d'avoir un accès simultané au « disque ». Pour maintenir l'intégrité des données, GFS2 utilise DLM (« *Distributed Lock Manager* »), qui autorise uniquement à un système d'écrire sur un emplacement spécifique à la fois.

GFS2 est particulièrement adapté aux applications de basculement qui nécessitent une haute disponibilité du stockage.

Pour obtenir des informations supplémentaires sur GFS2, veuilles consulter le guide *Global File System 2*. Pour obtenir davantage d'informations sur le stockage en général, veuillez consulter le *Guide d'administration du stockage*. Ces deux ouvrages peuvent être trouvés sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

1.3.3. Réseaux convergents

Les communications sur le réseau sont habituellement effectuées à travers Ethernet et le trafic du stockage est effectué à l'aide d'un environnement SAN Fibre Channel dédié. Il est commun de posséder un réseau ou lien sériel dédié pour la gestion de systèmes et même un *heartbeat*[2]. Par conséquent, un seul serveur se trouve habituellement sur de multiples réseaux.

Fournir de multiples connexions sur chaque serveur est coûteux, encombrant et complexe à gérer. Ceci a donné lieu au besoin de trouver une manière de consolider toutes les connexions en une seule. FCoE (de l'anglais, « *Fibre Channel over Ethernet* ») et iSCSI (« *Internet SCSI* ») ont répondu à ce besoin.

FCoE

Avec FCoe, les commandes Fibre Channel standard et les paquets de données sont transportés sur une infrastructure 10GbE via une CNA (*converged network card*, ou « carte réseau convergente »). Le trafic Ethernet TCP/IP standard et les opérations de stockage Fibre Channel peuvent être transportés via le

même lien. FCoE utilise une carte d'interface réseau physique (et un câble) pour de multiples connexions de stockage ou réseau logique.

FCoE offre les avantages suivants :

Nombre de connexions réduit

FCoE réduit de moitié le nombre de connexions réseau à un serveur. Vous pouvez toujours choisir d'avoir de multiples connexions pour les performances ou pour la disponibilité ; cependant, une seule connexion offre une connectivité stockage et réseau. Ceci est particulièrement utile pour les serveurs « Pizza box » et les serveurs « Blade », puisqu'ils ont tous deux un espace très limité pour les composants.

Moindre coût

Un nombre de connexions réduit signifie un nombre immédiatement réduit de câbles, d'interrupteurs et autres équipements de réseau. L'histoire d'Ethernet démontre aussi des économies de grande taille ; le coût des réseaux a dramatiquement diminué alors que le nombre de périphériques sur le marché est passé de plusieurs millions à des milliards, tout comme le déclin des prix des périphériques 100 Mo Ethernet et gigabit Ethernet a pu être observé.

De manière similaire, 10GbE deviendra aussi moins coûteux alors que davantage d'entreprises s'adapteront à son utilisation. Aussi, tandis que le matériel CNA est intégré en une seule puce, son utilisation de masse augmentera sur le marché, ce qui, au cours du temps, aura pour conséquence la diminution significative de son prix.

iSCSI

iSCSI (« *Internet SCSI* ») est un autre type de protocole de réseau convergé, une alternative a FCoE. Tout comme Fibre Channel, iSCSI fournit un stockage de niveau bloc sur un réseau. Cependant, iSCSI ne fournit pas d'environnement de gestion complet. Le principal avantage d'iSCSI par rapport à FCoE est qu'iSCSI offre la plupart des capacités et de la flexibilité de Fibre Channel, mais pour un moindre coût.

---

[1] « Red Hat Certified Engineer » (Ingénieur certifié Red Hat). Pour obtenir des informations supplémentaires, veuillez consulter http://www.redhat.com/training/certifications/rhce/.

[2] *Heartbeat* (battement de cœur) est l'échange de messages entre des systèmes pour s'assurer que chaque système fonctionne toujours. Si un système n'a plus de « battements de cœur », il est supposé que ce système a échoué et est éteint et qu'un autre système a pris la relève.

CHAPITRE 2. CARACTÉRISTIQUES DES PERFORMANCES DE RED HAT ENTERPRISE LINUX 6

2.1. PRISE EN CHARGE 64 BITS

Red Hat Enterprise Linux 6 prend en charge les processeurs 64 bits ; ces processeurs peuvent, en théorie, utiliser jusqu'à 16 *exaoctets* de mémoire. À partir de sa mise à disponibilité générale (« GA »), Red Hat Enterprise Linux 6 a été testé et certifié pour une prise en charge allant jusqu'à 8 To de mémoire physique.

Il est prévu que la taille de la mémoire prise en charge par Red Hat Enterprise Linux 6 augmente avec plusieurs mises à jour mineures puisque Red Hat continue d'introduire et de d'améliorer des fonctionnalités permettant l'utilisation de blocs mémoire plus grands. Voici quelques exemples de ces améliorations (à partir de la date de mise à disponibilité générale de Red Hat Enterprise Linux 6) :

- Huge pages et transparent huge pages
- Améliorations de l'accès mémoire non-uniforme (NUMA)

Ces améliorations sont décrites avec plus de détails dans les sections qui suivent.

Huge pages et transparent huge pages

L'implémentation de « *huge pages* » dans Red Hat Enterprise Linux 6 permet au système de gérer l'utilisation de la mémoire efficacement sur de multiples charges de travail de mémoire. Les « Huge pages » utilisent dynamiquement des pages de 2 Mo au lieu de la taille de page standard de 4 Ko, permettant aux applications de bien s'adapter au traitement des gigaoctets et même aux téraoctets de mémoire.

Les « huge pages » sont difficiles à créer, gérer et utiliser manuellement. Pour répondre à ce problème, Red Hat Enterprise Linux propose l'utilisation de THP (« *transparent huge pages* »). THP gère automatiquement de nombreuses complexités impliquées dans l'utilisation des « huge pages ».

Pour plus d'informations sur « huges pages » et THP, veuillez vous reportez à la Section 5.2, « « Huge Pages » et « Transparent Huge Pages » ».

Améliorations NUMA

De nombreux systèmes prennent maintenant en charge NUMA (« *Non-Uniform Memory Access* »). NUMA simplifie le design et la création de matériel pour des systèmes de grande taille ; cependant, une couche de complexité est aussi ajoutée au développement de l'application. Par exemple, NUMA implémente une mémoire locale et une mémoire distante, la mémoire distante peut prendre plusieurs fois plus de temps à accéder que la mémoire locale. Cette fonctionnalité (entre autres) comprend de nombreuses implications quant à la performance qui ont un impact sur les systèmes d'exploitation, les applications et les configurations de systèmes devant être déployés.

Red Hat Enterprise Linux 6 est mieux optimisé pour l'utilisation de NUMA, grâce à plusieurs fonctionnalités qui aident à gérer les utilisateurs et les applications sur les systèmes NUMA. Ces fonctionnalités incluent les affinités de CPU, le pinning de CPU (cpusets), numactl et les groupes de contrôle, qui permettent à un processus (affinité) ou à une application (pinning) de se « lier » à un CPU ou à un ensemble de CPU spécifique(s).

Pour obtenir des informations supplémentaires sur la prise en charge de NUMA dans Red Hat Enterprise Linux 6, reportez-vous à la Section 4.1.1, « Topologie des CPU et de NUMA ».

2.2. TICKET SPINLOCKS

Une partie clé de tout design de système consiste à s'assurer qu'aucun processus n'altère la mémoire utilisée par un autre processus. Des changements incontrôlés dans la mémoire peuvent provoquer la corruption de données et l'échec du système. Pour prévenir ceci, le système d'exploitation autorise un processus à verrouiller une portion de mémoire, à effectuer une opération, puis à déverrouiller, ou « libérer » la mémoire.

Une implémentation commune du verrouillage de la mémoire est avec les *spin locks* (ou verrous tournants), qui permettent à un processus de continuer à vérifier si un verrou est disponible et de prendre le verrou dès qu'il devient disponible. Si de multiples processus sont en compétition pour le même verrou, le premier à requérir le verrou une fois qu'il a été « libéré » l'obtiendra. Lorsque tous les processus ont le même accès à la mémoire, cette approche est « juste » et fonctionne plutôt bien.

Malheureusement, sur un système NUMA, tous les processus n'ont pas le même accès aux verrous. Les processus sur le même nœud NUMA que le verrou ont un avantage injuste pour obtenir le verrou. Les processus sur des nœuds NUMA distants seront en manque de verrous et leurs performances en seront dégradées.

Pour répondre à ce problème, Red Hat Enterprise Linux a implémenté « *ticket spinlocks* ». Cette fonctionnalité ajoute un mécanisme de file de réservation au verrou, permettant ainsi à *tous* les processus de prendre un verrou dans l'ordre dans lequel ils ont été requis. Cela élimine les problèmes de chronologie et les avantages injustes dans les requêtes de verrous.

Lorsqu'un « ticket spinlock » possède légèrement plus de temps système qu'un verrou tournant ordinaire, il évolue mieux et fournit des meilleures performances sur les systèmes NUMA.

2.3. STRUCTURE DE LISTE DYNAMIQUE

Le système d'exploitation requiert un ensemble d'informations sur chaque processeur dans le système. Dans Red Hat Enterprise Linux 5, cet ensemble d'informations était alloué à une matrice de taille fixe dans la mémoire. Des informations sur chaque processeur individuel étaient obtenues en effectuant un index dans cette matrice. Cette méthode était rapide, facile et directe pour les systèmes contenant relativement peu de processeurs.

Cependant, au fur et à mesure que le nombre de processeurs d'un système a augmenté, cette méthode a produit des en-têtes significatifs. Comme la matrice à taille fixe dans la mémoire est une ressource unique partagée, elle peut devenir une source de congestion car davantage de processeurs tentent d'y accéder au même moment.

Pour répondre à ce problème, Red Hat Enterprise Linux 6 utilise une *structure de liste dynamique* pour les informations des processeurs. Ceci permet à la matrice utilisée pour les informations des processeurs d'être allouée de manière dynamique : si uniquement huit processeurs se trouvent dans le système, alors seules huit entrées sont créées dans la liste. Si 2048 processeurs se trouvent dans le système, alors 2048 entrées seront créées aussi.

Une structure de liste dynamique permet un verrouillage à granularité fine. Par exemple, si les informations doivent être mises à jour au même moment pour les processeurs 6, 72, 183, 657, 931 et 1546, ceci peut être effectué avec un meilleur parallélisme. Bien évidemment, de telles situations se produisent plus fréquemment sur des systèmes de haute performance que sur des systèmes de petite taille.

2.4. NOYAU TICKLESS

Dans les précédentes versions de Red Hat Enterprise Linux, le noyau utilisait un mécanisme basé sur un minuteur qui produisait une interruption système de manière continuelle. À chaque interruption, le système effectuait un *sondage* ; autrement dit, il vérifiait si du travail devait être effectué.

En fonction de ce paramètre, cette interruption système, ou « *timer tick* » (tic du minuteur ou de l'horloge) pouvait se produire plusieurs centaines ou milliers de fois par seconde. Ceci se produisait chaque seconde, peu importe la charge de travail du système. Sur un système légèrement chargé, cela a un impact sur la *consommation d'électricité* en empêchant au processeur d'utiliser les états de veille de manière efficace. Le système utilise moins d'électricité lorsqu'il est en état de veille.

La manière d'opérer la plus économe en énergie pour un système est d'effectuer le travail le plus rapidement possible, aller dans l'état de veille le plus profond possible, puis dormir aussi longtemps que possible. Pour implémenter cela, Red Hat Enterprise Linux 6 utilise un noyau sans tic (« *tickless kernel* »). Avec ceci, le minuteur d'interruptions a été supprimé de la boucle d'inactivité, transformant ainsi Red Hat Enterprise Linux 6 en un environnement complètement géré par interruptions.

Le noyau sans tic (« tickless kernel ») permet au système d'aller en état de veille profond pendant les moments d'inactivité et de répondre rapidement lorsque des tâches doivent être effectuées.

Pour obtenir des informations supplémentaires, veuillez consulter le *Guide de gestion de l'alimentation*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

2.5. GROUPES DE CONTRÔLE

Red Hat Enterprise Linux fournit de nombreuses options utiles au réglage des performances. Les systèmes de grande taille, allant jusqu'à des centaines de processeurs, peuvent être réglé afin de délivrer de superbes performances. Mais le réglage de tels systèmes requiert une expertise considérable ainsi qu'une charge de travail bien définie. Lorsque les systèmes de grande taille étaient chers et moins nombreux, il était acceptable de leur offrir un traitement de faveur. Maintenant que ces systèmes sont couramment utilisés, des outils plus efficaces sont nécessaires.

Pour compliquer les choses, des systèmes plus puissants sont maintenant utilisés pour la consolidation de services. Des charges de travail qui étaient avant exécutées sur quatre à huit serveurs sont maintenant placées sur un seul serveur. Et comme mentionné plus tôt dans la Section 1.2.1, « Informatique parallèle », de nos jours, de nombreux systèmes à moyenne portée contiennent davantage de cœurs que les machines à haute performance d'hier.

De nombreuses applications modernes sont conçues pour le traitement parallèle, utilisant de multiples threads ou processus pour améliorer les performances. Cependant, peu d'applications peuvent faire usage de plus de huit threads de manière efficace. Ainsi, de multiples applications doivent habituellement être installées sur des systèmes à 32 CPU pour maximiser la capacité.

Prenez en considération la situation suivante : des systèmes de petite taille et peu coûteux sont maintenant à égalité avec les performances des machines chères et de haute performance d'hier. Les machines moins chères de haute performance ont offert aux architectes la possibilité de consolider davantage de services sur moins de machines.

Cependant, certaines ressources (telles que les E/S et les communications réseau) sont partagées et n'augmentent pas à la même vitesse que le compte des CPU. Ainsi, un système hébergeant de multiples applications peut expérimenter des performances générales dégradées lorsqu'une application s'accapare trop d'une ressource en particulier.

Pour répondre à ce problème, Red Hat Enterprise Linux 6 prend maintenant en charge les *groupes de contrôle* (cgroups). Les cgroups permettent aux administrateurs d'allouer des ressources aux tâches spécifiques selon les besoins. Cela signifie par exemple être en mesure d'allouer 80% de quatre CPU,

60 Go de mémoire et 40% des E/S de disque à une application de base de données. Une application web exécutée sur le même système pourrait recevoir deux CPU, 2 Go de mémoire et 50% de la bande passante du réseau.

Par conséquent, les applications de la base de données et du web délivreront de meilleures performances car le système les empêchera de consommer les ressources système de manière excessive. En outre, de nombreux aspects des cgroups se *règlent automatiquement*, permettant au système de répondre conformément aux changements de la charge de travail.

Un cgroup possède deux composants majeurs :

- Une liste de tâches assignées au cgroup
- Des ressources allouées à ces tâches

Les tâches assignées au cgroup sont exécutées *dans* le cgroup. Toute tâche enfant engendrée sera aussi exécutée dans le cgroup. Cela permet à un administrateur de gérer une application entière en tant qu'unité unique. Un administrateur peut aussi configurer des allocations pour les ressources suivantes :

- CPUsets
- Mémoire
- E/S
- Réseau (bande passante)

Dans les CPUsets, les cgroups permettent aux administrateurs de configurer le nombre de CPU, les affinités pour des CPU ou nœuds[3] spécifiques et combien de temps CPU est utilisé par un ensemble de tâches. L'utilisation des cgroups pour configurer des CPUsets est vitale pour assurer de bonnes performances générales, empêchant ainsi à une application de consommer trop de ressources au détriment d'autres tâches tout en s'assurant simultanément que l'application ne manque pas de temps CPU.

Les bandes passantes des E/S et du réseau sont gérées par d'autres contrôleurs de ressources. Les contrôleurs de ressources vous permettent de déterminer quelle quantité de bande passante peut être consommée par les tâches dans un cgroup et de s'assurer que les tâches dans un cgroup ne consomment pas trop de ressources et n'en manquent pas non plus.

Les cgroups permettent à l'administrateur de définir et d'allouer, à un haut niveau, les ressources systèmes dont diverses applications ont consommé et vont consommer. Le système gère et équilibre ensuite de manière automatique les diverses applications, délivrant de bonnes performances prévisibles et optimisant les performances du système en général.

Pour obtenir des informations supplémentaires sur la manière d'utiliser les groupes de contrôle, veuillez consulter le *Guide de gestion des ressources*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

2.6. AMÉLIORATIONS DU STOCKAGE ET DES SYSTÈMES DE FICHIERS

Red Hat Enterprise Linux 6 présente aussi plusieurs améliorations à la gestion du stockage et des systèmes de fichiers. Deux des plus importantes avancées dans cette version sont les prises en charge d'ext4 et de XFS. Pour obtenir une couverture plus complète des améliorations des performances liées au stockage et aux systèmes de fichiers, veuillez consulter le Chapitre 7, *Systèmes de fichiers*.

Ext4

Ext4 est le système de fichiers par défaut de Red Hat Enterprise Linux 6. Il est la version de quatrième génération de la gamme de systèmes de fichiers EXT, prenant en charge un système de fichiers d'une taille maximale théorique de 1 exaoctet, et un fichier unique d'une taille maximale de 16 To. Red Hat Enterprise Linux 6 prend en charge un système de fichiers d'une taille maximale de 16 To et un fichier unique d'une taille maximale de 16 To. Hormis sa capacité de stockage bien plus grande, ext4 inclut aussi plusieurs autres nouvelles fonctionnalités, comme :

- Des métadonnées basées sur extensions
- L'allocation différée
- Le checksum de journaux

Pour obtenir des informations sur le système de fichiers ext4, veuillez consulter la Section 7.3.1, « Le système de fichiers Ext4 ».

XFS

XFS est un système de fichiers de journalisation de 64 bits, robuste et mature, qui prend en charge des fichiers et systèmes de fichiers de très grande taille sur un hôte unique. À l'origine, ce système de fichiers fût développé par SGI et possède une longue histoire d'exécution sur des serveurs et des matrices de stockage extrêmement grands. XFS inclut :

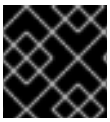
- L'allocation différée
- Des inodes alloués de manière dynamique
- Une indexation d'arborescence B pour l'évolutivité de la gestion de l'espace libre
- La défragmentation en ligne et l'agrandissement du système de fichiers
- Des algorithmes de lecture à l'avance de métadonnées sophistiqués

Même si XFS peut s'étendre à des exaoctets, la taille de système de fichiers XFS maximale prise en charge par Red Hatest de 100 To. Pour obtenir des informations supplémentaires sur XFS, veuillez consulter la Section 7.3.2, « Le système de fichiers XFS ».

Lecteurs de démarrage de grande taille

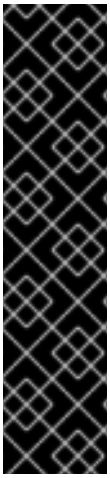
Un BIOS traditionnel prend en charge un disque d'une taille maximale de 2,2 To. Les systèmes Red Hat Enterprise Linux 6 qui utilisent BIOS peuvent prendre en charge des disques de plus de 2,2 To en utilisant une nouvelle structure de disque appelée GPT (« *Global Partition Table* », table de partitionnement global). GPT peut uniquement être utilisé pour les disques de données, il ne peut pas être utilisé pour les lecteurs de démarrage avec BIOS. Ainsi, les lecteurs de démarrage peuvent uniquement faire une taille maximale de 2,2 To. À l'origine, BIOS avait été créé pour le PC IBM ; tandis que BIOS a considérablement évolué afin de s'adapter au matériel moderne, UEFI (« *Unified Extensible Firmware Interface* ») a été conçu pour prendre en charge le matériel nouveau et émergent.

Red Hat Enterprise Linux 6 prend aussi en charge UEFI, qui peut être utilisé pour remplacer BIOS (toujours pris en charge). Les systèmes avec UEFI exécutant Red Hat Enterprise Linux 6 permettent l'utilisation de GPT et de partitions de 2,2 To (et plus) pour les partitions de démarrage et des données.



IMPORTANT

Red Hat Enterprise Linux 6 ne prend pas en charge UEFI pour les systèmes x86 32-bit.



IMPORTANT

Remarquez que les configurations de démarrage d'UEFI et de BIOS sont significativement différentes. Ainsi, le système installé doit démarrer en utilisant le même microprogramme que celui qui avait été utilisé lors de l'installation. Vous ne pouvez pas installer le système d'exploitation sur un système qui utilise BIOS puis démarrer cette installation sur un système qui utilise UEFI.

Red Hat Enterprise Linux 6 prend en charge la version 2.2 de la spécification UEFI. Le matériel qui prend en charge la version 2.3 de la spécification UEFI et les versions plus récentes devrait démarrer et exécuter Red Hat Enterprise Linux 6, mais les fonctionnalités supplémentaires définies par ces spécifications ne seront pas disponibles. Les spécifications UEFI sont disponibles sur http://www.uefi.org/specs/agreement/.

[3] Un nœud est généralement défini comme un ensemble de CPU ou de cœurs dans un socket.

CHAPITRE 3. CONTRÔLE ET ANALYSE DES PERFORMANCES DU SYSTÈME

Ce chapitre présente brièvement les outils pouvant être utilisés pour contrôler et analyser les performances du système et des applications et décrit les situations dans lesquelles chaque outil est le plus utile. Les données collectées peuvent révéler une certaine congestion ou d'autres problèmes de système contribuant à des performances non-optimales.

3.1. LE SYSTÈME DE FICHIERS PROC

Le « système de fichiers » **proc** est un répertoire contenant une hiérarchie de fichiers qui représentent l'état actuel du noyau Linux. Il permet aux applications et utilisateurs de voir la vue qu'a le noyau du système.

Le répertoire **proc** contient aussi des informations sur le matériel du système et sur tout processus actuellement en cours d'exécution. La plupart de ces fichiers sont en lecture-seule, mais certains fichiers (principalement ceux en **/proc/sys**) peuvent être manipulés par les utilisateurs et les applications afin de communiquer les changements de configuration au noyau.

Pour obtenir des informations supplémentaires sur l'affichage et la modification de fichiers dans le répertoire **proc**, veuillez consulter le *Guide de déploiement*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.2. MONITEURS SYSTÈME GNOME ET KDE

Les environnements de bureau GNOME et KDE possèdent tous deux des outils graphiques pour vous assister dans le contrôle et la modification de votre système.

Moniteur système GNOME

Le **Moniteur système GNOME** affiche les informations système de base et vous permet de contrôler les processus du système, ainsi que l'utilisation des ressources et du système de fichiers. Ouvrez-le avec la commande **gnome-system-monitor** dans le **Terminal**, ou cliquez sur le menu **Applications** et sélectionnez **Outils système** > **Moniteur système**.

Le **Moniteur système GNOME** possède quatre onglets :

Système

Affiche des informations de base sur les logiciels et le matériel de l'ordinateur.

Processus

Affiche les processus actifs et les relations entre ces processus, ainsi que des informations détaillées sur chaque processus. Vous permet aussi de filtrer les processus affichés et d'effectuer certaines actions sur ces processus (« start », « stop », « kill », « change priority », etc...).

Ressources

Affiche le temps actuel d'utilisation du CPU, d'espace mémoire, d'espace swap et du réseau.

Systèmes de fichiers

Répertorie tous les systèmes de fichiers montés aux côtés d'informations de base sur chacun d'entre eux, comme le type de système de fichiers, le point de montage et l'utilisation de la mémoire.

Pour obtenir des informations supplémentaires sur le **Moniteur système GNOME**, veuillez consulter le menu d'**Aide** dans l'application, ou le *Guide de déploiement*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

KDE System Guard

Le **KDE System Guard** vous permet de contrôler la charge du système et les processus actuellement en cours. Il vous permet aussi d'effectuer des actions sur les processus. Ouvrez-le avec la commande **`ksysguard`** dans le **Terminal**, ou cliquez sur le **Kickoff Application Launcher** et sélectionnez **Applications** > **Système** > **Moniteur système**.

KDE System Guard possède deux onglets :

Table des processus

Affiche une liste de tous les processus en cours d'exécution, cette liste est alphabétique par défaut. Vous pouvez aussi classer les processus selon un certain nombre d'autres propriétés, dont la totalité de l'utilisation du CPU, l'utilisation de mémoire physique ou partagée, le propriétaire et la priorité. Vous pouvez aussi filtrer les résultats visibles, rechercher des processus spécifiques ou effectuer certaines actions sur un processus.

Charge du système

Affiche des graphes historiques de l'utilisation du CPU, de la mémoire et de l'espace swap, ainsi que du réseau. Passez sur les graphes pour une analyse détaillée et les clés du graphe.

Pour obtenir des informations supplémentaires sur **KDE System Guard**, veuillez consulter le menu d'**Aide** dans l'application.

3.3. OUTILS DE CONTRÔLE INTÉGRÉS SUR LIGNE DE COMMANDE

En plus des outils de contrôle graphique, Red Hat Enterprise Linux fournit plusieurs outils pouvant être utilisés pour contrôler un système à partir de la ligne de commande. L'avantage de ces outils est qu'ils peuvent être utilisés hors du niveau d'exécution 5. Cette section discute brièvement de chaque outil et suggère les cas d'application auxquels chaque outil convient le mieux.

`top`

L'outil **top** offre un affichage dynamique et en temps réel des processus dans un système en cours d'exécution. Il peut afficher tout un éventail d'informations, y compris un sommaire du système et les tâches actuellement gérées par le noyau Linux. Il possède aussi la capacité limitée de manipuler les processus. Son opération et les informations qu'il affiche sont hautement configurables et tous les détails de configuration peuvent être rendus persistants à travers les redémarrages.

Par défaut, les processus affichés sont classés par le pourcentage d'utilisation du CPU, offrant ainsi un aperçu facile des processus consommant le plus de ressources.

Pour obtenir des informations détaillées sur l'utilisation de **top**, veuillez consulter sa page man : **`man top`**.

`ps`

L'outil **ps** prend un instantané d'un groupe sélectionné de processus actifs. Par défaut, ce groupe est limité aux processus appartenant à l'utilisateur actuel et associés au même terminal.

Cet outil peut fournir des informations sur les processus plus détaillées que **top**, mais il n'est pas dynamique.

Pour obtenir des informations détaillées sur l'utilisation de **ps**, veuillez consulter sa page man : **man ps**.

vmstat

vmstat (« Virtual Memory Statistics », statistiques de mémoire virtuelle) émets instantanément des rapports sur les processus de votre système, la mémoire, la pagination, les E/S de bloc, les interruptions et sur l'activité du CPU.

Même s'il n'est pas dynamique comme **top**, vous pouvez spécifier un intervalle d'échantillonnage, qui vous permettra d'observer l'activité du système en temps quasi réel.

Pour obtenir des informations détaillées sur l'utilisation de **vmstat**, veuillez consulter sa page man : **man vmstat**.

sar

sar (« System Activity Reporter », rapporteur d'activité système) collecte et rapporte des informations sur l'activité du système du jour jusqu'à présent. La sortie par défaut couvre l'utilisation du CPU pendant la journée à dix minutes d'intervalle, ce à partir du commencement de la journée.

```
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
...
```

Cet outil est une alternative utile à la création de rapports périodiques sur l'activité du système avec **top** ou autres outils similaires.

Pour obtenir des informations détaillées sur l'utilisation de **sar**, veuillez consulter sa page man : **man sar**.

3.4. TUNED ET KTUNE

Tuned est un démon qui contrôle et collecte les données sur l'utilisation de divers composants d'un système et utilise ces informations pour régler les paramètres système dynamiquement, en fonction des besoins. Il peut réagir aux modifications d'utilisation du CPU et du réseau et ajuster les paramètres pour améliorer les performances des périphériques actifs ou réduire la consommation de l'alimentation des périphériques inactifs.

ktune, qui l'accompagne, s'utilise en partenariat avec l'outil **tuned-adm** afin de fournir un certain nombre de profils de réglages pré-configurés pour améliorer les performances et réduire la consommation de l'alimentation dans un certain nombre de cas d'utilisation spécifiques. Modifiez ces profils ou créez de nouveaux profils pour créer des solutions de performance adaptées à votre environnement.

Les profils offerts dans le cadre de **tuned-adm** incluent :

default

Profil d'économie d'énergie par défaut. Ce profil est le profil d'économies d'énergie le plus basique. Il active uniquement les plugins du disque et du CPU. Remarquez qu'il ne s'agit pas de la même chose qu'éteindre **tuned-adm**, dans quel cas **tuned** et **ktune** seraient tous deux désactivés.

latency-performance

Profil de serveur pour optimisation typique des performances de la latence. Les mécanismes d'économies d'énergie de **tuned** et **ktune** sont désactivés. Le mode **cpuspeed** bascule sur **performance**. L'élévateur d'entrée et sorties est modifié sur **deadline** pour chaque périphérique. Pour une meilleure qualité de service de la gestion de l'alimentation, la condition préalable **0** de **cpu_dma_latency** est enregistrée.

throughput-performance

Profil de serveur pour une optimisation typique des performances de débit. Ce profil est recommandé si le système ne possède pas de stockage de classe entreprise. Ce profil est le même que **latency-performance**, à l'exception de :

- ***kernel.sched_min_granularity_ns*** (granularité de préemption minimale de l'ordonnanceur), qui est paramétré que **10** millisecondes,
- ***kernel.sched_wakeup_granularity_ns*** (granularité de réveil de l'ordonnanceur), qui est paramétré sur **15** millisecondes,
- ***vm.dirty_ratio*** (ratio dirty de la machine virtuelle), qui est paramétré sur 40% et
- les huge pages transparentes sont activées.

enterprise-storage

Ce profil est recommandé pour les configurations de serveurs de taille entreprise avec un stockage de classe entreprise, incluant une protection de cache de contrôleur sur batterie et la gestion de caches sur disque. Ce profil est le même que le profil **throughput-performance**, mais inclut un ajout : les systèmes de fichiers sont montés à nouveau avec **barrier=0**.

virtual-guest

Ce profil est recommandé pour les configurations de serveurs de taille entreprise avec un stockage de classe entreprise, incluant une protection de cache de contrôleur sur batterie et la gestion de caches sur disque. Ce profil est le même que **throughput-performance**, à l'exception de :

- La valeur ***readahead***, qui est paramétrée sur **4x** et
- Les systèmes de fichiers non root/boot sont montés à nouveau avec **barrier=0**.

virtual-host

Basé sur le profil ***enterprise-storage***, ***virtual-host*** réduit aussi le « swappiness » de la mémoire virtuelle et active une réécriture plus agressive des pages modifiées. Ce profil est disponible sur Red Hat Enterprise Linux 6.3 et ses versions supérieures et est le profil recommandé pour les hôtes de virtualisation, y compris KVM et les hôtes Red Hat Enterprise Virtualization.

Pour obtenir des informations supplémentaires sur **tuned** et **ktune**, veuillez consulter le *Guide de gestion de l'alimentation* Red Hat Enterprise Linux 6, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.5. PROFILEURS D'APPLICATION

Un profilage est le processus de rassemblement d'informations sur le comportement d'un programme pendant qu'il l'exécute. On peut profiler une application pour déterminer quelles zones d'un programme

peuvent être optimisées pour améliorer la vitesse générale du programme, réduire l'utilisation de la mémoire, etc. Les outils de profilage d'applications aident à simplifier ce processus.

Trois outils de profilage sont pris en charge pour une utilisation avec Red Hat Enterprise Linux 6 : **SystemTap**, **OProfile** et **Valgrind**. Documenter ces outils de profilage est hors de la portée de ce guide ; cependant, cette section offre des liens vers des informations supplémentaires ainsi qu'un bref aperçu des tâches auxquelles conviennent chacun de ces profileurs.

3.5.1. SystemTap

SystemTap est un outil de traçage et de sondage qui permet aux utilisateurs de contrôler et d'analyser minutieusement les activités du système d'exploitation (notamment les activités du noyau). Il fournit des informations similaires à la sortie d'outils comme **netstat**, **top**, **ps**, et **iostat**. Cependant, SystemTap est conçu pour fournir davantage d'options de filtrage et d'analyse sur les informations collectées.

SystemTap offre une analyse bien plus profonde et précise des activités du système et du comportement de l'application afin de vous permettre d'identifier les goulots d'étranglement du système et de l'application.

Le plugin « Function Callgraph » pour Eclipse utilise SystemTap comme serveur d'arrière-plan, lui permettant de contrôler minutieusement le statut d'un programme, y compris les appels de fonctions, retours, horodatages et variables de l'espace utilisateur. Les informations sont affichées visuellement pour une optimisation plus facile.

Pour obtenir des informations supplémentaires sur SystemTap, veuillez consulter le *Guide du débutant SystemTap*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.5.2. OProfile

OProfile (**oprofile**) est un outil de contrôle des performances système globales. Il utilise le matériel de contrôle des performances dédié du processeur pour récupérer des informations sur les exécutables du système et du noyau, comme lorsque la mémoire est référencée, mais aussi le nombre de caches L2 et le nombre d'interruptions de matériel reçues. Il peut aussi être utilisé pour déterminer l'utilisation du processeur et quels services et applications sont le plus utilisés.

OProfile peut aussi être utilisé avec Eclipse via le plugin OProfile Eclipse. Ce plugin permet aux utilisateurs de facilement déterminer les zones de code prenant le plus de temps et d'effectuer toutes les fonctions de ligne de commande OProfile avec une riche visualisation des résultats.

Cependant, les utilisateurs doivent être conscient de plusieurs limitations d'OProfile :

- Les échantillons de contrôle des performances peuvent ne pas être précis. Comme le processeur peut exécuter des instructions issues d'un ordre, un échantillon peut être enregistré à partir d'une instruction avoisinante, au lieu d'à partir de l'instruction ayant avoisiné l'interruption.

- Comme OProfile est global et suppose que des processus démarrent et s'arrêtent de multiples fois, les échantillons de multiples exécutions sont autorisés à s'accumuler. Cela signifie que vous devrez probablement supprimer les données des échantillons des précédentes exécutions.

- Il se concentre sur l'identification des problèmes de processus limités par les CPU et ainsi, il n'identifie pas les processus qui sont en veille pendant l'attente de verrous pour d'autres événements.

Pour obtenir des informations supplémentaires sur l'utilisation d'OProfile, veuillez consulter le *Guide de déploiement*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, ou

consultez la documentation **oprofile** de votre système, située dans **/usr/share/doc/oprofile-<*version*>**.

3.5.3. Valgrind

Valgrind offre un certain nombre d'outils de détection et de profilage pour aider à améliorer les performances et la justesse de vos applications. Ces outils peuvent détecter les erreurs de mémoire et les erreurs liées aux threads, ainsi que les dépassements de capacité de tas, de pile(s) et de matrice(s), vous permettant ainsi de facilement identifier et corriger des erreurs dans le code de votre application. Ils peuvent aussi profiler le cache, le tas et la prédiction de branches pour identifier les facteurs pouvant augmenter la vitesse de l'application et minimiser son utilisation de mémoire.

Valgrind analyse votre application en l'exécutant sur un CPU synthétique et en instrumentant le code de l'application pendant son exécution. Puis il imprime « commmentary » (commentaire), identifiant ainsi clairement chaque processus impliqué dans l'exécution de l'application sur un descripteur de fichier, un fichier ou un socket de réseau spécifié par l'utilisateur. Le niveau d'instrumentation varie en fonction de l'outil Valgrind utilisé et de ses paramètres, mais il est important de remarquer que l'exécution du code instrumenté peut prendre de 4 à 50 fois plus longtemps qu'une exécution normale.

Valgrind peut être utilisé sur votre application tel quel, sans recompiler. Cependant, comme Valgrind utilise des informations de débogage pour identifier des problèmes dans votre code, si vos bibliothèques d'application et d'assistance n'ont pas été compilées avec les informations de débogage activées, il est recommandé de recompiler afin d'inclure ces informations.

À partir de Red Hat Enterprise Linux 6.4, Valgrind s'intègre avec gdb (« GNU Project Debugger ») pour améliorer l'efficacité du débogage.

Des informations supplémentaires sur Valgrind se trouvent dans le *Guide du développeur*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, ou en utilisant la commande **man valgrind** lorsque le paquetage valgrind est installé. Des documents accompagnateurs peuvent aussi être trouvés dans :

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

Pour obtenir des informations la manière d'utiliser Valgrind pour profiler la mémoire système, veuillez consulter la Section 5.3, « Utiliser Valgrind pour établir un profil de l'utilisation de mémoire ».

3.5.4. Perf

L'outil **perf** offre un certain nombre de compteurs de performance utiles permettant à l'utilisateur d'évaluer l'impact des autres commandes sur leur système :

perf stat

Cette commande fournit des statistiques générales sur les événements communs des performances, y compris les instructions exécutées et les cycles d'horloge consommés. Vous pouvez utiliser les indicateurs d'option pour rassembler des statistiques sur des événements différents des événements de mesure par défaut. À partir de Red Hat Enterprise Linux 6.4, il est possible d'utiliser **perf stat** pour filtrer le contrôle en se basant sur un ou plusieurs groupe(s) de contrôle spécifié(s) (cgroups). Pour obtenir des informations supplémentaires, veuillez lire la page man : **man perf-stat**.

perf record

Cette commande enregistre les données des performances dans un fichier qui peut être analysé ultérieurement à l'aide de **perf report**. Pour obtenir de détails supplémentaires, veuillez lire la page man : **man perf-record**.

perf report

Cette commande lit les données des performances à partir d'un fichier et analyse les données enregistrées. Pour obtenir des détails supplémentaires, veuillez lire la page man : **man perf-report**.

perf list

Cette commande répertorie les événements disponibles sur une machine en particulier. Ces événements vont varier en se basant sur le matériel de contrôle des performances et sur la configuration logicielle du système. Pour obtenir des informations supplémentaires, veuillez lire la page man : **man perf-list**.

perf top

Cette commande effectue une fonction similaire à l'outil **top**. Elle génère et affiche un profil de compteur des performances en temps réel. Pour obtenir des informations supplémentaires, veuillez lire la page man : **man perf-top**.

Des informations supplémentaires sur **perf** sont disponibles sur le *Guide du développeur* Red Hat Enterprise Linux, qui se trouve sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

3.6. RED HAT ENTERPRISE MRG

Le composant Realtime de Red Hat Enterprise MRG inclut **Tuna**, un outil qui permet aux utilisateurs d'ajuster les valeurs réglables de leurs systèmes et d'afficher les résultats de ces changements. Même si cet outil a été développé pour une utilisation avec le composant Realtime, il peut aussi être utilisé pour régler des systèmes Red Hat Enterprise Linux standards.

Avec Tuna, vous pouvez ajuster ou désactiver toute activité non-nécessaire du système, y compris :

- Les paramètres BIOS liés à la gestion de l'alimentation, à la détection d'erreurs et aux interruptions de gestion du système ;
- les paramètres réseau, comme la fusion d'interruptions et l'utilisation de TCP ;
- les activités de journalisation dans les systèmes de fichiers de journalisation ;
- les journalisations de système ;
- si les interruptions et les processus utilisateur sont gérés par un CPU ou une gamme de CPU spécifique ;
- si l'espace swap est utilisé ;
- et comment gérer les exceptions de mémoire saturée.

Pour obtenir des informations conceptuelles plus détaillées sur le réglage de Red Hat Enterprise MRG avec l'interface Tuna, veuillez consulter le chapitre « Réglages du système général » du *Guide de réglage de Realtime*. Pour obtenir des instructions détaillées sur l'utilisation de l'interface Tuna, reportez-

vous au *Guide d'utilisation Tuna*. Ces deux guides sont disponibles sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_MRG/.

CHAPITRE 4. CPU

Le terme CPU, de l'anglais *Central Processing Unit*, est un nom inapproprié pour la plupart des systèmes puisque le mot *central* sous-entend *unique*, alors que la plupart des systèmes modernes possèdent plus d'une unité de traitement, ou cœur. Physiquement, les CPU sont contenus dans un paquet attaché à une carte-mère dans un *socket*. Chaque socket sur la carte-mère possède plusieurs connexions : vers d'autres sockets de CPU, vers des contrôleurs mémoire, vers des contrôleurs d'interruptions, et autres périphériques. Un socket vers le système d'exploitation est un groupement logique de CPU et de ressources associées. Ce concept est central à la plupart des discussions sur l'optimisation des CPU.

Red Hat Enterprise Linux conserve une mine de statistiques sur les événements CPU de systèmes ; ces statistiques sont utiles à la planification d'une stratégie d'amélioration des performances des CPU. La Section 4.1.2, « Régler les performances CPU » discute de certaines des statistiques les plus utiles, où les trouver et comment les analyser pour optimiser les performances.

TOPOLOGIE

Les ordinateurs plus anciens possèdent relativement moins de CPU par système, ce qui permettait une architecture connue sous le nom de SMP (de l'anglais, *Symmetric Multi-Processor*, « multiprocesseur symétrique »). Cela signifie que chaque CPU dans le système possède un accès similaire (ou symétrique) à la mémoire disponible. Ces dernières années, le nombre de CPU par socket a augmenté au point où tenter de donner un accès symétrique à toute la mémoire RAM dans un système est devenu très coûteux. De nos jours, la plupart des systèmes comptant de nombreux CPU possèdent une architecture appelée NUMA (de l'anglais, *Non-Uniform Memory Access*, « accès mémoire non-uniforme ») au lieu de SMP.

Les processeurs AMD ont possédé ce type d'architecture depuis quelques temps avec leurs interconnexions HT (de l'anglais, *Hyper Transport*), tandis qu'Intel a commencé à implémenter NUMA dans leurs designs QPI (de l'anglais, *Quick Path Interconnect*). NUMA et SMP sont optimisés de différentes manières, puisque vous devez fournir un acompte de la *topologie* du système lors de l'allocation de ressources d'une application.

THREADS

À l'intérieur d'un système d'exploitation Linux, l'unité d'exécution est appelée un *thread*. Les threads possèdent un contexte de registre, une pile et un segment de code exécutable qu'ils exécutent sur un CPU. Le travail du système d'exploitation est de programmer ces threads sur les CPU disponibles.

Le système d'exploitation maximise l'utilisation du CPU en équilibrant les charges des threads sur les cœurs disponibles. Puisque le système d'exploitation est principalement intéressé à garder les CPU occupés, il peut ne pas prendre les décisions optimales quand aux performances des applications. Déplacer un thread d'application vers le CPU sur un autre socket peut être pire au niveau performance que simplement attendre que le CPU redevienne disponible, puisque les opérations d'accès mémoire peuvent ralentir de manière drastique sur les sockets. Pour des applications de haute performance, il est habituellement mieux que le créateur détermine où les threads devraient être placés. La Section 4.2, « Ordonnancement CPU » traite des meilleures manières d'allouer les CPU et la mémoire afin de mieux exécuter les threads d'application.

INTERRUPTIONS

L'un des événements système moins évident (mais tout aussi important) pouvant avoir un impact sur les performances des applications sont les *interruptions* (aussi appelées des IRQ sous Linux). Ces événements sont gérés par le système d'exploitation et sont utilisés par des périphériques pour signaler l'arrivée de données ou l'achèvement d'une opération, comme une opération d'écriture réseau ou un événement d'horodatage.

La manière par laquelle le système d'exploitation ou le CPU qui exécute le code de l'application gère une interruption n'affecte pas le fonctionnement de l'application. Cependant, il peut y avoir un impact sur

la performance de l'application. Ce chapitre discute fournit aussi des conseils sur comment empêcher les interruptions d'avoir un impact négatif sur les performances des applications.

4.1. TOPOLOGIE DE CPU

4.1.1. Topologie des CPU et de NUMA

Les premiers processeurs d'ordinateurs étaient des *monoprocesseurs*, ce qui signifie que le système ne possédait qu'un seul CPU. L'illusion d'exécution des processus en parallèle était due au fait que le système d'exploitation faisait rapidement basculer l'unique CPU d'un thread d'exécution (ou processus) à un autre. En vue d'améliorer les performances du système, les concepteurs ont remarqué qu'augmenter la vitesse de l'horloge pour exécuter des instructions plus rapidement fonctionnait jusqu'à un certain point uniquement (correspondant habituellement aux limitations de la création d'une forme d'onde d'horloge stable avec la technologie actuelle). Dans un effort pour accomplir de meilleures performances générales du système, les concepteurs ont ajouté un autre CPU au système, permettant ainsi deux flux d'exécution parallèles. Cette tendance d'ajout de processeurs a continué au cours du temps.

La plupart des premiers systèmes à multiples processeurs étaient conçus de manière à ce que chaque CPU possédait le même chemin logique vers chaque emplacement mémoire (habituellement un bus parallèle). Ceci laisse autant de temps à chaque CPU dans le système pour accéder à tout emplacement dans la mémoire. Ce type d'architecture est connu sous le nom de système SMP (« Symmetric Multi-Processor »). SMP est convenable pour un petit nombre de CPU, mais lorsque le compte de CPU dépasse un certain seuil (8 ou 16), le nombre de traces parallèles requises pour permettre un accès égal à la mémoire utilise trop d'emplacements disponibles sur la carte, laissant moins d'espace aux périphériques.

Deux nouveaux concepts combinés pour autoriser un nombre de CPU élevé dans un système :

1. Bus sériels

2. Topologies NUMA

Un bus sériel est un chemin de communication unifilaire avec une très haute fréquence d'horloge, qui transfère les données en rafales de paquets. Les concepteurs de matériel ont commencé à utiliser les bus sériels en tant qu'interconnexions à haute vitesse entre les CPU, les contrôleurs mémoire et les autres périphériques. Cela signifie qu'au lieu de nécessiter entre 32 et 64 traces depuis la carte de *chaque* CPU vers le sous-système de la mémoire, il n'y a plus qu'*une* trace, réduisant la quantité d'espace requis sur la carte de manière substantielle.

Les concepteurs de matériel empaquetaient davantage de transistors dans le même espace en réduisant les tailles des dies. Au lieu de mettre les CPU individuels directement sur la carte-mère, ils se sont mis à les empaqueter dans des paquets (ou cartouches) de processeurs en tant que processeurs multi-cœurs. Puis, au lieu d'essayer de fournir un accès égal à la mémoire pour chaque paquet de processeurs, les concepteurs ont eu recours à la stratégie NUMA (accès mémoire non-uniforme, de l'anglais « Non-Uniform Memeroy Access »), avec laquelle chaque combinaison paquet/socket possède une ou plusieurs zone(s) de mémoire dédiée(s) pour un accès à grande vitesse. Chaque socket possède aussi une interconnexion vers d'autres sockets pour un accès plus lent vers la mémoire des autres sockets.

Prenez en considération ce simple exemple de NUMA ; supposons que nous disposons d'une carte-mère à deux sockets et que chaque socket contient un paquet quad-core. Cela signifie que le nombre total de CPU dans le système est de huit, quatre dans chaque socket. Chaque socket possède aussi une banque mémoire attachée avec quatre gigaoctets de mémoire vive (RAM), pour un total de huit gigaoctets de mémoire système. Dans cet exemple, les CPU 0 à 3 sont dans le socket 0 et les CPU 4 à 7 sont dans le socket 1. Chaque socket de cet exemple correspond aussi à un nœud NUMA.

L'accès mémoire de la banque 0 du CPU 0 peut prendre trois cycles d'horloge : un cycle pour présenter l'adresse au contrôleur mémoire, un cycle pour paramétrer l'accès à l'emplacement mémoire et un cycle pour lire ou écrire sur l'emplacement. Cependant, l'accès à la mémoire depuis le même emplacement par le CPU 4 peut prendre six cycles ; comme il se trouve sur un autre socket, il doit passer par deux contrôleurs mémoire : le contrôleur mémoire local sur le socket 1, puis le contrôleur mémoire distant sur le socket 0. Si la mémoire est contestée sur cet emplacement (c'est-à-dire si plus d'un CPU tente d'accéder au même emplacement simultanément), les contrôleurs mémoire devront arbitrer et sérialiser l'accès à la mémoire, l'accès mémoire prendra ainsi plus de temps. L'ajout de consistance de cache (s'assurer que les caches du CPU local contiennent bien les mêmes données pour le même emplacement mémoire) complique encore plus le processus.

Les plus récents processeurs hauts de gamme d'Intel (Xeon) et d'AMD (Opteron) possède des topologies NUMA. Les processeurs AMD utilisent une interconnexion appelée HyperTransport, ou HT, alors qu'Intel en utilise une nommée QuickPath Interconnect, ou QPI. Les interconnexions diffèrent dans la manière par laquelle elles se connectent aux autres interconnexions, à la mémoire, ou aux appareils périphériques, mais en réalité, elles sont un commutateur permettant un accès transparent vers un périphérique connecté depuis un autre périphérique connecté. Dans ce cas, transparent fait référence au fait qu'aucune API de programmation particulière n'est requise pour utiliser l'interconnexion.

Comme les architectures de systèmes sont très diverses, il n'est pas commode de caractériser de manière spécifique les pénalités de performance imposées par l'accès à la mémoire non-locale. Il peut être dit que chaque *saut* à travers une interconnexion impose au moins une pénalité de performance relativement constante. Ainsi, référencer un emplacement mémoire se trouvant à deux interconnexions du CPU actuel imposera au moins *2N + temps de cycle mémoire*, où N est la pénalité par saut.

Au vu de cette pénalité de performance, les applications sensibles aux performances devraient éviter d'accéder à la mémoire distante de manière régulière dans un système avec une topologie NUMA. L'application devrait être paramétrée de manière à rester sur un nœud en particulier et alloue de la mémoire à partir de ce nœud.

Pour ce faire, les applications doivent connaître quelques choses :

1. Quelle est la *topologie* du système ?
2. Où se trouve l'application actuellement en cours d'exécution ?
3. Où se trouve la banque mémoire la plus proche ?

4.1.2. Régler les performances CPU

Veuillez lire cette section pour comprendre comment effectuer les réglages afin d'améliorer les performances CPU et pour une introduction à plusieurs outils pouvant assister dans ce processus.

À l'origine, NUMA était utilisé pour connecter un seul processeur à de multiples banques de mémoire. Comme les fabricants de CPU raffinaient leurs processus et que les tailles de die diminuaient, de multiples cœurs de CPU ont pu être inclus dans un seul paquetage. Ces cœurs de CPU ont été mis en grappe afin qu'ils aient tous un accès équitable à la banque de mémoire locale et que le cache puisse être partagé entre les cœurs ; cependant, chaque « saut » à travers une interconnexion entre le cœur, la mémoire et le cache implique une petite pénalité de performance.

L'exemple de système Figure 4.1, « Accès local et distant dans la topologie NUMA » contient deux nœuds NUMA. Chaque nœud possède quatre CPU, une banque de mémoire et un contrôleur de mémoire. Tout CPU sur un nœud a un accès direct à la banque de mémoire de ce nœud. En suivant les flèches sur le nœud 1 (« Node 1 »), les étapes sont comme suit :

1. Un CPU (de 0 à 3) présente l'adresse mémoire au contrôleur de mémoire local.

2. Le contrôleur de mémoire paramètre l'accès à l'adresse mémoire.

3. Le CPU effectue des opérations de lecture ou d'écriture sur cette adresse mémoire.

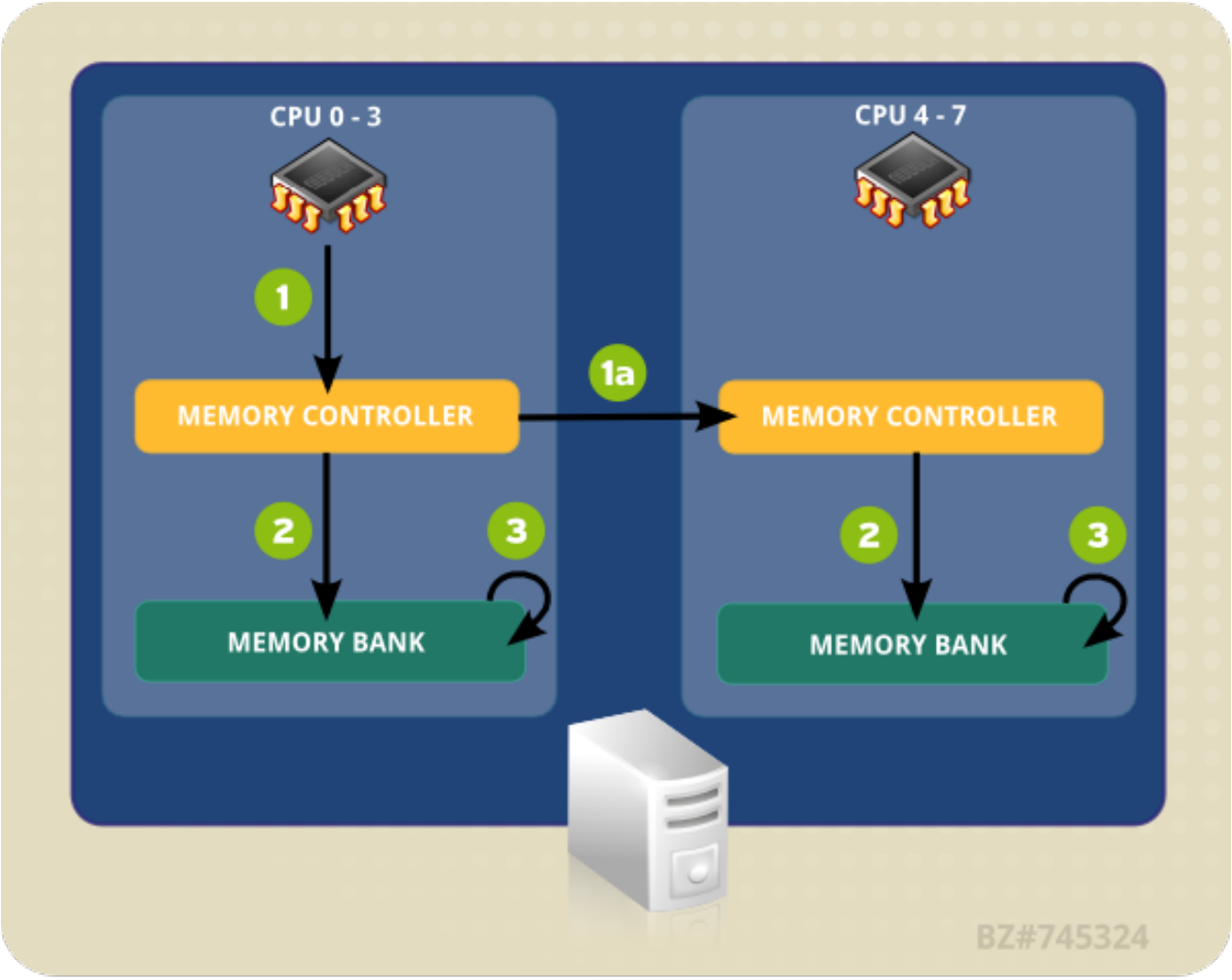


Figure 4.1. Accès local et distant dans la topologie NUMA

Cependant, si un CPU sur un nœud doit accéder au code résidant sur la banque mémoire d'un autre nœud NUMA, le chemin qu'il doit prendre sera moins direct :

1. Un CPU (de 0 à 3) présente l'adresse mémoire distante au contrôleur de mémoire local.

    1. La requête du CPU pour cette adresse mémoire distante est passée à un contrôleur de mémoire distant, local au nœud contenant cette adresse mémoire.

2. Le contrôleur de mémoire distant paramètre l'accès à l'adresse mémoire distante.

3. Le CPU effectue des opérations de lecture ou d'écriture sur cette adresse mémoire distante.

Chaque action doit passer à travers de multiples contrôleurs de mémoire. Ainsi, l'accès peut prendre plus de deux fois plus longtemps lors de tentatives d'accès aux adresses mémoire distantes. Ainsi, la préoccupation principale en termes de performances dans un système multi-core est de s'assurer que les informations puissent circuler le plus efficacement possible, via la chemin le plus court ou le plus rapide.

Pour configurer une application pour des performances CPU optimales, vous devez connaître :

- la topologie du système (de quelle manière sont connectés ses composants),
- le cœur sur lequel l'application s'exécute et
- l'emplacement de la banque de mémoire la plus proche.

Red Hat Enterprise Linux 6 est fourni avec un certain nombre d'outils pour vous aider à trouver ces informations et régler votre système en conséquence. Les sections suivantes vous offrent un aperçu des outils utiles aux réglages des performances CPU.

4.1.2.1. Définir les affinités de CPU avec taskset

taskset récupère et définit les affinités de CPU d'un processus en cours d'exécution (par ID de processus). Il peut aussi être utilisé pour lancer un processus avec une affinité de CPU donnée, ce qui lie le processus spécifié à un CPU ou à un ensemble de CPU spécifié. Cependant, **taskset** ne garantit pas l'allocation de mémoire locale. Si vous avez besoin des bénéfices de performance supplémentaires offerts par l'allocation de mémoire locale, nous recommandons **numactl** plutôt que **taskset** ; veuillez consulter la Section 4.1.2.2, « Contrôler la stratégie NUMA avec **numactl** » pour obtenir de détails supplémentaires.

Les affinités de CPU sont représentées en tant que masque de bits. Le bit d'ordre le plus bas correspond au premier CPU logique et le plus haut correspond au dernier CPU logique. ces masques sont habituellement donnés en valeurs hexadécimales, ainsi **0x00000001** représente le processeur 0 et **0x00000003** représente les processeurs 0 et 1.

Pour définir les affinités de CPU d'un processus en cours d'exécution, veuillez exécuter la commande suivante, en remplaçant *mask* par le masque du ou des processeurs auxquels vous souhaitez voir le processus lié et remplacez *pid* par l'ID de processus du processus pour lequel vous souhaitez modifier les affinités.

```
# taskset -p mask pid
```

Pour lancer un processus avec une affinité en particulier, exécutez la commande suivante en remplaçant *mask* par le masque du ou des processeurs auxquels vous souhaitez voir le processus lié et *program* par le programme, les options et les arguments du programme que vous souhaitez exécuter.

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

Au lieu de spécifier les processeurs en tant que masque de bits, vous pouvez aussi utiliser l'option **-c** pour fournir une liste séparée par des virgules de différents processeurs, ou une gamme de processeurs, de la manière suivante :

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

Des informations supplémentaires sur **taskset** sont disponibles sur la page man : **man taskset**.

4.1.2.2. Contrôler la stratégie NUMA avec numactl

numactl exécute les processus avec une stratégie d'ordonnancement ou de placement de mémoire spécifiée. La stratégie sélectionnée est définie pour ce processus et tous ses enfants. **numactl** peut aussi définir une stratégie persistante pour des segments de mémoire ou des fichiers partagés et les affinités CPU de la mémoire d'un processus. Il utilise le système de fichiers **/sys** pour déterminer la topologie du système.

Le système de fichiers **/sys** contient des informations sur la manière par laquelle les CPU, la mémoire et les appareils périphériques sont connectés via des interconnexions NUMA. Plus spécifiquement, le répertoire **/sys/devices/system/cpu** contient des informations sur la manière par laquelle les CPU d'un système sont connectés les uns aux autres. Le répertoire **/sys/devices/system/node** contient des informations sur les nœuds NUMA dans le système et les distances relatives entre ces nœuds.

Dans un système NUMA, plus la distance est grande entre un processeur et banque de mémoire, plus l'accès de ce processeur à cette banque de mémoire sera lent. Ainsi, les applications dépendantes des performances devraient donc être configurées de manière à allouer de la mémoire depuis la banque de mémoire la plus proche possible.

Les applications dépendantes des performances doivent aussi être configurées de manière à s'exécuter sur un nombre de cœurs défini, particulièrement dans le cas d'applications à multiples threads. Comme les caches de premier niveau sont habituellement petits, si de multiples threads sont exécutés sur un seul cœur, chaque thread pourrait potentiellement supprimer les données mises en cache accédées par thread précédent. Lorsque le système d'exploitation tente d'effectuer plusieurs tâches à la fois (« multitask ») sur ces threads et que les threads continuent de supprimer les données mises en cache des uns et des autres, un grand pourcentage de leur temps d'exécution est passé à remplacer les lignes du cache. Ce problème est appelé le *trashing de cache*. Il est ainsi recommandé de lier une application à multiples threads à un nœud plutôt qu'à un seul cœur, puisque cela permet aux threads de partager des lignes de cache sur de multiples niveaux (premier, second et dernier niveau de cache) et minimise le besoin d'opérations de remplissage des caches. Cependant, lier une application à un seul cœur peut être performant si tous les threads accèdent aux mêmes données mises en cache.

numactl vous permet de lier une application à un cœur ou à un nœud NUMA en particulier, ainsi que d'allouer la mémoire associée à un cœur ou à un ensemble de cœurs à cette application. Voici quelques options utiles offertes par numactl :

--show

Affiche les paramètres de la stratégie NUMA du processus actuel. Ce paramètre ne requiert pas de paramètres supplémentaires et peut être utilisé comme ceci : **numactl --show**.

--hardware

Affiche un inventaire des nœuds disponibles sur le système.

--membind

Alloue la mémoire des nœuds spécifiés uniquement. Lorsque ceci est en cours d'utilisation, l'allocation échouera si la mémoire sur ces nœuds est insuffisante. L'utilisation de ce paramètre se fait ainsi : **numactl --membind=*nodes program***, où *nodes* est la liste des nœuds à partir desquels vous souhaitez allouer de la mémoire et *program* est le programme dont les conditions nécessaires de mémoire devraient être allouées à partir de ce nœud. Le nombre de nœuds peut être donné en tant que liste séparée par des virgules, en tant que gamme, ou avec une combinaison des deux. Des détails supplémentaires sont disponibles sur la page man **numactl** : **man numactl**.

--cpunodebind

Exécute une commande (et ses processus enfants) sur les CPU appartenant au(x) nœud(s) spécifié(s) uniquement. L'utilisation de ce paramètre se fait ainsi : **numactl --cpunodebind=*nodes program***, où *nodes* est la liste des nœuds dont les CPU devraient être liés au programme spécifié (*program*). Les numéros des nœuds peuvent être donnés en tant que liste séparée par des virgules, en tant que gamme, ou avec une combinaison des deux. Des détails supplémentaires sont disponibles sur la page man **numactl** : **man numactl**.

--physcpubind

Exécute une commande (et ses processus enfants) sur les CPU spécifiés uniquement. L'utilisation de ce paramètre se fait ainsi : **numactl --physcpubind=*cpu program***, où *cpu* est la liste séparée par des virgules des numéros des CPU physiques comme affichés dans les champs du processeur de **/proc/cpuinfo** et *program* est le programme qui doit uniquement effectuer des exécutions sur ces CPU. Les CPU peuvent aussi être spécifiés de manière relative au **cpuset** actuel. Veuillez consulter la page man **numactl** pour obtenir des informations supplémentaires : **man numactl**.

--localalloc

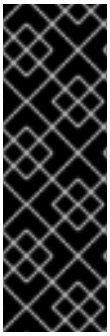
Spécifie que la mémoire devrait toujours être allouée sur le nœud actuel.

--preferred

Lorsque possible, la mémoire est allouée sur le nœud spécifié. Si la mémoire ne peut pas être allouée sur le nœud spécifié, elle le sera sur d'autres nœuds. Cette option prend uniquement un seul numéro de nœud, comme suit : **numactl --preferred=*node***. Veuillez consulter la page man numactl pour obtenir des informations supplémentaires : **man numactl**.

La bibliothèque **libnuma** incluse dans le paquetage numactl offre une simple interface de programmation à la stratégie NUMA prise en charge par le noyau. Elle est utile pour effectuer des réglages à granularité plus fine que l'utilitaire **numactl**. Des informations supplémentaires sont disponibles sur la page man : **man numa(3)**.

4.1.3. numastat



IMPORTANT

Auparavant, l'outil **numastat** était un script Perl écrit par Andi Kleen. Il a été réécrit de manière significative pour Red Hat Enterprise Linux 6.4.

Même si la commande par défaut (**numastat**, sans options ni paramètres) maintient une stricte compatibilité avec la version précédente de l'outil, remarquez que fournir des options ou des paramètres à cette commande modifiera le contenu et le format de sa sortie de manière significative.

numastat affiche des statistiques de mémoire (comme les allocations réussies et manquées) pour les processus et le système d'exploitation sur une base « par nœud NUMA ». Par défaut, exécuter **numastat** affiche combien de pages de mémoire sont occupées par les catégories d'événements suivantes sur chaque nœud.

Les performances optimales du CPU sont indiquées par des valeurs **numa_miss** et **numa_foreign** basses.

Cette version mise à jour de **numastat** affiche aussi si la mémoire du processus est étendue à travers un système ou centralisée sur des nœuds spécifiques à l'aide de **numactl**.

Recoupez la sortie **numastat** avec la sortie **top** par CPU pour vérifier que les threads de processus sont exécutés sur les mêmes nœuds que ceux auxquels la mémoire est allouée.

Catégories de suivi par défaut

numa_hit

Nombre d'allocations tentées sur ce nœud ayant réussi.

numa_miss

Nombre de tentatives d'allocations sur un autre nœud qui ont été allouées à ce nœud en raison de la faible mémoire sur le nœud souhaité. Chaque événement **numa_miss** possède un événements **numa_foreign** correspondant sur un autre nœud.

numa_foreign

Nombre d'allocations initialement souhaitées pour ce nœud qui ont été allouées à un autre nœud. Chaque événement **numa_foreign** possède un événement **numa_miss** correspondant sur un autre nœud.

interleave_hit

Nombre d'allocations de stratégie d'entrelacement réussies sur ce nœud.

local_node

Nombre de fois qu'un processus sur ce nœud a effectivement réussi à allouer de la mémoire sur ce nœud.

other_node

Nombre de fois qu'un processus sur un autre nœud a alloué de la mémoire sur ce nœud.

Fournir l'une des options suivantes change les unités affichées en mégaoctets de mémoire, arrondis à deux décimales) et modifie d'autres comportements **numastat** spécifiques, comme décrit ci-dessous.

-c

Condense horizontalement la table d'informations affichée. Ceci est utile sur des systèmes comportant un grand nombre de nœuds NUMA, mais la largeur des colonnes et l'espacement entre colonnes reste quelque peu imprévisible. Lorsque cette option est utilisée, la quantité de mémoire est arrondie au mégaoctet le plus proche.

-m

Affiche les informations d'utilisation de la mémoire du système global sur une base « par nœud », de manière similaire aux informations trouvées dans **/proc/meminfo**.

-n

Affiche les mêmes informationa que la commande d'origine **numastat** (numa_hit, numa_miss, numa_foreign, interleave_hit, local_node et other_node), avec un format mis à jour utilisant les mégaoctets comme unité de mesure.

-p *pattern*

Affiche des informations de mémoire par nœud pour le schéma spécifié. Si la valeur de *pattern* comprend des chiffres, **numastat** suppose qu'il s'agit un identifiant de processus numérique. Sinon, **numastat** recherche le schéma spécifié dans les lignes de commande des processus.

Les arguments de ligne de commande saisis après la valeur de l'option **-p** sont supposées être des schémas supplémentaires avec lesquels filtrer. Les schémas supplémentaires agrandissent le filtre, plutôt que le réduire.

-s

Arrange les données en ordre descendant de manière à ce que les plus gros consommateurs de mémoire (selon la colonne **total**) soient répertoriés en premier.

Optionnellement, vous pouvez spécifier un *nœud* et le tableau sera ordonné selon la colonne *nœuds*. Lors de l'utilisation de cette option, la valeur *nœud* doit suivre l'option **-s** immédiatement, comme décrit ici :

```
numastat -s2
```

N'incluez pas d'espace blanc entre l'option et sa valeur.

-v

Affiche des informations plus détaillées (« verbose information »). C'est-à-dire des informations de processus sur de multiples processus afficheront des informations détaillées sur chaque processus.

-V

Affiche les informations de version de **numastat**.

-z

Omet les lignes et colonnes du tableau ne comportant que la valeur zéro du reste des informations affichées. Remarquez que les valeurs proches de zéro qui sont arrondies à zéro pour des raisons d'affichage ne seront pas omises de la sortie affichée.

4.1.4. numad (« NUMA Affinity Management Daemon »)

numad est un démon de gestion des affinités NUMA automatique. Il contrôle la topologie NUMA et l'utilisation des ressources dans un système afin de dynamiquement améliorer l'allocation et la gestion des ressources NUMA (et donc les performances système).

Selon la charge de travail du système, **numad** peut fournir une amélioration des performances de référence allant jusqu'à 50%. pour réaliser ces gains de performance, **numad** accède périodiquement aux informations du système de fichiers **/proc** pour contrôler les ressources système disponibles sur une base « par nœud ». Le démon tente ensuite de placer des processus significatifs sur les nœuds NUMA possédant suffisamment de mémoire alignée et de ressources CPU pour des performances NUMA optimales. Les limites actuelles pour la gestion des processus sont d'au moins 50% d'un CPU et au moins 300 Mo de mémoire.**numad** tente de maintenir un niveau d'utilisation des ressources et ré-équilibre les allocations lorsque nécessaire en déplaçant les processus entre les nœuds NUMA.

numad offre aussi un service de conseils pré-placement pouvant être interrogé par divers systèmes de gestion de tâches pour fournir une certaine assistance avec la liaison initiale des ressources CPU et de mémoire pour leurs processus. Ce service de conseils pré-placement est disponible que **numad** soit exécuté en tant que démon sur le système ou non. Veuillez vous reporter à la page man pour obtenir des détails supplémentaires sur l'utilisation de l'option **-w** pour les conseils pré-placement : **man numad**.

4.1.4.1. Bénéfices de numad

numad bénéficie principalement aux systèmes possédant des processus de longue durée qui consomment des quantités de ressources significatives, particulièrement lorsque ces processus sont contenus dans un sous-ensemble de la totalité des ressources système.

numad peut aussi être bénéfique à des applications consommant une quantité de ressources correspondant à de multiples nœuds NUMA. Cependant, plus le pourcentage de ressources consommées sur un système augmente, plus les bénéfices offerts par **numad** diminuent.

Il est improbable que **numad** puisse améliorer les performances lorsque les processus sont exécutés depuis quelques minutes uniquement ou lorsqu'ils ne consomment que peu de ressources. Il est aussi improbable que des systèmes avec des schémas d'accès mémoire continuellement imprévisibles, comme les bases de données intégrées de grande taille, puissent bénéficier de l'utilisation de **numad**.

4.1.4.2. Modes d'opération

NOTE

Les statistiques de comptabilité de la mémoire du noyau peuvent se contredire après des fusions de grande taille. Ainsi, **numad** peut être confondu lorsque le démon KSM fusionne de grandes quantités de mémoire. Le démon KSM sera davantage conscient de NUMA dans le futur. Cependant, si votre système possède actuellement une grande quantité de mémoire libre, vous pourriez obtenir de meilleures performances en éteignant et désactivant le démon KSM.

numad peut être utilisé de deux manières :

- en tant que service
- en tant qu'exécutable

4.1.4.2.1. Utiliser numad en tant que service

Pendant que le service **numad** est en cours d'exécution, il tentera de régler le système dynamiquement en se basant sur sa charge de travail.

Pour lancer le service, veuillez exécuter :

```
# service numad start
```

Pour rendre le service persistant à travers les redémarrages, veuillez exécuter :

```
# chkconfig numad on
```

4.1.4.2.2. Utiliser numad en tant qu'exécutable

Pour utiliser **numad** en tant qu'exécutable, veuillez exécuter :

```
# numad
```

numad sera exécuté jusqu'à ce qu'il soit arrêté. Lorsqu'il est exécuté, ses activités sont journalisées dans **/var/log/numad.log**.

Pour restreindre la gestion **numad** à un processus spécifique, veuillez le démarrer avec les options suivantes.

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

-p pid

Ajoute le *pid* spécifié sur une liste d'inclusions spécifiques. Le processus spécifié ne sera pas géré jusqu'à ce qu'il atteigne la limite d'importance de processus **numad**.

-S mode

Le paramètre ***-S*** spécifie le type de scan de processus. Le définir sur **0** comme suit limite la gestion **numad** aux processus explicitement inclus.

Pour arrêter **numad**, veuillez exécuter :

```
# numad -i 0
```

Arrêter **numad** ne supprime pas les changements effectués pour améliorer les affinités NUMA. Si l'utilisation du système change de manière significative, exécuter **numad** à nouveau ajustera les affinités pour améliorer les performances sous les nouvelles conditions.

Pour obtenir des informations supplémentaires sur les options **numad** disponibles, veuillez consulter la page man **numad** : **man numad**.

4.2. ORDONNANCEMENT CPU

L'*ordonnanceur* est responsable de garder les CPU dans le système occupés. L'ordonnanceur Linux implémente un certain nombre de *stratégies d'ordonnancement* qui déterminent quand et pour combien de temps un thread est exécuté sur un cœur de CPU en particulier.

Les stratégies d'ordonnancement sont divisées en deux catégories majeures :

1. Les stratégies en temps réel
    - SCHED_FIFO
    - SCHED_RR
2. Les stratégies normales
    - SCHED_OTHER
    - SCHED_BATCH
    - SCHED_IDLE

4.2.1. Stratégies d'ordonnancement en temps réel

Les threads en temps réel sont ordonnancés en premier et les threads normaux sont ordonnancés une fois que tous les threads en temps réel ont été ordonnancés.

Les stratégies en *temps réel* sont utilisées pour des tâches pour lesquelles le temps est critique, des tâches devant être effectuées sans interruption.

SCHED_FIFO

Cette stratégie est aussi appelée *ordonnancement à priorité statique* car elle définit une priorité fixe (de 1 à 99) pour chaque thread. L'ordonnanceur scanne une liste de threads SCHED_FIFO par ordre de priorité et programme le thread prêt avec la priorité plus élevée pour exécution. Ce thread est

exécuté jusqu'à ce qu'il se bloque, qu'il échoue, ou qu'il soit préempté par un thread de plus haute priorité prêt à être exécuté.

Même le thread en temps réel avec la plus basse priorité sera programmé avant tout thread possédant une stratégie qui n'est pas en temps réel, s'il n'existe qu'un seul thread en temps réel, la valeur de la priorité **SCHED_FIFO** n'importe pas.

SCHED_RR

Variante round-robin (tourniquet) de la stratégie **SCHED_FIFO**. Les threads **SCHED_RR** se voient aussi donner une priorité fixe entre 1 et 99. Cependant, les threads avec la même priorité sont programmés de manière round-robin (ou comme un tourniquet) avec un certain quantum, ou tranche, de temps. L'appel système **sched_rr_get_interval(2)** retourne la valeur de la tranche de temps, mais la durée de la tranche de temps ne peut pas être définie par un utilisateur. Cette stratégie est utile si vous avez besoin que de multiples threads soient exécutés avec la même priorité.

Pour obtenir des informations plus détaillées sur les sémantiques définies des stratégies d'ordonnancement en temps réel, veuillez consulter le *standard POSIX IEEE 1003.1* disponible sous « System Interfaces — Realtime », qui se trouve sur http://pubs.opengroup.org/onlinepubs/009695399/functions/xsh_chap02_08.html.

La meilleure manière de définir la priorité des threads est de commencer par les plus faibles priorités et d'augmenter uniquement lorsqu'une latence légitime est identifiée. Les threads en temps réel ne possèdent pas de tranches de temps comme les threads normaux. Les threads **SCHED_FIFO** sont exécutés jusqu'à ce qu'il soient bloqués, qu'ils échouent, ou qu'ils soient préemptés par un thread possédant une priorité plus élevée. Ainsi, définir une priorité de 99 n'est pas recommandé, cela placerait votre processus au même niveau de priorité que des threads de migration et de surveillance. Si ces threads sont bloqués parce que votre thread se retrouve dans une boucle de calcul, ils ne pourront pas être exécutés. Les systèmes à monoprocesseur se retrouveront éventuellement dans une telle situation.

Dans le noyau Linux, la stratégie **SCHED_FIFO** inclut un mécanisme de limite de bande passante. Cela protège les programmeurs d'applications en temps réel des tâches en temps réel qui pourraient monopoliser le CPU. Ce mécanisme peut être ajusté via les paramètres du système de fichiers **/proc** suivants :

/proc/sys/kernel/sched_rt_period_us

Définit la période de temps devant être considérée comme cent pour cent de la bande passante du CPU, en microsecondes (« us » est le plus proche équivalent de « µs » en texte brut). La valeur par défaut est 1000000µs, ou 1 seconde.

/proc/sys/kernel/sched_rt_runtime_us

Définit la période de temps devant être dévouée à l'exécution de threads en temps réel, en microsecondes (« us » est le plus proche équivalent de « µs » en texte brut). La valeur par défaut est 950000µs, ou 0.95 secondes.

4.2.2. Stratégies d'ordonnancement normal

Il existe trois stratégies d'ordonnancement normal : **SCHED_OTHER**, **SCHED_BATCH** et **SCHED_IDLE**. Cependant, les stratégies **SCHED_BATCH** et **SCHED_IDLE**, conçues pour des tâches à très basse priorité, ne présentent qu'un intérêt limité dans un guide de réglage des performances.

SCHED_OTHER, ou SCHED_NORMAL

Stratégie d'ordonnancement par défaut. Cette stratégie utilise l'ordonnanceur CFS (de l'anglais « Completely Fair Scheduler », ordonnanceur complètement équitable) afin de fournir des périodes d'accès équitables à tous les threads utilisant cette stratégie. CFS établit une liste de priorités dynamiques, en partie basée sur la valeur ***niceness*** de chaque thread de processus. (Reportez-vous au *Guide de déploiement* pour obtenir des détails sur ce paramètre et sur le système de fichiers **/proc**.) Cela offre aux utilisateurs un certain niveau de contrôle indirect sur les priorités des processus, mais la liste des priorités dynamiques peut uniquement être directement modifiée par l'ordonnanceur CFS.

4.2.3. Sélection de la stratégie

Sélectionner la bonne stratégie d'ordonnanceur pour les threads d'une application n'est pas toujours une tâche simple. En général, des stratégies en temps réel devraient être utilisées pour les tâches pour lesquelles le temps est critique ou les tâches importantes devant être ordonnancées rapidement et ne pas être exécutées pendant de longues périodes. Les stratégies normales obtiennent habituellement de meilleurs résultats de débit de données que les stratégies en temps réel car elles permettent à l'ordonnanceur d'exécuter les threads de manière plus efficace (C'est-à-dire qu'ils n'ont plus besoin d'être aussi souvent à nouveau ordonnancés pour préemption).

Si vous gérez de grands nombres de threads et que vous êtes principalement préoccupé par le débit des données (paquets réseau par seconde, écritures sur disque, etc...), alors veuillez utiliser **SCHED_OTHER** et laissez le système gérer l'utilisation du CPU à votre place.

Si vous êtes plus préoccupé par le temps de réponse d'un événement (la latence), alors utilisez **SCHED_FIFO**. Si vous possédez un petit nombre de threads, considérez la possibilité d'isoler un socket du CPU et de déplacer vos threads sur les cœurs de ce socket afin qu'aucun autre thread ne se mette en compétition pour du temps sur les cœurs.

4.3. RÉGLAGES DES INTERRUPTIONS ET DES IRQ

Une requête d'interruption (IRQ, de l'anglais « Interrupt Request ») est une requête de service envoyée au niveau du matériel. Les interruptions peuvent être envoyées par ligne de matériel dédiée ou à travers un bus de matériel en tant que paquet d'informations (MSI, de l'anglais « Message Signaled Interrupt »).

Lorsque les interruptions sont activées, la réception d'une IRQ invite un interrupteur à interrompre le contexte. Le code d'envoi des interruptions du noyau récupère le numéro de l'IRQ et sa liste associée d'ISR (de l'anglais, « Interrupt Service Routines ») enregistrés, puis appelle chaque ISR à son tour. L'ISR reconnaît l'interruption et ignore les interruptions redondantes de la même IRQ, puis met en file d'attente un gestionnaire différé pour terminer le traitement de l'interruption et faire en sorte que l'ISR n'ignore pas les futures interruptions.

Le fichier **/proc/interrupts** répertorie le nombre d'interruptions par CPU par périphérique d'E/S. Il affiche le numéro de l'IRQ, le numéro de cette interruption géré par chaque cœur de CPU, le type d'interruption et une liste séparée par des virgules des pilotes enregistrés pour recevoir cette interruption. (Pour obtenir des détailes supplémentaires, reportez-vous à la page man proc(5) : **man 5 proc**)

Les IRQ ont une propriété « d'affinité » associée, ***smp_affinity***, qui définit les cœurs des CPU autorisés à exécuter l'ISR pour cette IRQ. Cette propriété peut être utilisée pour améliorer les performances de l'application en assignant l'affinité de l'interruption et celle du thread de l'application vers un ou plusieurs cœurs de CPU spécifiques. Ceci permet le partage de ligne de cache entre l'interruption spécifiée et les threads de l'application.

La valeur d'affinité de l'interruption avec un numéro d'IRQ en particulier est stockée dans le fichier

/proc/irq/*NUMÉRO_IRQ*/smp_affinity, qui peut être affiché et modifié par l'utilisateur root. La valeur stockée dans ce fichier est un masque de bits représentant tous les cœurs des CPU dans le système.

Par exemple, pour définir l'affinité de l'interruption du pilote Ethernet sur un serveur avec quatre cœurs de CPU, commencez par déterminer le numéro de l'IRQ associé à ce pilote Ethernet :

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

Utilisez le numéro de l'IRQ pour localiser le fichier ***smp_affinity*** approprié :

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

La valeur par défaut de smp_affinity est **f**, ce qui signifie que l'IRQ peut être servie sur n'importe quel CPU dans le système. Définir cette valeur sur **1** comme suit signifie que seul CPU 0 peut servir cette interruption :

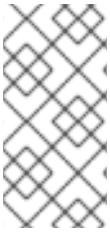
```
# echo 1 >/proc/irq/32/smp_affinity
# cat /proc/irq/32/smp_affinity
1
```

Des virgules peuvent être utilisées pour délimiter les valeurs ***smp_affinity*** pour des groupes discrets de 32 bits. Ceci est requis sur les systèmes avec polus de 32 cœurs. Par exemple, l'exemple suivant montre que l'IRQ 40 est servie sur tous les cœurs d'un système de 64 cœurs :

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

Pour servir IRQ 40 uniquement sur les 32 cœurs supérieurs d'un système de 64 cœurs, vous devrez effectuer :

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



NOTE

Sur les systèmes qui prennent en charge le *pilotage d'interruptions*, la modification du ***smp_affinity*** d'une IRQ paramètre le matériel de manière à ce que la décision de servir une interruption avec un CPU en particulier soit prise au niveau du matériel, sans intervention de la part du noyau.

4.4. AMÉLIORATIONS DE NUMA RED HAT ENTERPRISE LINUX 6

Red Hat Enterprise Linux 6 inclut un certain nombre d'améliorations afin de capitaliser sur la totalité du potentiel du matériel hautement évolutif moderne. Cette section vous offre un aperçu de haut niveau des plus importantes améliorations des performances NUMA apportées par Red Hat Enterprise Linux 6.

4.4.1. Optimisation de l'évolutivité et du bare-metal

4.4.1.1. Améliorations de la conscience de la topologie

Les améliorations suivantes permettent à Red Hat Enterprise Linux de détecter des détails du matériel de bas niveau et de l'architecture, améliorant ainsi sa capacité à automatiquement optimiser le traitement sur votre système.

détection de topologie améliorée

Ceci permet au système d'exploitation de détecter les détails du matériel de bas niveau (comme les CPU logiques, les hyper threads, les cœurs, sockets, nœuds NUMA et le temps d'accès entre les nœuds) au démarrage et d'optimiser le traitement sur votre système.

completely fair scheduler (ordonnanceur complètement équitable)

Ce nouveau mode de planification assure que le runtime est partagé de manière équitable entre les processus éligibles. Combiner celui-ci à la détection de topologie permet aux processus d'être planifiés sur des CPU dans le même socket afin d'éviter de développer le besoin d'un accès mémoire distant coûteux et d'assurer que le contenu du cache soit préservé lorsque possible.

`malloc`

`malloc` est maintenant optimisé pour assurer que les régions de la mémoire qui sont allouées à un processus soient physiquement aussi proches que possible du cœur sur lequel le processus est en cours d'exécution. Ceci augmente la vitesse d'accès mémoire.

allocation du tampon d'E/S skbuff

De la même manière que **`malloc`**, ceci est maintenant optimisé de manière à utiliser la mémoire qui est physiquement proche des opérations d'E/S de gestion de CPU, comme les interruptions de périphériques.

affinités des interruptions de périphériques

Informations enregistrées par des pilotes de périphérique sur quels identificateurs de CPU effectuant des interruptions peuvent être utilisés pour restreindre la gestion des interruptions à des CPU situés dans le même socket physique, préservant les affinités du cache et limitant les communications haut volume à travers les sockets.

4.4.1.2. Améliorations de la synchronisation de multiples processeurs

Coordonner des tâches entre multiples processeurs requiert des opérations fréquentes et assez longues afin de s'assurer que les processus exécutés en parallèle ne compromettent pas l'intégrité des données. Red Hat Enterprise Linux inclut les améliorations suivantes pour obtenir de meilleures performances dans ce domaine.

Verrous RCU (« Read-Copy-Update »)

Habituellement, 90% des verrous sont acquis à des fins de lecture-seule.Le verrouillage RCU supprime le besoin d'obtenir un verrou d'accès exclusif lorsque les données accédées ne sont pas modifiées. Ce mode de verrouillage est maintenant utilisé dans l'allocation de mémoire du cache de page : le verrouillage est uniquement utilisé pour les opérations d'allocation ou de dés-allocation.

Algorithmes « par CPU » et « par socket »

De nombreux algorithmes ont été mis à jour pour réaliser une coordination des verrous parmi les CPU coopérant sur le même socket afin de permettre un meilleur verrouillage à grains fins. De nombreux verrous tournants globaux ont été remplacés par des méthodes de verrouillage « par

socket » et la mise à jour des zones de l'allocateur de mémoire et des listes de pages mémoire liées permettent à la logique de l'allocation de mémoire de traverser un sous-ensemble des structures de données mappant la mémoire plus efficace lors des opérations d'allocation et de dés-allocation.

4.4.2. Optimisation de la virtualisation

Comme KVM utilise la fonctionnalité du noyau, les invités basés KVM bénéficient immédiatement de toutes les optimisations bare-metal. Red Hat Enterprise Linux inclut aussi un certain nombre d'améliorations permettant aux invités virtualisés de s'approcher du niveau de performance d'un système bare-metal. Ces améliorations se concentrent sur le chemin des E/S dans l'accès réseau et stockage, permettant même des charges de travail intensives, comme des bases de données et des services des fichiers utilisant le déploiement virtualisé. Les améliorations spécifiques à NUMA qui améliorent les performances des systèmes virtualisés incluent :

CPU pinning

Les invités virtuels peuvent être tenus de s'exécuter sur un socket en particulier afin d'optimiser l'utilisation du cache local et de supprimer le besoin d'effectuer de coûteuses communications inter-sockets et d'effectuer des accès mémoire distants.

transparent hugepages (THP)

Lorsque THP est activé, le système effectue automatiquement des requêtes d'allocation de mémoire conscientes de NUMA pour des quantités de mémoire de grande taille, réduisant la contention de verrous et le nombre d'opérations de gestion de mémoire TLB (« Translation Lookaside Buffer ») requises et générant une augmentation des performances allant jusqu'à 20% quant aux invités virtuels.

Implémentation des E/S basées sur le noyau

Le sous-système des E/S de l'invité virtuel est maintenant implémenté dans le noyau, réduisant fortement le coût des communications inter-nœuds et des accès mémoire en évitant une quantité importante de basculements de contextes et de surcharges de la synchronisation et des communications.

CHAPITRE 5. MÉMOIRE

Veuillez lire ce chapitre pour avoir un aperçu des fonctionnalités de gestion de mémoire disponibles avec Red Hat Enterprise Linux et pour savoir comment utiliser ces fonctionnalités de gestion afin d'optimiser l'utilisation de mémoire dans votre système.

5.1. HUGETLB (« HUGE TRANSLATION LOOKASIDE BUFFER »)

Les adresses de la mémoire physique sont traduites en adresses de mémoire virtuelle dans le cadre de la gestion de la mémoire. La relation mappée des adresses physiques aux adresses virtuelles est stockée dans une structure de données appelée la table des pages. Comme la lecture de la table des pages pour chaque mappage d'adresse prendrait du temps et serait coûteuse en termes de ressources, il existe un cache pour les adresses récemment utilisées. Ce cache est se nomme le TLB (« Translation Lookaside Buffer »).

Cependant, le TLB peut uniquement mettre en cache un certain nombre de mappages d'adresses. Si un mappage d'adresse requis ne se trouve pas dans le TLB, la table des pages devra tout de même lire pour déterminer le mappage entre l'adresse physique et l'adresse virtuelle. Ceci est appelé raté TLB (de l'anglais, « TLB miss »). Les applications ayant des besoins importants en mémoire sont les plus susceptibles d'être affectées par les ratés TLB, plus que les applications ayant des besoins en mémoire minimes, ce à cause de la relation entre leurs besoins en mémoire et la taille des pages utilisées pour mettre en cache les mappages dans le TLB. Puisque chaque raté implique la lecture de la table des pages, il est important d'éviter ces ratés à chaque fois que possible.

HugeTLB (« Huge Translation Lookaside Buffer ») permet à la mémoire d'être gérée en segments de très grande taille afin que davantage de mappages d'adresses puissent être mis en cache à la fois. Ceci réduit la probabilité de ratés TLB, ce qui se traduit par une amélioration des performances des applications ayant des besoins importants en mémoire.

Des informations sur la configuration de HugeTLB peuvent être trouvées dans la documentation du noyau : **/usr/share/doc/kernel-doc-*version*/Documentation/vm/hugetlbpage.txt**

5.2. « HUGE PAGES » ET « TRANSPARENT HUGE PAGES »

La mémoire est gérée en blocs appelés des *pages*. Une page fait 4096 octets. 1 Mo de mémoire est équivalent à 256 pages, 1 Go de mémoire équivaut à 256 000 pages, etc. Les CPU possèdent une *unité de gestion de mémoire* intégrée qui contient une liste de ces pages, avec chaque page référencée via une *entrée de table de pages*.

Il existe deux manières d'activer le système afin qu'il gère de grandes quantités de mémoire :

- Augmenter le nombre d'entrées de tables de pages dans l'unité de gestion de mémoire du matériel
- Augmenter la taille de page

La première méthode est coûteuse puisque l'unité de gestion de mémoire du matériel dans un processeur moderne prend uniquement en charge des centaines ou des milliers d'entrées de tables de pages. En outre, les algorithmes de gestion de mémoire et de matériel qui fonctionnent bien avec des milliers de pages (mégaoctets de mémoire) peuvent avoir des difficultés avec des millions (ou même des milliards) de pages. Il en résulte des problèmes de performance : lorsqu'une application doit utiliser plus de pages de mémoire que ce que l'unité de gestion de mémoire prend en charge, alors le système bascule sur une gestion de mémoire plus lente et basée sur logiciel, ce qui cause au système entier de fonctionner plus lentement.

Red Hat Enterprise Linux 6 implémente la seconde méthode via l'utilisation des *huge pages*.

Les huges pages sont des blocs de mémoire d'une taille de 2 Mo et de 1 Go. La table des pages utilisée par les pages de 2 Mo sont convenables pour gérer de multiples gigaoctets de mémoire, alors que les tables de pages des pages faisant 1 Go sont mieux pour étendre sur des téraoctets de mémoire.

Les huges pages doivent être assignées au moment du démarrage. Elles sont aussi difficiles à gérer manuellement et requièrent souvent des modifications significatives du code afin d'être utilisées de manière efficace. Ainsi, Red Hat Enterprise Linux 6 a aussi implémenté l'utilisation de THP (*transparent huge pages*). THP est une couche d'abstraction qui automatise les plupart des aspects de la création, de la gestion et de l'utilisation de huges pages

Les THP cachent une grande partie de la complexité de l'utilisation des huges pages aux administrateurs et développeurs. Comme le but de THP est d'améliorer les performances, ses développeurs (de la communauté et de Red Hat) ont testé et optimisé les THP sur un large éventail de systèmes, de configurations, d'applications et de charges de travail. Ceci permet aux paramètres par défaut des THP d'améliorer les performances de la plupart des configurations de système.

Remarquez qu'actuellement, THP peut uniquement mapper des régions de mémoire anonymes, telles que les espaces du tas et de la pile.

5.3. UTILISER VALGRIND POUR ÉTABLIR UN PROFIL DE L'UTILISATION DE MÉMOIRE

Valgrind est un framework qui fournit l'instrumentation des binaires de l'espace utilisateur. Il est fournit avec un certain nombre d'outils pouvant être utilisés pour profiler et analyser les performances de programmes. Les outils décrits dans cette section offrent une analyse pouvant aider à la détection d'erreurs de mémoire telles que l'utilisation de mémoire non-initialisée et l'allocation ou la dés-allocation de mémoire incorrecte. Tous sont inclus dans le paquetage valgrind et peuvent être exécutés avec la commande suivante :

```
valgrind --tool=toolname program
```

Remplacez *toolname* par le nom de l'outil que vous souhaitez utiliser (pour profiler la mémoire, **memcheck**, **massif**, ou **cachegrind**) et *program* par le programme que vous souhaitez profiler avec Valgrind. Soyez conscient que l'instrumentation de Valgrind causera à votre programme de fonctionner plus lentement que normalement.

Un aperçu des capacités de Valgrind est offert dans la Section 3.5.3, « Valgrind ». Des détails supplémentaires, y compris des informations sur les plugins disponibles pour Eclipse, sont inclus dans le *Guide du développeur*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/. La documentation l'accompagnant peut être affichée avec la commande **man valgrind** lorsque le paquetage valgrind est installé ou trouvé dans les emplacements suivants :

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

5.3.1. Profiler l'utilisation de mémoire avec Memcheck

Memcheck est l'outil Valgrind par défaut et peut être exécuté avec **valgrind *program***, sans spécifier **--tool=memcheck**. Il détecte et rapporte un certain nombre d'erreurs de mémoire pouvant être difficiles à détecter et diagnostiquer, comme les accès mémoire qui n'auraient pas dû se produire,

l'utilisation de valeurs non-définies ou non-initialisées, la libération incorrecte de la mémoire du tas, les pointeurs superposés et les fuites de mémoire. Avec Memcheck, les programmes fonctionnent dix à trente fois plus lentement que normalement.

Memcheck retourne des erreurs spécifiques selon le type de problème rencontré. Ces erreurs sont décrites en détails dans la documentation Valgrind incluse sur **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

Remarquez que Memcheck peut uniquement rapporter ces erreurs, il ne peut pas les empêcher de se produire. Si votre programme accède à la mémoire d'une manière qui aurait normalement résulté avec un défaut de segmentation, cette mauvaise segmentation se produira tout de même. Cependant, Memcheck journalisera un message d'erreur immédiatement avant la faute.

Memcheck fournit des options de ligne de commande pouvant être utilisées pour préciser le processus de vérification. Certaines des options disponibles incluent :

--leak-check

Lorsque activé, Memcheck recherches des fuites de mémoire quand le programme client se termine. La valeur par défaut est **summary**(sommaire), ce qui retourne le nombre de fuites découvertes. Les autres valeurs possibles sont **yes** (oui) et **full** (plein), qui offrent toutes deux des détails sur chaque fuite individuelle et **no** (non), qui désactive la vérification de fuites de mémoire.

--undef-value-errors

Quand activé (définit sur **yes**), Memcheck rapporte des erreurs lorsque des valeurs non-définies sont utilisées. Quand désactivé (définit sur **no**), les erreurs de valeurs non-définies ne sont pas rapportées. Cette option est activée par défaut. La désactiver accélère légèrement Memcheck.

--ignore-ranges

Permet à l'utilisateur de spécifier une ou plusieurs gammes que Memcheck devrait ignorer lors de la vérification de l'adressage. De multiples gammes sont délimitées par des virgules. Par exemple, **--ignore-ranges=0xPP-0xQQ,0xRR-0xSS**.

Pour une liste complète des options, veuillez vous reporter à la documentation incluse sur **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

5.3.2. Profiler l'utilisation du cache avec Cachegrind

Cachegrind simule l'interaction de votre programme avec la hiérarchie du cache de votre machine et (optionnellement) avec le prédicteur de branches. Il traque l'utilisation des instructions de premier niveau simulées et des caches de données pour détecter de mauvaises interactions de code avec ce niveau de cache, ainsi que le dernier niveau de cache, qu'il s'agisse d'un second ou troisième niveau de cache, afin de traquer l'accès à la mémoire principale. Ainsi, les programmes exécutés avec Cachegrind fonctionnent 20 à 100 fois plus lentement que normalement.

Pour exécuter Cachegrind, exécutez la commande suivante en remplaçant *program* par le programme que vous souhaitez profiler avec Cachegrind :

```
# valgrind --tool=cachegrind program
```

Cachegrind peut rassembler les statistiques suivantes pour le programme entier, ainsi que pour chaque fonction dans le programme :

- lectures du cache des instructions de premier niveau (ou instructions exécutées) et échecs de lecture, ainsi que les échecs de lecture des instructions du cache du dernier niveau ;
- lectures du cache de données (ou lectures de mémoire), échecs des lectures et échecs des lectures de données du cache du dernier niveau ;
- écritures du cache de données (ou écritures de mémoire), échecs des écritures et échecs des écritures du cache du dernier niveau ;
- branches conditionnelles exécutées et mal prédites ;
- branches indirectes exécutées et mal prédites.

Cachegrind imprime des informations sommaires à propos de ces statistiques sur la console et écrit des informations de profilage plus détaillées sur un fichier (par défaut **cachegrind.out.*pid***, où *pid* est l'ID de processus du programme sur lequel vous avez exécuté Cachegrind). Ce fichier peut encore être traité par l'outil accompagnateur **cg_annotate**, ainsi :

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

NOTE

cg_annotate peut délivrer en sortie des lignes de plus de 120 caractères, selon la longueur du chemin. Pour rendre la sortie plus claire et plus facile à lire, il est recommandé d'élargir votre fenêtre terminal au minimum à cette même taille avant d'exécuter la commande mentionnée ci-dessus.

Vous pouvez aussi comparer les fichiers de profils créés par Cachegrind afin de faciliter l'impression des performances du programme avant et après un changement. Pour ce faire, utilisez la commande **cg_diff**, en remplaçant *first* par le fichier de sortie du profil initial et *second* par le fichier de sortie du profil suivant :

```
# cg_diff first second
```

Cette commande produit un fichier de sortie combiné, qui peut être affiché avec plus de détails avec **cg_annotate**.

Cachegrind prend en charge un certain nombre d'options pour concentrer sa sorte. Voici quelques-unes des options disponibles :

--I1

Spécifie la taille, l'associativité et la taille de ligne du cache d'instructions du premier niveau, séparé par des virgules : **--I1=*size*,*associativity*,*line size***.

--D1

Spécifie la taille, l'associativité et la taille de ligne du cache de données du premier niveau, séparé par des virgules : **--D1=*size*,*associativity*,*line size***.

--LL

Spécifie la taille, l'associativité et la taille de ligne du cache du dernier niveau, séparé par des virgules : **--LL=*size*,*associativity*,*line size***.

--cache-sim

Active ou désactive la collection des accès au cache et du compte des échecs. La valeur par défaut est **yes** (activé).

Remarquez que désactiver cette option ainsi que **--branch-sim** laisse Cachegrind sans la moindre information à collecter.

--branch-sim

Active ou désactive la collection des instructions de branches et du compte des mauvaises prédictions. Cette option est définie sur **no** (désactivé) par défaut car elle ralentit Cachegrind d'environ 25 pour cent.

Remarquez que désactiver cette option ainsi que **--cache-sim** laisse Cachegrind sans la moindre information à collecter.

Pour une liste complète des options, veuillez vous reporter à la documentation incluse sur **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

5.3.3. Profiler les espaces du tas et de pile avec Massif

Massif mesure l'espace du tas utilisé par un programme spécifique ; l'espace utile, ainsi que tout espace supplémentaire alloué à la comptabilité et à des fins d'alignement. Il peut vous aider à réduire la quantité de mémoire utilisée par votre programme, ce qui peut augmenter la vitesse de votre programme et réduire les chances que votre programme n'épuise l'espace swap de la machine sur laquelle il est exécuté. Massif peut aussi fournir des détails sur les parties de votre programme qui sont responsables pour l'allocation de la mémoire du tas. Les programmes exécutés avec Massif fonctionnent environ 20 fois plus lentement que leur vitesse d'exécution normale.

Pour profiler l'utilisation du tas d'un programme, spécifiez **massif** comme l'outil Valgrind que vous souhaitez utiliser :

```
# valgrind --tool=massif programme
```

Les données de profilage rassemblées par Massif sont écrites sur un fichier, qui est appelé par défaut **massif.out.*pid***, où *pid* est l'ID de processus du *programme* spécifié.

Les données de profilage peuvent aussi être cartographiées avec la commande **ms_print**, ainsi :

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

Ceci produit un graphe affichant la consommation de mémoire pendant l'exécution du programme, ainsi que des informations détaillées sur les sites responsables de l'allocation à divers points du programme, y compris au summum de l'allocation de mémoire.

Massif offre un nombre d'options de ligne de commande pouvant être utilisées pour diriger la sortie de l'outil. Certaines des options disponibles incluent :

--heap

Spécifie s'il faut effectuer le profilage du tas. La valeur par défaut est **yes** (oui). Le profilage du tas peut être désactivé en définissant cette option sur **no** (non).

--heap-admin

Spécifie le nombre d'octets par bloc à utiliser pour l'administration lorsque le profilage du tas est activé. La valeur par défaut est de **8** octets par bloc.

--stacks

Spécifie s'il faut effectuer le profilage de la pile. La valeur par défaut est **no** (désactivé). Pour activer le profilage de la pile, définissez cette option sur **yes**, mais n'oubliez pas que cela va fortement ralentir Massif. Remarquez aussi que Massif suppose que la pile principale a une taille de zéro au démarrage afin de mieux indiquer la taille de la portion de la pile sur laquelle le programme en cours de profilage est en contrôle.

--time-unit

Spécifie l'unité de temps utilisée pour le profilage. Il existe trois valeurs valides pour cette option : les instructions exécutées (**i**), la valeur par défaut, qui est utile dans la plupart des cas ; le temps réel (**ms**, en millisecondes), qui peut être utile dans certains cas ; et les octets alloués ou dés-alloués sur le tas et/ou sur la pile (**B**), ce qui est utile pour des programmes exécutés de manière très rapide et pour effectuer des tests, car ceux-ci sont les plus facilement reproductibles sur différentes machines. Cette option est utile lors de l'impression du graphe de la sortie de Massif avec **ms_print**.

Pour une liste complète des options, veuillez vous reporter à la documentation incluse sur **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

5.4. RÉGLAGE DES CAPACITÉS

Veuillez lire cette section pour des informations sur la mémoire, le noyau et les capacités du système de fichiers, les paramètres liés à chacun d'entre eux, et les compromis impliqués dans l'ajustement de ces paramètres.

Pour définir ces valeurs temporairement pendant le réglage, effectuez une opération echo de la valeur souhaitée sur le fichier correspondant dans le système de fichiers proc. Par exemple, pour définir ***overcommit_memory*** temporairement sur **1**, exécutez :

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

Remarquez que le chemin vers le paramètre dans le système de fichiers proc varie en fonction du système affecté par le changement.

Pour définir ces valeurs de manière persistante, vous devrez utiliser la commande **sysctl**. Pour obtenir des détails supplémentaires, veuillez consulter le *Guide de déploiement*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Réglages de mémoire liés aux capacités

Chacun des paramètres suivants est placé sous **/proc/sys/vm/**, dans le système de fichiers proc.

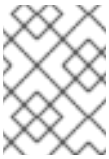
overcommit_memory

Définit les conditions déterminant si une requête mémoire de grande taille est acceptée ou non. Trois valeurs sont possibles pour ce paramètre :

- **0** — Paramètre par défaut. Le noyau effectue la gestion de l'« overcommit » de mémoire heuristique en estimant la quantité de mémoire disponible et en refusant les requêtes qui sont manifestement invalides. Malheureusement, comme la mémoire est allouée avec une

heuristique plutôt qu'avec un algorithme précis, ce paramètre peut parfois autoriser la surcharge de la mémoire disponible sur le système.

- **1** — Le noyau n'effectue pas la gestion de l'« overcommit » de la mémoire. Sous ce paramètre, le potentiel pour la surcharge de la mémoire est augmenté, mais les performances aussi pour les tâches demandant beaucoup de mémoire.
- **2** — Le noyau refuse les requêtes de mémoire supérieures ou égales à la somme du total de l'espace swap disponible et du pourcentage de RAM physique spécifié dans ***overcommit_ratio***. Ce paramètre est le meilleur si vous souhaitez réduire le risque d'« overcommit » de mémoire.



NOTE

Ce paramètre est uniquement recommandé pour les systèmes avec des zones swap de plus grande taille que leur mémoire physique.

overcommit_ratio

Spécifie le pourcentage de RAM physique considérée lorsque ***overcommit_memory*** est paramétré sur **2**. La valeur par défaut est **50**.

max_map_count

Définit le nombre maximal de zones de cartes mémoires qu'un processus peut utiliser. Dans la plupart des cas, la valeur par défaut **65530** est appropriée. Augmentez cette valeur si votre application doit cartographier plus que ce nombre de fichiers.

nr_hugepages

Définit le nombre de hugepages configurées dans le noyau. La valeur par défaut est 0. Il est uniquement possible d'allouer (ou de dés-allouer) des hugepages si suffisamment de pages libres physiquement contiguës se trouvent dans le système. Les pages réservées par ce paramètre ne peuvent pas être utilisées à d'autres fins. Des informations supplémentaires sont disponibles dans la documentation installée : **/usr/share/doc/kernel-doc-*kernel_version*/Documentation/vm/hugetlbpage.txt**

Réglages noyau liés aux capacités

Chacun des paramètres suivants est placé sous **/proc/sys/kernel/**, dans le système de fichiers proc.

msgmax

Définit la taille maximale autorisée en octets de tout message unique dans une file d'attente de messages. Cette valeur ne doit pas excéder la taille de la file (***msgmnb***). La valeur par défaut est **65536**.

msgmnb

Définit la taille maximale en octets d'une seule file d'attente de messages. La valeur par défaut est **65536**.

msgmni

Définit le nombre maximal d'identifiants de files d'attente de messages (et donc le nombre maximal de files). La valeur par défaut sur les machines avec une architecture 64 bits est **1985** ; pour les architectures 32 bits, la valeur par défaut est de **1736**.

shmall

Définit la quantité totale en octets de mémoire partagée pouvant être utilisée sur le système à la fois. La valeur par défaut pour les machines avec une architecture de 64 bits est de **4294967296** ; pour les architectures de 32 bits, la valeur est de **268435456**.

shmmax

Définit le segment de mémoire partagée maximal autorisé par le noyau, en octets. La valeur par défaut sur les machines avec une architecture de 64 bits est **68719476736** ; avec une architecture de 32 bits, la valeur par défaut est **4294967295**. Remarquez cependant que le noyau prend en charge des valeurs bien plus élevées que celles-ci.

shmmni

Définit le nombre de segments de mémoire partagée maximal sur tout le système. La valeur par défaut est **4096** sur les architectures 32 et 64 bits.

threads-max

Définit le nombre maximal sur tout le système de threads (tâches) pouvant être utilisés par le noyau à la fois. La valeur par défaut est égale à la valeur ***max_threads*** du noyau. La formule utilisée est :

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

La valeur minimale de ***threads-max*** est **20**.

Réglages des systèmes de fichiers liés aux capacités

Chacun des paramètres suivants est placé sous **/proc/sys/fs/**, dans le système de fichiers proc.

aio-max-nr

Définit le nombre maximal d'événements autorisés dans tous les contextes d'E/S asynchrones actifs. La valeur par défaut est **65536**. Remarquez que modifier cette valeur ne pré-allouera ou ne redimmensionnera aucune structure de données du noyau.

file-max

Répertorie le nombre maximal de gestionnaires de fichiers que le noyau alloue. La valeur par défaut correspond à la valeur de ***files_stat.max_files*** dans le noyau, qui est défini sur la valeur la plus élevée de **(mempages * (PAGE_SIZE / 1024)) / 10**, ou **NR_FILE** (8192 dans Red Hat Enterprise Linux). Augmenter cette valeur peut résoudre des erreurs causées par un manque de gestionnaires de fichiers disponibles.

Réglages « Kill » OOM

OOM (« Out of Memory », ou mémoire saturée) fait référence à un état informatique où toute la mémoire disponible, y compris l'espace swap, a été alloué. Par défaut, cette situation cause au système de paniquer et d'arrêter de fonctionner comme il le devrait. Cependant, définir le paramètre

/proc/sys/vm/panic_on_oom sur **0** instruit au noyau d'appeler la fonction **oom_killer** lorsque se produit une saturation de mémoire OOM . Normalement, **oom_killer** peut interrompre les processus et le système survit.

Le paramètre suivant peut être défini sur une base « par processus », vous offrant ainsi un contrôle amélioré sur quels processus sont interrompus par la fonction **oom_killer**. Il se trouve sous **/proc/*pid*/** dans le système de fichiers proc, où *pid* est le numéro d'ID du processus.

oom_adj

Définit une valeur de **-16** à **15** qui aide à déterminer l'***oom_score*** d'un processus. Plus la valeur de ***oom_score*** est élevée, plus il est possible que le processus sera interrompu par **oom_killer**. Définir une valeur ***oom_adj*** de **-17** désactivera **oom_killer** pour ce processus.

IMPORTANT

Tout processus engendré par un processus ajusté héritera de la valeur ***oom_score*** de ce processus. Par exemple, si un processus **sshd** est protégé par la fonction **oom_killer**, tous les processus initiés par cette session SSH seront aussi protégés. Ceci peut affecter la capacité de la fonction **oom_killer** à sauver le système si une saturation de mémoire OOM se produit.

5.5. RÉGLAGES DE LA MÉMOIRE VIRTUELLE

La mémoire virtuelle est habituellement consommée par les processus, les caches de systèmes de fichiers et le noyau. L'utilisation de la mémoire virtuelle dépend d'un certain nombre de facteurs qui peuvent être affectés par les paramètres suivants :

swappiness

Une valeur de 0 à 100 qui contrôle le degré auquel le système effectue des swaps. Une valeur élevée donnera la priorité aux performances du système, effectuant des swaps de manière agressive pour pousser les processus hors de la mémoire physique lorsqu'ils ne sont pas actifs. Une valeur faible donne priorité à l'interactivité et évite de swapper les processus hors de la mémoire physique aussi longtemps que possible, ce qui réduit la latence des réponses. La valeur par défaut est **60**.

min_free_kbytes

Nombre minimum de kilooctets à garder libres à travers le système. Cette valeur est utilisée pour calculer une valeur limite pour chaque zone de mémoire basse, celles-ci se voient ensuite assigner un nombre de pages libres réservées proportionnel à leur taille.

AVERTISSEMENT

Soyez prudent lorsque vous définissez ce paramètre, car des valeurs trop élevées ou trop basses peuvent endommager le système.

Définir le paramètre ***min_free_kbytes*** trop bas empêche le système de réclamer de la mémoire. Ceci peut résulter en la suspension du système et l'interruption pour cause de mémoire saturée de multiples processus.

Cependant, définir ce paramètre sur une valeur trop élevée (5 à 10% de la mémoire système totale) causera à votre système de se retrouver avec une mémoire saturée immédiatement. Linux est désigné pour utiliser toute la mémoire vive disponible pour mettre en cache les données de système de fichiers. Définir une valeur ***min_free_kbytes*** élevée fera dépenser au système trop de temps à réclamer de la mémoire.

dirty_ratio

Définit une valeur de pourcentage. La réécriture des données modifiées débutera (via **pdflush**) lorsque les données modifiées comprendront ce pourcentage de mémoire système totale. La valeur par défaut est de **20**.

dirty_background_ratio

Définit une valeur de pourcentage. La réécriture des données modifiées débutera dans l'arrière-plan (via **pdflush**) lorsque les données modifiées comprendront ce pourcentage de mémoire totale. La valeur par défaut est de **10**.

drop_caches

Définir cette valeur sur **1**, **2**, ou **3** cause au noyau d'abandonner diverses combinaisons de caches de pages et de caches de dalles.

1

Le système effectue une invalidation et libère toute la mémoire du cache des pages.

2

Le système libère toute la mémoire du cache de dalle non-utilisée.

3

Le système libère toute la mémoire du cache de pages et du cache de dalle.

Ceci est une opération non-destructive. Comme les objets modifiés ne peuvent pas être libérés, exécuter **sync** avant de définir la valeur de ce paramètre est recommandé.

IMPORTANT

Utiliser ***drop_caches*** pour libérer la mémoire n'est pas recommandé dans un environnement de production.

Pour définir ces valeurs de manière temporaire pendant les réglages, effectuez une opération echo de la valeur souhaitée dans le fichier correspondant dans le système de fichiers proc. Par exemple, pour définir ***swappiness*** temporairement sur **50**, veuillez exécuter :

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

Pour définir cette valeur de manière persistante, vous devrez utiliser la commande **sysctl**. Pour obtenir davantage d'informations, veuillez consulter le *Guide de déploiement*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

CHAPITRE 6. ENTRÉES/SORTIES

6.1. FONCTIONNALITÉS

Red Hat Enterprise Linux 6 présente un certain nombre d'amélioration des performances dans la pile des E/S :

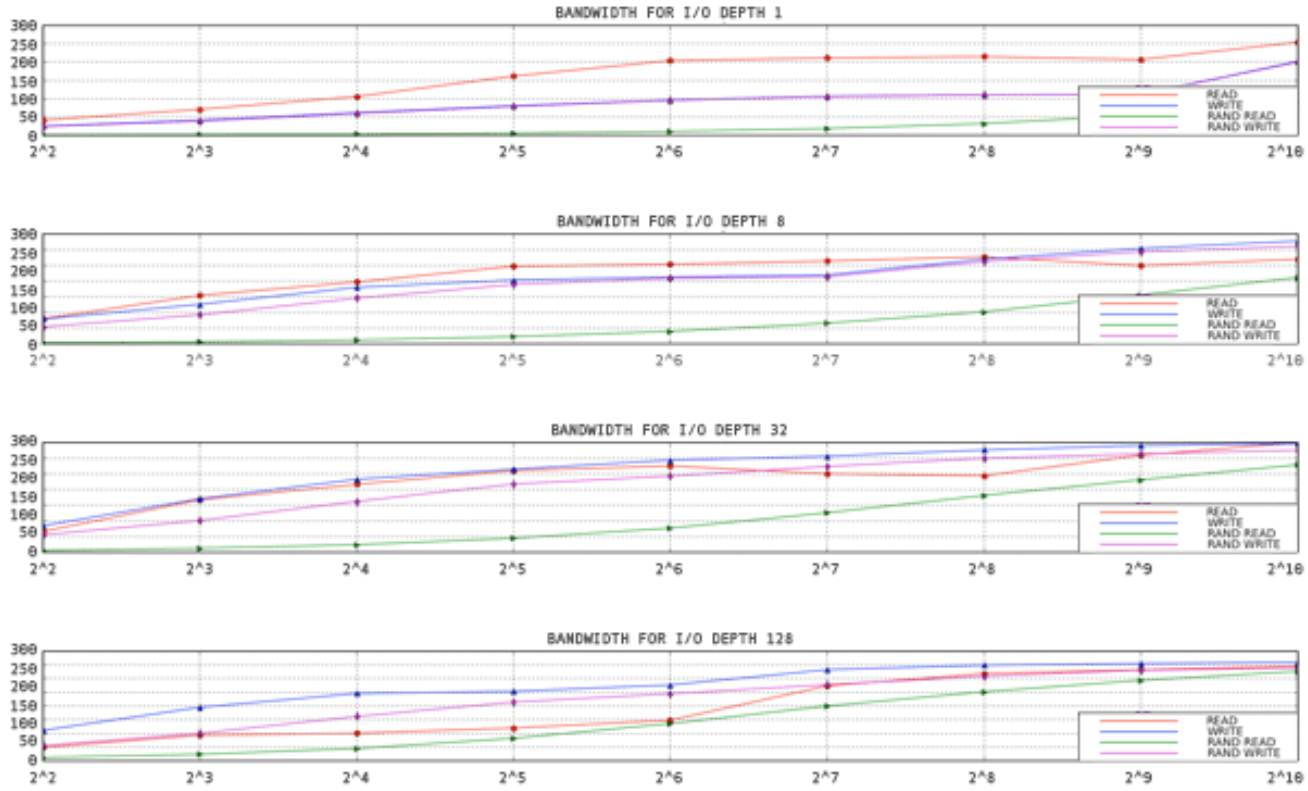
- Les disques SSD sont maintenant officiellement reconnus automatiquement et les performances de l'ordonnanceur d'E/S sont réglées pour tirer profit du haut niveau d'E/S par secondes (« IOPS ») que ces périphériques peuvent effectuer.
- La prise en charge de l'annulation a été ajoutée au noyau pour signaler les gammes de blocs inutilisées au stockage sous-jacent. Ceci aide les disques SSD avec leurs algorithmes d'égalisation de l'usure. Cela aide aussi le stockage qui prend en charge l'approvisionnement des blocs logiques (une sorte d'espace d'adresse virtuelle pour le stockage) en suivant de plus près la quantité réelle de stockage en cours d'utilisation.
- L'implémentation de la barrière du système de fichiers a été révisée dans Red Hat Enterprise Linux 6.1 afin de la rendre plus performante.
- **`pdflush`** a été remplacé par des threads de vidage « par périphérique de sauvegarde », qui améliorent grandement l'évolutivité du système sur des configurations avec un nombre important de LUN.

6.2. ANALYSE

Régler les performances de la pile de stockage correctement requiert une certaine compréhension de la manière par laquelle les données traversent le système, ainsi que la connaissance intime du stockage sous-jacent et de ses performances sous différentes charges de travail. Il est aussi nécessaire de comprendre comment la charge de travail est réellement réglée.

Lorsque vous déployez un nouveau système, il est recommandé de profiler le stockage de bas en haut. Commencez par des LUN ou des disques bruts et évaluez leurs performances à l'aide d'E/S directes (des E/S qui contournent le cache de la page du noyau). Ce test est le test le plus basique que vous pouvez effectuer et sera le standard avec lequel vous mesurerez les performances des E/S dans la pile. Commencez par un générateur de charges de travail de base (tel que **aio-stress**), qui produit des lectures et écritures séquentielles et aléatoires à travers toute une gamme de tailles d'E/S et profondeurs de files d'attente.

Ci-dessous figure un graphique provenant d'une série d'exécutions de **aio-stress**, chacune d'entre elles passe par quatre étapes : une écriture séquentielle, une lecture séquentielle, une écriture aléatoire et une lecture aléatoire. Dans cet exemple, l'outil est configuré pour être exécuté sur toute une gamme de tailles d'enregistrements (l'axe x) et profondeurs de files d'attente (une par graphique). La profondeur des files d'attente représente le nombre total d'opérations d'E/S en progrès à un moment donné.



L'axe « y » montre la bande passante en mégaoctets par seconde. L'axe « x » montre la taille des E/S en kilo-octets.

Figure 6.1. Sortie aio-stress pour 1 thread, 1 fichier

Remarquez comme la ligne de la bande passante tend à passer du coin inférieur gauche au coin supérieur droit. Remarquez aussi que, pour une taille d'enregistrement donnée, la bande passante du stockage peut être plus importante en augmentant le nombre d'E/S en cours d'exécution.

En exécutant ces simples charges de travail sur votre stockage, vous comprendrez mieux comment fonctionne votre stockage sous des charges. Conservez les données générées par ces tests pour effectuer des comparaisons lors de l'analyse de charges de travail plus complexes.

Si vous allez utiliser le mappeur de périphériques (« device-mapper ») ou md, veuillez ajouter cette couche et répéter les tests. S'il y a une forte baisse des performances, assurez-vous que celle-ci était prévue, ou tout du moins qu'elle puisse être expliquée. Par exemple, on peut s'attendre à une baisse des performances si une couche RAID effectuant des checksums a été ajoutée à la pile. Des baisses de performances inattendues peuvent être causées par des opérations d'E/S mal alignées. Par défaut, Red Hat Enterprise Linux aligne ses partitions et métadonnées de mappeur de périphériques de manière optimale. Cependant, les types de stockage ne rapportent pas tous leur alignement optimal, et peuvent ainsi nécessiter des réglages manuels.

Après avoir ajouté le mappeur de périphériques ou la couche md, veuillez ajouter un système de fichiers sur le haut du périphérique bloc et effectuez des tests dessus, tout en continuant d'utiliser des E/S directes. Une fois de plus, comparez les résultats avec ceux des tests précédents et assurez-vous de bien comprendre tout écart. Habituellement, les E/S directes fonctionnent mieux sur des fichiers pré-alloués, assurez-vous donc de bien pré-allouer les fichiers avant de tester les performances.

Les générateurs de charges de travail synthétiques que vous trouverez utiles incluent :

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

- **fio**

6.3. OUTILS

De nombreux outils sont disponibles pour aider à diagnostiquer les problèmes de performance dans le sous-système des E/S. **vmstat** fournit un aperçu grossier des performances du système. Les colonnes suivantes concernent davantage les E/S : ***si*** (swap in), ***so*** (swap out), ***bi*** (block in), ***bo*** (block out) et ***wa*** (temps d'attente des E/S). ***si*** et ***so*** sont utiles lorsque votre espace swap est situé sur le même périphérique que la partition de vos données et sont un indicateur de la pression mémoire générale. ***si*** et ***bi*** sont des opérations de lecture, tandis que ***so*** et ***bo*** sont des opérations d'écriture. Chacune de ces catégories est rapportée en kilo-octets. ***wa*** est le temps d'inactivité, il indique quelle portion de la file d'attente d'exécution est bloquée, en train d'attendre que les E/S soient terminées.

L'analayse de votre système avec **vmstat**vous indiquera si le sous-système des E/S pourrait être responsable des problèmes de performance ou non. Veuillez aussi prendre note des colonnes ***free***, ***buff*** et ***cache***. L'augmentation de la valeur ***cache*** aux côtés de la valeur ***bo***, suivie par une baisse de ***cache*** et une augmentation de ***free*** indique que le système est en train d'effectuer une ré-écriture et une invalidation du cache de la page.

Remarquez que les numéros des E/S rapportées par **vmstat** sont des agrégations de toutes les E/S sur tous les périphériques. Une fois que vous avez déterminé qu'il y a un écart de performance dans le sous-système des E/S, vous pouvez examiner le problème de plus près avec **iostat**, ce qui divisera les E/S rapportées par périphérique. Vous pouvez aussi récupérer des informations plus détaillées, comme la taille de requête moyenne, le nombre de lecture et écritures par seconde, ainsi que le nombre de fusions d'E/S en cours.

En vous aidant de la taille de requête moyenne et de la taille de file d'attente moyenne (**avgqu-sz**), vous pourrez effectuer quelques estimations sur la manière par laquelle le stockage devrait fonctionner en utilisant les graphes générés lors de la caractérisation des performances de votre stockage. Certaines généralisations s'appliquent, par exemple : si la taille de requête moyenne est de 4 Ko et que la taille de file d'attente moyenne est de 1, il est improbable que le débit soit très performant.

Si les chiffres des performances ne correspondent pas aux performances auxquelles vous vous attendez, vous pouvez effectuer une analyse plus précise avec **blktrace**. L'ensemble d'utilitaires **blktrace** vous offre des informations précises sur le temps passé dans le sous-système des E/S. La sortie de **blktrace** est un ensemble de fichiers de suivi pouvant être traités après-coup par d'autres utilitaires comme **blkparse**.

blkparse est l'utilitaire complémentaire à **blktrace**. Il lit les sorties brutes du suivi est produit une courte version texte.

Ci-dessous figure un exemple de sortie 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]
```

Comme vous pouvez le constater, la sortie est dense et difficile à lire. Vous pouvez voir quels processus sont responsables des E/S effectuées sur votre périphérique et lesquels sont utiles, mais dans son résumé, **blkparse** vous offre des informations supplémentaires sous un format plus facile à assimiler. Les informations du résumé de **blkparse** sont imprimées à la fin de sa sortie :

```
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
```

Le résumé affiche les taux d'E/S moyens, l'activité liée aux fusions et compare la charge de travail de lecture avec la charge de travail d'écriture. Néanmoins, la plupart du temps, la sortie de **blkparse** est trop volumineuse pour être utilisée en tant que telle. Fort heureusement, il existe plusieurs outils pour assister à la visualisation des données.

btt fournit une analyse du temps pris par les E/S dans les différentes zones de la pile d'E/S. Ces zones sont les suivantes :

- Q — une E/S de bloc est en attente
- G — obtenir une requête

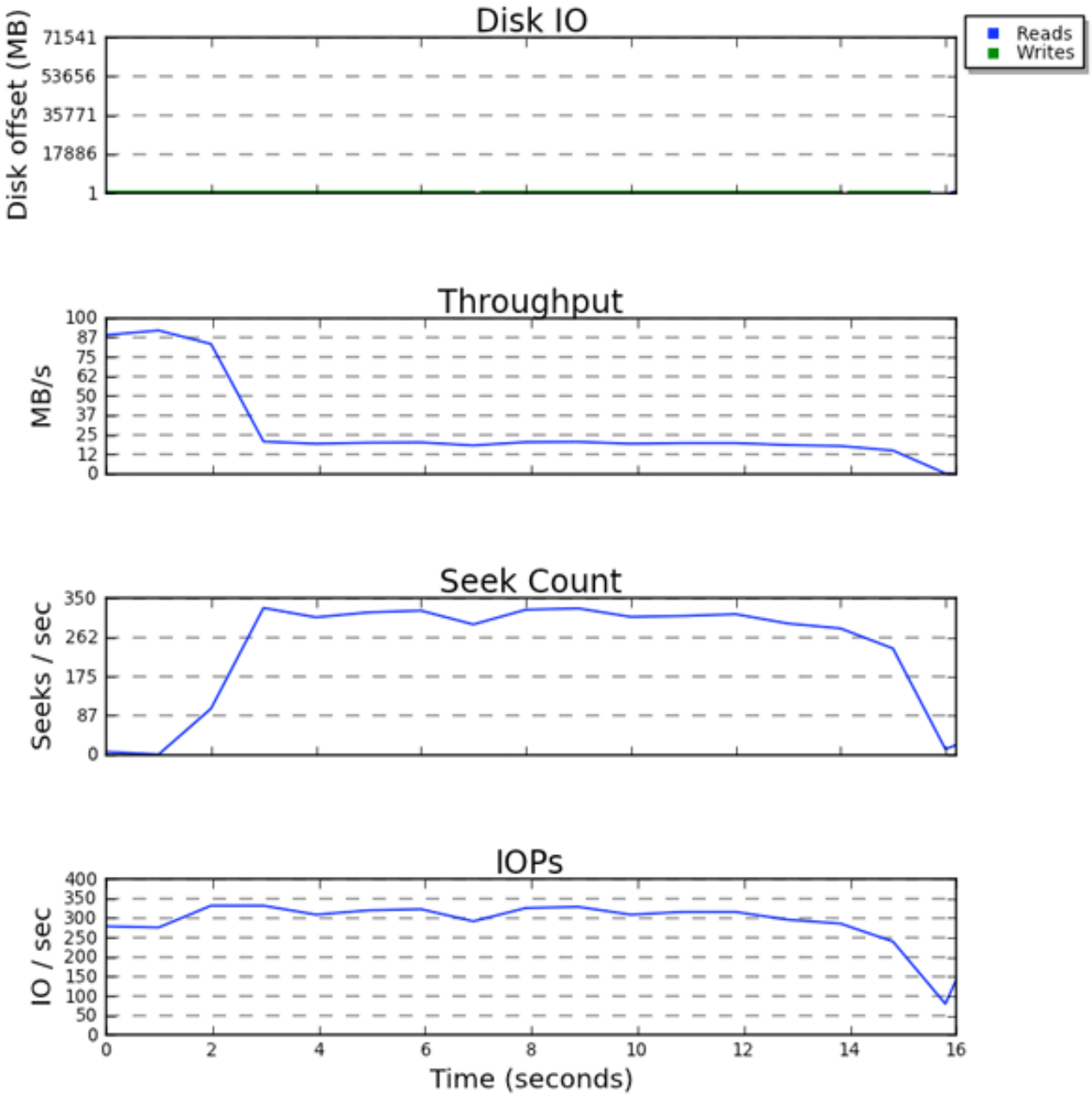
  Une nouvelle E/S de bloc en file d'attente n'était pas candidate pour une fusion avec toute requête existante, ainsi une requête pour une nouvelle couche de bloc est allouée.
- M — une E/S de bloc est fusionnée avec une requête existante.
- I — Une requête est insérée dans la file d'attente du périphérique.
- D — Une requête est délivrée au périphérique.
- C — Une requête est complétée par le pilote.
- P — La file d'attente du périphérique bloc est branchée (« Plugged ») afin d'autoriser l'agrégation des requêtes.
- U — La file d'attente du périphérique est débranchée (« Unplugged »), autorisant ainsi aux requêtes agrégées d'être délivrées au périphérique.

btt divise le temps pris par chacune de ces zones ainsi que le temps pris pour effectuer les transitions entre celles-ci, comme suit :

- Q2Q — temps écoulé entre les requêtes envoyées à la couche du bloc.
- Q2G — temps écoulé entre le moment où une E/S de bloc est mise en file d'attente et le moment où une requête y est allouée.
- G2I — temps écoulé entre le moment où une requête est allouée et le moment où elle est insérée à la file d'attente du périphérique.
- Q2M — temps écoulé entre le moment où une E/S de bloc est mise en file d'attente et le moment où elle est fusionnée avec une requête existante.
- I2D — temps écoulé entre le moment où une requête est insérée à la file d'attente du périphérique et le moment où elle est réellement délivrée au périphérique.
- M2D — temps écoulé à partir du moment où une E/S de bloc est fusionnée avec une requête sortante jusqu'au moment où la requête est délivrée au périphérique
- D2C — temps de service de la requête par le périphérique.
- Q2C — temps total écoulé dans la couche du bloc par une requête.

Vous pouvez déduire de nombreuses choses sur une charge de travail à partir du tableau ci-dessus. Par exemple, si Q2Q est de bien plus grande taille que Q2C, cela signifie que l'application ne délivre pas d'E/S en rapides successions. Ainsi, tout problème de performance n'est pas forcément lié au sous-système des E/S. Si D2C est très élevé, alors le périphérique prend trop de temps à répondre aux requêtes. Ceci peut indiquer que le périphérique est tout simplement surchargé (ce qui peut être dû au fait qu'il est une ressource partagée), ou que la charge de travail envoyée au périphérique est sous-optimale. Si Q2G est très élevé, cela signifie que de nombreuses requêtes sont en file d'attente en même temps. Ceci pourrait indiquer que le stockage n'est pas en mesure de supporter la charge des E/S.

Finalement, **seekwatcher** consomme les données binaires de **blktrace** et génère un ensemble de graphiques, y compris pour l'adresse LBA (« Logical Block Address »), le débit, les recherches par seconde et les IOPS (« I/O Per Second », ou E/S par seconde).



Avg Seeks/s	Avg MB/s	Avg IO/s	Run time (s)
252.57	31.72	305.99	16.21

Figure 6.2. Exemple de sortie de « seekwatcher »

Sur tous les graphiques, l'axe X représente le temps. Le graphique LBA affiche les lectures et écritures de différentes couleurs. Il est intéressant de remarquer la relation entre les graphes du débit et des recherches par seconde. Pour un stockage sensible aux opérations de recherches, il existe une relation inverse entre les deux graphiques. Par exemple, le graphe IOPS est utile si vous n'obtenez pas le débit auquel vous vous attendiez d'un périphérique, mais que vous atteigniez les limitations de ses IOPS.

6.4. CONFIGURATION

L'une des premières décisions à prendre est quel ordonnanceur d'E/S vous souhaitez utiliser. Cette section fournit un aperçu de chacun des principaux ordonnanceurs afin de vous aider à décider lequel sera le meilleur pour votre charge de travail.

6.4.1. CFQ (« Completely Fair Queuing »)

CFQ tente de fournir une certaine équité dans les décisions de programmation des E/S en se basant sur le processus qui a initié les E/S. Trois différentes classes de programmation sont fournies : RT (« Real-time », ou temps réel), BE (« Best-effort », ou meilleur effort) et « idle » (inactivité). Une classe de programmation peut être assignée manuellement à un processus avec la commande **ionice**, ou assignée de manière programmée via l'appel système **ioprio_set**. Par défaut, les processus sont placés dans la classe de programmation BE. Les classes de programmation RT et BE sont ensuite sous-divisées en huit priorités d'E/S dans chaque classe, la priorité 0 est la plus élevée et la priorité 7 est la plus basse. Les processus dans la classe RT sont programmés de manière bien plus agressive que les processus dans BE ou dans « idle », ainsi toute E/S RT programmée est effectuée avant les E/S BE ou « idle ». Cela signifie que les E/S de priorité RT peuvent dépasser les E/S BE et « idle ». La programmation BE est la classe de programmation par défaut et 4 est la priorité par défaut dans cette classe. Les processus dans la classe de programmation « idle » sont uniquement traités lorsqu'il ne reste aucune autre E/S en attente dans le système. Ainsi, il est très important de définir la classe de programmation des E/S d'un processus sur « idle » uniquement si les E/S du processus ne sont pas du tout requises pour progresser.

CFQ fournit une certaine équité en assignant une tranche de temps à chaque processus effectuant des E/S. Pendant sa tranche de temps, un processus peut avoir (par défaut) jusqu'à 8 requêtes en cours au même moment. L'ordonnanceur tente d'anticiper si une application va délivrer davantage d'E/S dans un futur proche, en se basant sur les données historiques. S'il est prévu qu'un processus délivrera davantage d'E/S, alors CFQ sera inactif, en attente de ces E/S, même si des E/S provenant d'autres processus sont en attente d'être délivrées.

À cause de la mise en veille effectuée par CFQ, cela est souvent inapproprié pour le matériel ne souffrant pas d'une grosse pénalité lors des opérations de recherche, tel que des matrices de stockage externes ou des disques SSD. Si l'utilisation de CFQ sur un tel stockage est une condition préalable (par exemple, sur vous souhaitez aussi utiliser l'ordonnanceur d'E/S de poids proportionnel de cgroup), vous devrez ajuster certains paramètres pour améliorer les performances de CFQ. Définissez les paramètres suivants dans les fichiers de même nom situés dans **/sys/block/*périphérique*/queue/iosched/** :

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

Lorsque ***group_idle*** est défini sur 1, il peut tout de même se produire des ralentissements d'E/S (ainsi le stockage principal n'est pas occupé à cause de la mise en veille). Cependant, ces ralentissements seront moins fréquents que les mises en veille sur chaque file d'attente dans le système.

CFQ est un ordonnanceur d'E/S ne conservant pas ses tâches, c'est-à-dire qu'il peut être inactif même lorsque des requêtes sont en attente (comme nous l'avons discuté ci-dessus). L'empilage des ordonnanceurs ne conservant pas leurs tâches peut présenter des latences de grande taille dans le chemin des E/S. Utiliser CFQ sur le dessus d'un contrôleur RAID matériel basé sur hôte est un exemple d'un tel empilage. Le contrôleur RAID peut implémenter son propre ordonnanceur ne conservant pas ses tâches, provoquant ainsi des délais sur deux niveaux dans la pile. Les ordonnanceurs de ne conservant pas leurs tâches opèrent au mieux lorsqu'ils possèdent un maximum de données sur lesquelles baser leurs décisions. Dans le cas où de tels algorithmes de programmation sont empilés, l'ordonnanceur le plus bas ne verra que ce que l'ordonnanceur du dessus ne lui a envoyé. Ainsi, la couche la plus basse verra un schéma d'E/S qui n'est pas du tout représentatif de la charge de travail réelle.

Réglages

back_seek_max

Les recherches en arrière sont habituellement mauvaise pour la performance car elles provoquent de plus grands délais dans le repositionnement des têtes que les recherches en avant. Cependant, CFQ les effectuera quand même si elles sont assez petites. Ce réglage contrôle la distance maximale en Ko que l'ordonnanceur autorise en recherche arrière. La valeur par défaut est **16** Ko.

back_seek_penalty

À cause de l'inefficacité des recherches en arrières, une pénalité est associée à chacune d'entre elles. La pénalité est un multiplicateur ; par exemple, prenez en considération une position de tête de disque sur 1024 Ko. Supposez qu'il y a deux requêtes dans la file d'attente, une sur 1008 Ko et une autre sur 1040 Ko. Les deux requêtes sont équidistantes par rapport à la position actuelle de la tête. Cependant, après avoir appliqué la pénalité de la recherche en arrière (par défaut : 2), la requête sur la position suivante du disque est maintenant deux fois plus proche que la requête précédente. Ainsi, la tête pourra avancer.

fifo_expire_async

Ce réglage contrôle combien de temps une requête async (écriture en mémoire tampon) peut rester sans être effectuée. Une fois le délai d'expiration (en millisecondes) dépassé, une seule requête async manquant de temps sera déplacée sur la liste d'expédition. La valeur par défaut est de **250** ms.

fifo_expire_sync

Ce réglage est le même que fifo_expire_async, mais pour les requêtes synchronisées (lectures et écritures O_DIRECT). La valeur par défaut est de **125** ms.

group_idle

Lorsqu'il est défini, CFQ deviendra inactif sur le dernier processus délivrant une E/S dans un cgroup. Il devrait être défini sur **1** lorsque des cgroups d'E/S de poids proportionnels sont utilisés et que ***slice_idle*** est défini sur **0** (ce qui est habituellement le cas sur les stockages rapides).

group_isolation

Si l'isolation de groupes est activée (définie sur **1**), elle fournit une meilleure isolation entre les groupes, au détriment du débit. Généralement, si l'isolation de groupes est désactivée, une certaine équité est offerte aux charges de travail séquentielles uniquement. L'activation de l'isolation de groupes offre une équité pour les charges de travail séquentielles et aléatoires. La valeur par défaut est **0** (désactivé). Veuillez consulter **Documentation/cgroups/blkio-controller.txt** pour obtenir davantage d'informations.

low_latency

Lorsque « low latency » (latence basse) est activé (défini sur **1**), CFQ tente de fournir un temps d'attente maximal de 300 ms pour chaque processus délivrant des E/S sur un périphérique. Ceci facilite l'établissement d'une certaine équité avec le débit. La désactivation de « low latency » (en paramétrant sur **0**) ignore la latence cible, permettant à chaque processus dans le système d'obtenir une tranche complète. « Low latency » est activé par défaut.

quantum

Quantum contrôle le nombre d'E/S que CFQ enverra au stockage à la fois, principalement limitant la profondeur de la file d'attente du périphérique. Par défaut, ceci est défini sur **8**. Le stockage peut

supporter des files d'attente bien plus profondes, mais l'augmentation de ***quantum*** aura aussi un impact négatif sur la latence, particulièrement en présence de charges de travail d'écriture séquentielles de grande taille.

slice_async

Ce réglage contrôle la tranche de temps allouée à chaque processus délivrant des E/S asynchrones (écritures en mémoire-tampon). Il est défini par défaut sur **40** ms.

slice_idle

Ceci spécifie combien de temps CFQ restera en veille en attente de requêtes supplémentaires. La valeur par défaut sur Red Hat Enterprise Linux 6.1 et ses versions précédentes est **8** ms. Sur Red Hat Enterprise Linux 6.2 et ses versions supérieures, la valeur par défaut est **0**. La valeur zéro améliore la débit d'un stockage RAID externe en supprimant tout ce qui est en veille aux niveaux de la file d'attente et de l'arborescence des services. Cependant, une valeur de zéro peut dégrader le débit sur un stockage non-RAID interne, car elle augmente le nombre général de recherches. Pour un stockage non-RAID, nous recommandons une valeur ***slice_idle*** plus élevée que 0.

slice_sync

Ce réglage dicte la tranche de temps allouée à un processus délivrant des E/S synchrones (lecture ou écriture directe). La valeur par défaut est de **100** ms.

6.4.2. Programmateur d'E/S « Deadline »

L'ordonnanceur d'E/S « Deadline » tente de fournir une latence garantie pour les requêtes. Il est important de remarquer que la mesure de la latence démarre uniquement lorsque les requêtes arrivent à l'ordonnanceur d'E/S (ceci est une distinction importante car une application peut être mise en veille alors qu'elle attendait que des descripteurs de requêtes soient libérés). Par défaut, les lectures sont prioritaires par rapport aux écritures, car les applications sont plus susceptibles de bloquer lors des E/S de lecture.

« Deadline » expédie les E/S en lots. Un lot est une séquence d'E/S de lecture ou d'écriture qui sont dans un ordre LBA croissant (élévateur unidirectionnel). Après le traitement de chaque lot, l'ordonnanceur d'E/S vérifie pour voir si des requêtes d'écriture ont été en attente trop longtemps, puis il décide s'il faut lancer un nouveau lot de lectures ou d'écritures. La liste des requêtes FIFO est uniquement vérifiée pour trouver des requêtes expirées au début de chaque lot, puis uniquement pour trouver la direction des données de ce lot. Ainsi. si un lot d'écritures est sélectionné et qu'il y a une requête de lecture expirée, cette requête de lecture ne sera pas traitée tant que le lot d'écritures ne sera pas terminé.

Réglages

fifo_batch

Ceci détermine le nombre de lectures ou écritures à délivrer en un seul lot. La valeur par défaut est de **16**. Définir une valeur plus élevée peut améliorer le débit, mais cela augmentera aussi la latence.

front_merges

Vous pouvez définir ce réglage sur **0** si vous savez que votre charge de travail ne générera jamais de fusions frontales. À moins d'avoir mesuré l'en-tête de cette vérification, il est conseillé de la laisser avec son paramètre par défaut (**1**).

read_expire

Ce réglage vous permet de définir le nombre de millisecondes pour qu'une requête de lecture soit traitée. Par défaut, ce nombre est défini sur **500** ms (une demi-seconde).

write_expire

Ce réglage vous permet de définir le nombre de millisecondes pour qu'une requête d'écriture soit traitée. Par défaut, ce nombre est défini sur **5000** ms (cinq secondes).

writes_starved

Ce réglage contrôle combien de lots de lecture peuvent être traités avant qu'un seul lot d'écriture ne puisse être traité. Plus cette valeur est élevée, plus la préférence sera donnée aux lectures.

6.4.3. Noop

L'ordonnanceur d'E/S « Noop » implémente un simple algorithme de programmation FIFO (« Premier arrivé, premier sorti », de l'anglais « First-in first-out »). La fusion des requêtes se produit sur la couche bloc générique, mais il s'agit simplement du dernier cache traité. Si un système est limité au processeur et que le stockage est rapide, cet ordonnanceur d'E/S pourrait être le meilleur à utiliser.

Ci-dessous figurent les réglages disponibles à la couche de bloc.

réglages /sys/block/sdX/queue

add_random

Dans certains cas, l'en-tête des événements d'E/S contribuant au pool d'entropie de **/dev/random** est mesurable. Dans certains cas, il peut être désirable de définir cette valeur sur 0.

max_sectors_kb

Par défaut, la taille de requête maximale envoyée sur disque est de **512** Ko. Ce réglage peut être utilisé pour élever ou abaisser cette valeur. La valeur minimale est limitée par la taille du bloc logique ; la valeur maximale est limitée par ***max_hw_sectors_kb***. Il existe des disques SSD qui fonctionnent moins bien lorsque la taille des E/S dépasse la taille des blocs de suppression internes. Dans ces cas, il est recommandé de régler ***max_hw_sectors_kb*** vers le bas afin de correspondre avec la taille du bloc de suppression. Vous pouvez effectuer des tests à l'aide d'un générateur d'E/S tel que **iozone** ou **aio-stress** ; par exemple, en variant la taille d'enregistrement de **512** octets à **1** Mo.

nomerges

Ce réglage est principalement une aide au débogage. La plupart des charges de travail bénéficient des fusions de requêtes (même sur des stockages plus rapides, comme les SSD). Cependant, dans certains cas, il est préférable de désactiver la fusion, comme lorsque vous souhaitez voir combien d'IOPS (E/S par seconde) un backend de stockage peut traiter sans désactiver la lecture à l'avance ou sans effectuer d'E/S aléatoires.

nr_requests

Chaque file d'attente de requêtes possède une limite du nombre total de descripteurs de requêtes pouvant être alloués à chaque E/S de lecture et d'écriture. Par défaut, ce nombre est **128**, ce qui signifie que 128 lectures et 128 écritures peuvent être mises en file d'attente à la fois avant de mettre un processus en veille. Le processus mis en veille sera le suivant à essayer d'allouer une requête et pas nécessairement le processus ayant alloué toutes les requêtes disponibles.

Si vous possédez une application sensible à la latence, alors vous devriez considérer les possibilités d'abaisser la valeur de ***nr_requests*** dans la file d'attente de votre requête et de limiter la profondeur de la file d'attente de commandes sur le stockage à un chiffre bas (aussi bas que **1** si vous le souhaitez), de manière à ce que les E/S de ré-écriture ne puissent pas allouer tous les descripteurs de requêtes disponibles et remplir la file d'attente du périphérique avec des E/S d'écriture. Une fois que les ***nr_requests*** ont été alloués, tous les autres processus tentant d'effectuer des E/S seront mis en veille en attendant que des requêtes deviennent disponibles. Cela rend le procédé plus équitable puisque les requêtes sont ensuite distribuées en tourniquet « round-robin » (plutôt que de laisser un seul processus toutes les consommer en de rapides successions). Remarquez qu'il s'agit uniquement d'un problème lors de l'utilisation des ordonnanceurs « Deadline » ou « Noop » car la configuration par défaut de CFQ sert de protection contre cette situation.

optimal_io_size

Sous certaines circonstances, le stockage sous-jacent rapportera une taille d'E/S optimale. Ceci est très commun dans les RAID logiciels et matériels, où la taille d'E/S optimale est la taille de bande. Si cette valeur est rapportée, les applications devraient délivrer des E/S correspondantes à la taille des E/S optimales et si possible en multiples de celles-ci.

read_ahead_kb

Le système d'exploitation peut détecter lorsqu'une application lit des données de manière séquentielle depuis un fichier ou un disque. Dans de tels cas, il procède à un algorithme de lecture à l'avance intelligent, dans lequel plus de données du disque que celles requises par l'utilisateur sont lues. Ainsi, lorsque l'utilisateur tentera à nouveau de lire un bloc de données, elles se trouveront déjà dans le cache de la page du système d'exploitation. L'inconvénient potentiel de ce procédé est que le système d'exploitation pourrait lire plus de données que nécessaire, ce qui occupe de l'espace dans le cache de la page jusqu'au vidage résultant de la forte pression de mémoire. Sous ces circonstances, avoir de multiples processus effectuant de fausses lectures à l'avance augmenterait la pression sur la mémoire.

Pour les périphériques mappeurs de périphériques, il est de bonne augure d'augmenter la valeur de ***read_ahead_kb*** à un nombre élevé, comme **8192** par exemple. La raison derrière ceci est qu'un périphérique mappant des péripphériques est souvent composé de multiples périphériques sous-jacents. Définir cette valeur sur son défaut (**128** Ko), multiplié par le nombre de périphériques que vous mappez est un bon début pour effectuer une optimisation.

rotational

Les disques durs traditionnels sont rotationnels (composés de plateaux effectuant des rotations). Les disques SSD ne le sont pas. La plupart des disques SSD publiciseront cela correctement. Cependant, si vous rencontriez un jour un périphérique n'expliquant pas clairement ce procédé, il pourrait se révéler nécessaire de définir « Rotational » sur **0** ; lorsque « Rotational » est désactivé, l'élévateur des E/S n'utilise pas de logique servant à réduire les recherches puisqu'il n'existe que de petites pénalités pour les opérations de recherche sur des média non-rotationnels.

rq_affinity

Les complétions des E/S peuvent être traitées sur différents CPU à partir de celui qui a délivré l'E/S. Définir ***rq_affinity*** sur **1** cause au noyau de délivrer des achèvements au CPU sur lequel l'E/S a été effectuée. Ceci peut améliorer l'efficacité de la mise en cache de données du CPU.

CHAPITRE 7. SYSTÈMES DE FICHIERS

Veuillez lire ce chapitre pour avoir un aperçu des systèmes de fichiers pris en charge pour une utilisation avec Red Hat Enterprise Linux et pour optimiser leur performance.

7.1. CONSIDÉRATIONS POUR OPTIMISER DES SYSTÈMES DE FICHIERS

Plusieurs considérations, communes à tous les systèmes de fichiers, sont à prendre en compte pour une meilleure optimisation : les options de formatage et de montage sélectionnées sur votre système et les actions disponibles aux applications qui peuvent améliorer leur performance sur un système donné.

7.1.1. Options de formatage

Taille des blocs des systèmes de fichiers

La taille de bloc peut être sélectionnée au moment de **mkfs**. La gamme de tailles valides dépend du système : la limite supérieure est la taille de page maximum du système hôte, tandis que la taille inférieure dépend du système de fichiers. La taille de bloc par défaut est appropriée dans la plupart des cas.

Si vous pensez devoir créer de nombreux fichiers plus petits que la taille de bloc par défaut, vous pouvez définir une taille de bloc plus petite afin de minimiser la quantité d'espace gaspillé par le disque. Remarquez cependant que paramétrer une taille de bloc plus petite peut limiter la taille maximum du système de fichiers et causer une surcharge de runtime supplémentaire, particulièrement pour les fichiers plus importants que la taille de bloc sélectionnée.

Géométrie des systèmes de fichiers

Si votre système utilise un stockage par bandes tel que RAID5, vous pouvez améliorer la performance en alignant les données et les métadonnées sur la géométrie sous-jacente du stockage lors de **mkfs**. Pour RAID logiciel (LVM ou MD) et le stockage de matériel d'entreprise, cette information est recherchée et définie automatiquement, mais dans de nombreux cas l'administrateur doit spécifier cette géométrie manuellement avec **mkfs** sur la ligne de commande.

Reportez-vous au *Guide d'administration du stockage* pour obtenir des informations supplémentaires sur la création et la maintenance de ces systèmes de fichiers.

Journaux externes

Les charges de travail intensives de métadonnées signifient que la section « log » (journal) d'un système de fichiers de journalisation (tel que ext4 et XFS) est extrêmement fréquemment mise à jour. Pour minimiser le temps de recherche du système de fichiers au journal, vous pouvez placer le journal sur le stockage dédié. Remarquez cependant que placer le journal sur un stockage externe plus lent que le système de fichiers principal peut annuler tout avantage potentiellement associé à l'utilisation d'un stockage externe.

AVERTISSEMENT

Assurez-vous que le journal externe soit fiable. La perte d'un périphérique journal externe provoquera la corruption du système de fichiers.

Les journaux externes sont créés lors de **mkfs** et les périphériques journaux sont spécifiés lors du montage. Reportez-vous aux pages man **mke2fs(8)**, **mkfs.xfs(8)** et **mount(8)** pour obtenir davantage d'informations.

7.1.2. Options de montage

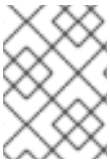
Barrières

Une barrière d'écriture est un mécanisme du noyau utilisé pour s'assurer que les métadonnées du système de fichiers soient écrites et ordonnées correctement sur un stockage persistant, même lorsque les périphériques de stockage avec des caches d'écriture volatiles subissent une perte de puissance. Les systèmes de fichiers avec des barrières d'écriture activées assurent aussi que toutes données transmises via **fsync()** persistent lors de pannes de courant. Red Hat Enterprise Linux active les barrières par défaut sur tout le matériel qui les prend en charge.

Cependant, l'activation des barrières d'écritures ralentit certaines applications de manière significative ; particulièrement les applications utilisant fortement **fsync()**, ou qui créent et suppriment de nombreux fichiers de petite taille. Pour les stockages sans cache d'écriture volatile, ou dans le rare cas où les inconsistances et pertes de données d'un système de fichiers après une panne de courant sont acceptables, les barrières peuvent être désactivées en utilisant l'option de montage ***nobarrier***. Pour obtenir davantage d'informations, veuillez vous reporter au *Guide d'administration du stockage*.

Temps d'accès (noatime)

Historiquement, lorsqu'un fichier est lu, le temps d'accès (***atime***) de ce fichier doit être mis à jour dans les métadonnées de l'inode, ce qui implique des écritures d'E/S supplémentaires. Si des métadonnées ***atime*** précises ne sont pas requises, montez le système de fichiers avec l'option ***noatime*** afin d'éliminer ces mises à jour de métadonnées. Cependant, dans la plupart des cas, ***atime*** n'est pas une grosse surcharge résultant du comportement « atime » relatif par défaut (ou ***relatime***) dans le noyau Red Hat Enterprise Linux 6. Le comportement ***relatime*** met uniquement à jour ***atime*** si le ***atime*** précédent est plus ancuen que le temps de modification (***mtime***) ou le temps de modification de statut (***ctime***).



NOTE

L'activation de l'option ***noatime*** active aussi le comportement ***nodiratime*** ; il n'est pas nécessaire de paramétrer ***noatime*** ainsi que ***nodiratime*** en même temps.

Prise en charge de la lecture à l'avance améliorée

La lecture à l'avance accélère l'accès aux fichiers en récupérant les données à l'avance et en les chargeant dans le cache de la page de manière à ce qu'elles soient disponibles plus tôt dans la mémoire au lieu de depuis le disque. Certaines charges de travail, comme celles impliquant une diffusion importante d'entrées et sorties séquentielles, bénéficient de hautes valeurs de lecture à l'avance.

L'outil **tuned** et l'utilisation de l'agrégation par bandes de LVM (« LVM striping ») augmentent la valeur de lecture à l'avance, mais ceci n'est pas toujours suffisant pour certaines charges de travail. En outre, Red Hat Enterprise Linux n'est pas toujours en mesure de définir une valeur de lecture à l'avance appropriée en se basant sur ce qui a été détecté de votre système de fichiers. Par exemple, si une matrice de stockage puissante se présente à Red Hat Enterprise Linux en tant que LUN unique puissant, le système d'exploitation ne la traitera pas en tant que matrice LUN puissante, et par défaut, ne fera pas plein usage des avantages de la lecture à l'avance potentiellement disponibles au stockage.

Utilisez la commande **blockdev** pour afficher et modifier la valeur de lecture à l'avance. Pour afficher la valeur de la lecture à l'avance actuelle pour un périphérique bloc particulier, vuillez exécuter :

-

```
# blockdev -getra périphérique
```

Pour modifier la valeur de lecture à l'avance pour ce périphérique bloc, veuillez exécuter la commande suivante. *N* représente le nombre de secteurs de 512 octets.

```
# blockdev -setra N périphérique
```

Remarquez que la valeur sélectionnée avec la commande **blockdev** ne persistera pas lors de redémarrages. Nous recommandons de créer un script **init.d** de niveau d'exécution pour définir cette valeur pendant le démarrage.

7.1.3. Maintenance de systèmes de fichiers

Abandonner les blocs inutilisés

Les opérations d'abandon en ligne et d'abandon du lot sont des fonctionnalités des systèmes de fichiers montés qui abandonnent les blocs qui ne sont pas utilisés par le système de fichiers. Les opérations sont utiles pour les disques SSD et les allocations fines et dynamiques de stockage.

Les *Opérations d'abandon par lot* sont exécutées de manière explicite par l'utilisateur avec la commande **fstrim**. Cette commande abandonne tous les blocs inutilisés dans un système de fichiers qui correspondent aux critères de l'utilisateur. Les deux types d'opérations sont pris en charge avec les systèmes de fichiers XFS et ext4 dans Red Hat Enterprise Linux 6.2 et ses versions supérieures tant que le périphérique bloc sous-jacent au système de fichiers prend en charge les opérations d'abandon physique. Les opérations d'abandon physique sont prises en charge si la valeur de **/sys/block/*périphérique*/queue/discard_max_bytes** n'est pas zéro.

Les *Opérations d'abandon en ligne* sont spécifiées lors du montage avec l'option **-o discard** (soit dans **/etc/fstab** ou en faisant partie de la commande **mount**) et exécutées en temps réeal sans intervention de la part de l'utilisateur. Les opérations d'abandon en ligne abandonnent uniquement les blocs passant de « Utilisé » à « Libre ». Les opérations d'abandon en ligne sont prises en charge sur les systèmes de fichiers ext4 dans Red Hat Enterprise Linux 6.2 et ses versions supérieures, ainsi que sur les systèmes de fichiers XFS dans Red Hat Enterprise Linux 6.4 et ses versions supérieures.

Red Hat recommande les opérations d'abandon par lot, à moins que la charge de travail du système ne soit telle que l'abandon par lot ne soit pas faisable, ou que des opérations d'abandon en ligne soient nécessaires pour effectuer la maintenance.

7.1.4. Considérations pour l'application

Pré-allocation

Les systèmes de fichiers ext4, XFS et GFS2 prennent en charge la pré-allocation d'espace efficace via l'appel glibc **fallocate(2)**. Dans les cas où les fichiers pouvaient être mal fragmentés à cause des schémas d'écriture, conduisant ainsi à de pauvres performances de lecture, la pré-allocation d'espace peut être une technique utile. La pré-allocation marque l'espace disque comme s'il a été alloué à un fichier, sans écrire de données dans cet espace. Jusqu'à ce que de véritables données soient écrites sur un bloc pré-alloué, les opérations de lecture retourneront des zéros.

7.2. PROFILS POUR LES PERFORMANCES DE SYSTÈMES DE FICHIERS

L'outil **tuned-adm** permet aux utilisateurs de facilement basculer entre différents profils qui ont été conçus pour améliorer les performances pour des cas d'utilisation spécifiques. Les profils particulièrement utiles pour améliorer les performances du stockage sont :

latency-performance

Profil de serveur pour optimisation typique des performances de la latence. Les mécanismes d'économies d'énergie de **tuned** et **ktune** sont désactivés. Le mode **cpuspeed** bascule sur **performance**. L'élévateur d'entrée et sorties est modifié sur **deadline** pour chaque périphérique. Le paramètre ***cpu_dma_latency*** est enregistré avec une valeur de **0** (la latence la plus basse possible) pour que la qualité de service de la gestion de l'alimentation limite la latence quand possible.

throughput-performance

Profil de serveur pour une optimisation typique des performances de débit. Ce profil est recommandé si le système ne possède pas de stockage de classe entreprise. Ce profil est le même que **latency-performance**, à l'exception de :

- ***kernel.sched_min_granularity_ns*** (granularité de préemption minimale de l'ordonnanceur), qui est paramétré que **10** millisecondes,
- ***kernel.sched_wakeup_granularity_ns*** (granularité de réveil de l'ordonnanceur), qui est paramétré sur **15** millisecondes,
- ***vm.dirty_ratio*** (ratio dirty de la machine virtuelle), qui est paramétré sur 40% et
- les huge pages transparentes sont activées.

enterprise-storage

Ce profil est recommandé pour les configurations de serveurs de taille entreprise avec un stockage de classe entreprise, incluant une protection de cache de contrôleur sur batterie et la gestion de caches sur disque. Ce profil est le même que **throughput-performance**, à l'exception de :

- La valeur ***readahead***, qui est paramétrée sur **4x** et
- Les systèmes de fichiers non root/boot sont montés à nouveau avec **barrier=0**.

Des informations supplémentaires sur **tuned-adm** sont disponibles sur la page man (**man tuned-adm**), ou dans le *Guide de gestion de l'alimentation*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

7.3. SYSTÈMES DE FICHIERS

7.3.1. Le système de fichiers Ext4

Le système de fichiers ext4 est une extension évolutive du système de fichiers par défaut ext3 disponible sur Red Hat Enterprise Linux 5. Ext4 est maintenant le système de fichiers par défaut de Red Hat Enterprise Linux 6 et prend en charge un système de fichiers d'une taille maximale de 16 To et un fichier unique d'une taille maximum de 16 To. La limite de 32000 sous-répertoires, présente dans ext3, est maintenant supprimée.

NOTE

Pour les systèmes de fichiers faisant plus de 16 To, nous recommandons l'utilisation d'un système de fichiers évolutif à haute capacité, tel que XFS. Pour obtenir davantage d'informations, voir la Section 7.3.2, « Le système de fichiers XFS ».

Les valeurs par défaut du système de fichiers ext4 sont optimales pour la plupart des charges de travail, mais si les analyses de performance indiquent que le comportement du système de fichiers a un impact sur la performance, plusieurs options de réglage sont disponibles :

Initialisation de la table des inodes

Pour de très grands systèmes de fichiers, le processus **mkfs.ext4** peut prendre très longtemps à initialiser toutes les tables d'inodes dans le système de fichiers. Ce processus peut être reporté avec l'option ***-E lazy_itable_init=1***. SI utilisé, les processus du noyau continueront à initialiser le système de fichiers après qu'il soit monté. La vitesse à laquelle cette initialisation se produit peut être contrôlée avec l'option ***-o init_itable=n*** de la commande **mount**, où le temps pris pour effectuer cette initialisation d'arrière-plan est d'environ 1/n. La valeur par défaut de **n** est **10**.

Comportement auto-fsync

Comme certaines applications n'effectuent pas correctement l'opération **fsync()** après avoir renommé un fichier existant ou après une troncature et ré-écriture, ext4 adopte par défaut la synchronisation automatique des fichiers après des opérations « remplacer en renommant » et « remplacer en tronquant ». Ce comportement est en grande partie conforme au comportement de l'ancien système de fichiers ext3. Cependant, comme les opérations **fsync()** peuvent prendre longtemps, si ce comportement n'est pas nécessaire, vous pouvez utiliser l'option ***-o noauto_da_alloc*** avec la commande **mount** pour le désactiver. Cela signifie que l'application doit explicitement utiliser **fsync()** afin d'assurer la persistance des données.

Priorité des E/S de journal

Par défaut, les E/S committées sur le journal ont une priorité légèrement plus élevée que les E/S normales. Cette priorité peut être contrôlée avec l'option ***journal_ioprio=n*** de la commande **mount**. La valeur par défaut est **3**. Les valeurs valides vont de 0 à 7, la valeur 0 étant l'E/S avec la plus haute priorité.

Pour les autres **mkfs** et options de réglage, veuillez voir les pages man **mkfs.ext4(8)** et **mount(8)**, ainsi que le fichier **Documentation/filesystems/ext4.txt** dans le paquetage kernel-doc.

7.3.2. Le système de fichiers XFS

XFS est un système de fichiers 64 bit journalisé à hôte unique robuste et hautement évolutif. Il est entièrement basé sur extensions et peut ainsi prendre en charge de très gros fichiers et systèmes de fichiers. Le nombre de fichiers qu'un système XFS peut contenir est uniquement limité par l'espace disponible dans le système de fichiers.

XFS prend en charge la journalisation de métadonnées, ce qui facilite une récupération après incident plus rapide. Le système de fichiers XFS peit aussi être défragmenté et élargi alors qu'il est monté et actif. En outre, Red Hat Enterprise Linux 6 prend en charge les utilitaires de sauvegarde et de restauration spécifiques à XFS.

XFS utilise une allocation basée sur extensions et offre un certain nombre de plans d'allocation, comme l'allocation retardée et la pré-allocation explicite. L'allocation basée sur extensions offre une méthode plus compacte et efficace pour surveiller l'espace dans un système de fichiers et améliore les performances sur fichiers de grande taille en réduisant la fragmentation et l'espace consommé par les

métadonnées. L'allocation retardée améliore les chances qu'un fichier soit écrit dans un groupe de blocs contigus, réduisant ainsi la fragmentation et améliorant les performances. La pré-allocation peut être utilisée pour totalement empêcher la fragmentation dans des cas où l'application sait combien de données devront être écrites à l'avance.

XFS offre une excellente évolutivité d'E/S en utilisant des arbres B pour indexer toutes les données et métadonnées des utilisateurs. Le nombre d'objets augmente en même temps que toutes les opérations des index héritent des caractéristiques d'évolutivité logarithmique des arbres B sous-jacents. Certaines des options de réglage que XFS fournit lors de l'opération **mkfs** font varier la largeur des arbres B, ce qui peut modifier les caractéristiques d'évolutivité des différents sous-systèmes.

7.3.2.1. Réglages de base pour XFS

En général, le format XFS et les options de montage par défaut sont optimaux pour la plupart des charges de travail ; Red Hat recommande que les valeurs par défaut soient utilisées, à moins qu'il soit prévu que des changements de configuration spécifiques soient bénéfiques aux charges de travail du système de fichiers. Si RAID logiciel est utilisé, la commande **mkfs.xfs** se configurera elle-même automatiquement avec la bonne unité de bande et la bonne largeur pour s'aligner sur le matériel. Ceci pourrait devoir être effectué manuellement si RAID logiciel est en cours d'utilisation.

L'option de montage **inode64** est fortement recommandée pour les systèmes de fichiers multi-téraoctets, sauf lorsque le système de fichiers est exporté via NFS et que les clients NFS 32 bits hérités requièrent accès au système de fichiers.

L'option de montage **logbsize** est recommandée pour les systèmes de fichiers qui sont fréquemment modifiés, ou qui sont modifiés en rafales. La valeur par défaut est **MAX** (32 Ko, unité de bande journalisée) et la taille maximum est 256 Ko. Une valeur de 256 Ko est recommandée pour les systèmes de fichiers subissant de lourdes modifications.

7.3.2.2. Réglages avancés pour XFS

Avant de modifier les paramètres XFS, vous devez comprendre pourquoi les paramètres XFS par défaut causent des problèmes de performance. Cela implique de comprendre ce que fait votre application et de quelle manière le système de fichiers réagit à ces opérations.

Les problèmes de performance pouvant être observés qui peuvent être corrigés ou réduits par le biais de réglages sont généralement causés par la fragmentation du fichier ou par des conflits de ressources dans le système de fichiers. Il existe plusieurs manières de répondre à ces problèmes et dans certains cas, la correction du problème requerra que ce soit l'application, plutôt que la configuration du système de fichiers, qui soit modifiée.

Si vous n'avez pas effectué ce processus précédemment, il est recommandé de vous renseigner auprès de votre ingénieur d'assistance Red Hat local.

Optimisation pour un grand nombre de fichiers

XFS impose une limite arbitraire sur le nombre de fichiers qu'un système de fichiers peut contenir. En général, cette limite est suffisamment élevée pour ne jamais être atteinte. Si vous savez à l'avance que la limite par défaut sera insuffisante, vous pouvez augmenter le pourcentage d'espace du système de fichiers autorisé aux inodes avec la commande **mkfs.xfs**. Si vous atteignez la limite de fichiers après la création du système de fichiers (habituellement indiqué par des erreurs ENOSPC lors d'une tentative de création de fichier ou de répertoire alors qu'il existe de l'espace disponible), vous pouvez ajuster la limite avec la commande **xfs_growfs**.

Optimisation pour un grand nombre de fichiers dans un seul répertoire

La taille de bloc d'un répertoire est fixée pour la durée de vie d'un système de fichiers et ne peut pas être

modifiée, sauf lors du formatage initial avec **mkfs**. Le bloc de répertoire minimum est la taille de bloc du système de fichiers, qui est par défaut **MAX** (4 Ko, taille de bloc du système de fichiers). En générale, il n'y a pas de raison de réduire la taille de bloc du répertoire.

Comme la structure de répertoire est basée sur un arbre B, modifier la taille de bloc affecte la quantité d'informations sur le répertoire qui peuvent être récupérées ou modifiées par entrée ou sortie (E/S) physique. Plus la taille d'un répertoire augmente, plus les entrées et sorties requises par chaque opération sur une taille de bloc donnée seront nombreuses.

Cependant, lorsque des tailles de blocs plus importantes sont en cours d'utilisation, plus de CPU sont consommés par chaque opération de modification comparé à la même opération sur un système de fichiers avec une plus petite taille de bloc de répertoire. Cela signifie que pour les répertoires de plus petite taille, des blocs de répertoire de grande taille résulteront en de plus mauvaises performances de modification. Lorsque le répertoire atteint une taille avec laquelle les entrées et sorties (E/S) constituent le facteur limitant la performance, les blocs de répertoire de plus grande taille fonctionneront mieux.

La configuration par défaut d'un bloc de système de fichiers d'une taille de 4 Ko et d'un bloc de répertoire d'une taille de 4 Ko est meilleure pour les répertoires avec 1 à 2 millions d'entrées avec une longueur de nom de 20 à 40 octets par entrée. Si votre système de fichiers requiert davantage d'entrées, les blocs de répertoire de plus grande taille auront tendance à mieux fonctionner - une taille de bloc de 16 Ko est plus appropriée pour des systèmes de fichiers avec 1 à 10 millions d'entrées et une taille de bloc de 64 Ko sera plus appropriée pour des systèmes de fichiers avec plus de 10 millions d'entrées de répertoires.

Si la charge de travail utilise plus de recherches aléatoires de répertoires que de modifications (c'est-à-dire si les lectures de répertoires sont bien plus communes ou importantes que les écritures de répertoires), alors les limites ci-dessus pour augmenter la taille des blocs sont d'environ un ordre de magnitude plus faible.

Optimisation pour la simultanéité

Contrairement aux autres systèmes de fichiers, XFS peut effectuer de nombreux types d'opérations d'allocation et de dé-allocation simultanément, à condition que les opérations se produisent sur des objets non-partagés. Les allocations et dé-allocations d'extensions peuvent se produire en même temps à condition que les opérations simultanées se produisent dans différents groupes d'allocation. De manière similaire, les allocations ou dé-allocations d'inodes peuvent se produire en même temps à condition que les opérations simultanées affectent différents groupes d'allocation.

Le nombre de groupes d'allocation devient important lorsque des machines avec un compte de CPU élevé et des applications à multiples threads qui tentent d'effectuer des opérations simultanément sont utilisées. Si seuls quatre groupes d'allocation existent, alors des opérations de métadonnées prolongées et parallèles n'évolueront pas plus que ces quatre CPU (qui est la limite de simultanéité fournie par le système). Pour de petits systèmes de fichiers, assurez-vous que le nombre de groupes d'allocation est pris en charge par la simultanéité fournie par le système. Pour des systèmes de fichiers de grande taille (plusieurs dizaines de téraoctets et au-delà), les options de formatage par défaut créent généralement suffisamment de groupes d'allocation pour éviter de limiter la simultanéité.

Les applications doivent être conscientes des points de contention uniques afin d'utiliser le parallélisme inhérent à la structure du système de fichiers XFS. Il n'est pas possible de modifier un répertoire simultanément. Ainsi, des applications créant et supprimant de nombreux fichiers éviteront de stocker tous les fichiers dans un répertoire unique. Chaque répertoire créé est placé dans un groupe d'allocation différent. Ainsi, des techniques telles que le hachage de fichiers sur de multiples sous-répertoires fournissent un schéma de stockage plus modulable comparé à l'utilisation d'un seul répertoire de grande taille.

Optimisation pour des applications utilisant des attributs étendus

XFS peut stocker des attributs de petite taille directement dans l'inode si l'espace est disponible dans l'inode. Si l'attribut peut être intégré dans l'inode, alors il peut être récupéré et modifié sans devoir faire

recours à des E/S supplémentaires pour récupérer des blocs d'attributs séparés. Le différentiel de performance entre les attributs « en ligne » et « hors-ligne » peut aisément être d'un ordre de magnitude plus lent pour les attributs « hors-ligne ».

Pour la taille d'inode par défaut de 256 octets, environ 100 octets de l'espace attribué est disponible selon le nombre de pointeurs d'extensions de données aussi stockés dans l'inode. La taille d'inode par défaut n'est vraiment utile que pour le stockage de quelques petits attributs.

Augmenter la taille d'inode lors de la commande mkfs peut augmenter la quantité d'espace disponible pour le stockage des attributs en ligne. Un inode d'une taille de 512 octets augmente l'espace disponible pour les attributs jusqu'à environ 350 octets ; un inode de 2 Ko possède environ 1900 octets d'espace disponible.

Il existe cependant une limite à la taille des attributs individuels pouvant être stockés en ligne. La limite de taille maximale est de 254 octets pour le nom de l'attribut et la valeur (c'est-à-dire un attribut avec une longueur de nom de 254 octets et une longueur de valeur de 254 octets restera en ligne). Le dépassement de ces limites force les attributs hors-ligne, même s'il y avait suffisamment d'espace pour stocker tous les attributs dans l'inode.

Optimisation pour les modifications de métadonnées prolongées

La taille du journal est le facteur principal déterminant le niveau réalisable de modifications prolongées de métadonnées. Le périphérique journal est circulaire et avant que les données de la queue puisse être écrasées, toutes les modifications dans le journal doivent être écrites sur les emplacements réels sur le disque. Cela implique un effort significatif de recherche pour réécrire toutes les métadonnées modifiées.

Un périphérique journal de petite taille résultera en de très fréquentes réécritures de métadonnées. Le journal poussera constamment sa queue pour libérer de l'espace et les métadonnées fréquemment modifiées seront fréquemment écrites sur disque, ralentissant ainsi les opérations.

L'augmentation de la taille du journal augment la période de temps entre les événements push sur la queue. Cela permet une meilleure agrégation des métadonnées modifiées, résultant en des meilleurs schémas de réécriture de métadonnées et des réécritures de métadonnées fréquemment modifiées moindres. Le compromis est que des journaux de plus grande taille nécessiteront plus de mémoire pour surveiller toutes les modifications arriérées en mémoire.

Si vous possédez une machine avec une mémoire limitée, alors des journaux de grande taille ne seront pas bénéfiques puisque les contraintes de mémoire entraîneront des réécritures de métadonnées longtemps avant que les bénéfices d'un journal de grande taille puisse être réalisé. Dans ces cas, des journaux de plus petite taille fourniront plus souvent une meilleure performance car des réécritures de métadonnées du journal manquant d'espace sont plus efficaces que les réécritures entraînées par la réclamation de mémoire.

Vous deviez toujours tenter d'aligner le journal sur l'unité en bande sous-jacente du périphérique qui contient le système de fichiers. **`mkfs`** effectue cela par défaut pour les périphériques MD et DM, mais pour le RAID logiciel, il se peut qu'il faille le spécifier. Paramétrer ceci correctement prévient toute possibilité d'E/S de journal provoquant des E/S non-alignées et des opérations de lecture-modification-écriture conséquentes lors de l'écriture de modifications sur disque.

Les opérations du journal peuvent être améliorées en modifiant les options de montage. L'augmentation de la taille des mémoires tampon de journaux situés dans la mémoire (**`logbsize`**) augmente la vitesse à laquelle les modifications peuvent être écrites sur le journal. La taille de la mémoire tampon de journal par défaut est **`MAX`** (32 Ko, unité de bande de journal) et la taille maximale est de 256 Ko. En général, une valeur plus importante se traduit par une performance plus rapide. Cependant, sous de lourdes charges de travail fsync, des mémoires tampon de journaux de petite taille sont visiblement plus rapides que des mémoires tampon de grande taille avec un alignement d'unité de bande de grande taille.

L'option de montage **delaylog** améliore aussi les performances des modifications prolongées de métadonnées en réduisant le nombre de modifications apportées au journal. Ceci est réalisé en agrégeant des modifications individuelles en mémoire avant de les écrire sur le journal : les métadonnées fréquemment modifiées sont écrites sur le journal de manière périodique plutôt qu'à chaque fois qu'une modification est effectuée. Cette option augmente l'utilisation par la mémoire du suivi des métadonnées modifiées et augmente le nombre d'opérations potentiellement perdues lorsqu'un incident se produit, mais elle peut améliorer la vitesse et l'évolutivité des modifications de métadonnées par un ordre de magnitude ou plus. L'utilisation de cette option ne réduit pas l'intégrité des données ou métadonnées lorsque **fsync**, **fdatasync** ou **sync** sont utilisés pour s'assurer que les données ou métadonnées soient bien écrites sur le disque.

7.4. CLUSTERING

Le stockage en cluster (ou en grappe) fournit une image de fichier consistante à travers tous les serveurs dans un cluster, permettant aux serveurs de lire e écrire sur un seul système de fichiers partagé. Ceci simplifie l'administration du stockage en limitant les tâches comme l'installation et la correction des applications à un seul système de fichiers. Un système de fichiers couvrant la totalité du cluster élimine aussi le besoin de copies redondantes des données d'application, simplifiant ainsi la sauvegarde et la récupération après sinistre.

Le module complémentaire Red Hat High Availability offre un stockage en cluster en conjonction avec Red Hat Global File System 2 (faisant partie du module complémentaire Resilient Storage)

7.4.1. Global File System 2

GFS2 (Global File System 2) est un système de fichiers natif qui interagit directement avec le système de fichiers du noyau Linux. Il permet à de multiples ordinateurs (ou nœuds) de simultanément partager le même périphérique de stockage dans un cluster. Le système de fichiers GFS2 effectue ses réglages automatiquement, mais un ajustement manuel est possible. Cette section décrit les considérations de performance à prendre en compte lors d'une tentative manuelle de réglage des performances.

Red Hat Enterprise Linux 4 présente des améliorations quant à la gestion de la fragmentation des fichiers dans GFS2. Les fichiers crées par Red Hat Enterprise Linux 6.3 ou ses versions précédentes étaient plus prônes à la fragmentation de fichiers si de multiples fichiers étaient écrits au même moment par plus d'un processus. Cette fragmentation ralentissait les opérations, particulièrement sur les charges de travail impliquant des fichiers de grande taille. Avec Red Hat Enterprise Linux 6.4, les écritures simultanées se traduisent par moins de fragmentations de fichiers et donc par de meilleures performances pour ces charges de travail.

Même s'il n'y a pas d'outil de défragmentation pour GFS2 sur Red Hat Enterprise Linux, vous pouvez défragmenter des fichiers individuels en les identifiant à l'aide de l'outil **filefrag**, puis en les copiant vers les fichiers temporaires et en renommant les fichiers temporaires pour remplacer les originaux. (Cette procédure peut aussi être effectuée dans les versions précédant 6.4 tant que l'écriture est réalisée de manière séquentielle.)

Puisque GFS2 utilise un mécanisme de verrouillage global qui nécessite potentiellement des communications entre les nœuds d'un cluster, les meilleures performances seront réalisées lorsque le système est conçu pour éviter les conflits de fichiers et répertoires entre ces nœuds. Certaines méthodes pour éviter des conflits peuvent être utilisées :

- Pré-allouez des fichiers et répertoires avec **fallocate** quand possible afin d'optimiser le processus d'allocation et d'éviter de devoir verrouiller les pages sources.
- Minimisez les zones du système de fichiers qui sont partagées entre de multiples nœuds pour minimiser les invalidations de cache à travers plusieurs nœuds et améliorer les performances.

Par exemple, si de multiples nœuds montent le même système de fichiers, mais accèdent à différents sous-répertoires, vous obtiendrez probablement de meilleures performances en déplaçant un sous-répertoire sur un autre système de fichiers.

- Sélectionnez une taille et un nombre de groupes de ressource optimaux. Ceci dépend des tailles de fichiers typiques et de l'espace libre disponible sur le système et affecte la probabilité que de multiples nœuds tenteront d'utiliser un groupe de ressources simultanément. Trop de groupes de ressources peut ralentir l'allocation des blocs pendant que l'espace alloué est localisé, tandis que trop peu de groupes de ressources peut provoquer un conflit de verrouillage pendant la dé-allocation. Il est généralement recommandé de tester de multiples configurations afin de déterminer laquelle conviendra le mieux à charge de travail.

Cependant, les contentions ne sont pas les seuls problèmes pouvant affecter les performances du système de fichiers GFS2. D'autres meilleures pratiques pour améliorer les performances générales existent aussi :

- Sélectionnez votre matériel de stockage en fonction des schémas d'E/S auxquels vous vous attendez de la part des nœuds des clusters et en fonction des conditions préalables de performance de votre système de fichiers.

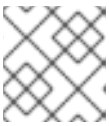
- Utilisez le stockage SSD quand possible afin de diminuer le temps de recherche.

- Créez un système de fichiers de taille appropriée pour votre charge de travail, assurez-vous aussi que le système de fichiers ne soit jamais à plus de 80% de sa capacité. Les systèmes de fichiers plus petits mettront un temps proportionnellement plus court pour effectuer des copies de sauvegarde et nécessiteront moins de temps et de mémoire pour les vérifications du système de fichiers, mais ils seront sujets à une fragmentation plus importante s'ils sont trop petits pour leur charge de travail.

- Définissez des tailles de journaux plus importantes pour les charges de travail de métadonnées intensives ou lorsque des données journalisées sont en cours d'utilisation. Même si cela utilise plus de mémoire, les performances en sont améliorées car davantage d'espace de journalisation est disponible pour stocker des données avant qu'une écriture ne soit nécessaire.

- Assurez-vous que les horloges des nœuds GFS2 sont synchronisées pour éviter les problèmes avec des applications mises en réseau. Nous recommandons d'utiliser NTP (« Network Time Protocol »).

- À moins que les heures d'accès aux fichiers ou répertoires ne soient critiques à l'opération de votre application, montez le système de fichiers avec les options de montage **`noatime`** et **`nodiratime`**.



NOTE

Red Hat recommande fortement d'utiliser l'option **`noatime`** avec GFS2.

- Si vous devez utiliser des quotas, essayez de réduire la fréquence des transactions de synchronisation des quotas ou utilisez la synchronisation de quotas approximative afin de prévenir l'apparition de problèmes de performance résultant des mises à jour constantes de fichiers de quotas.

NOTE

La comptabilité des quotas approximatifs peut permettre aux utilisateurs et aux groupes de légèrement excéder leurs limites de quotas. Pour minimiser le problème, GFS2 réduit dynamiquement les périodes de synchronisation lorsqu'un utilisateur ou un groupe approche de son quota limite.

Pour obtenir des informations détaillées sur chaque aspect des réglages de performances GFS2, veuillez consulter le guide *Global File System 2*, disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

CHAPITRE 8. « NETWORKING » (RÉSEAU)

Au cours du temps, la pile réseau de Red Hat Enterprise Linux a été mise à niveau avec de nombreuses fonctionnalités automatisées. Pour la plupart des charges de travail, les paramètres réseau automatiquement configurés offrent des performances optimisées.

Dans la plupart des cas, les problèmes de performance réseau sont en fait causés par un malfonctionnement du matériel ou par une infrastructure défectueuse. Ce type de causes sont au-delà de l'étendue des explications de ce document, les problèmes de performance et les solutions dont il est question dans ce chapitre servent à parfaitement optimiser des systèmes fonctionnels.

Le « Networking » est un sous-système délicat, contenant différentes parties avec des connexions sensibles. Ce qui explique pourquoi la communauté Open Source et Red Hat investissent tant de travail dans l'implémentation de diverses manières d'optimiser les performances réseau automatiquement.

8.1. AMÉLIORATIONS DES PERFORMANCES RÉSEAU

Ci-dessous figurent les améliorations des performances réseau apportées à Red Hat Enterprise Linux 6.1 :

RPS (« Receive Packet Steering »)

RPS active une seule file d'attente **rx** NIC pour que sa charge de travail **softirq** de réception soit distribuée sur plusieurs CPU. Ceci aide à empêcher que le trafic réseau ne soit congestionné sur une seule file d'attente de matériel NIC.

Pour activer RPS, veuillez spécifier les noms des CPU cibles dans **/sys/class/net/*ethX*/queues/*rx-N*/rps_cpus**, remplaçant ***ethX*** par le nom du périphérique correspondant à la NIC (par exemple, **eth1**, **eth2**) et ***rx-N*** par la file de réception NIC spécifiée. Ceci permettra aux CPU spécifiés dans le fichier de traiter les données de la file d'attente ***rx-N*** sur ***ethX***. Lorsque vous spécifiez les CPU, prenez en considération les *affinités du cache* [4]de la file d'attente.

RFS (« Receive Flow Steering »)

RFS est une extension de RPS, permettant à l'administrateur de configurer une table de hachage qui est automatiquement remplie lorsque les applications reçoivent des données et sont interrogées par la pile réseau. Ceci permet de déterminer quelles applications reçoivent quelles données réseau (en se basant sur des informations réseau source:destination).

À l'aide de ces informations, la pile réseau peut programmer le CPU optimal pour qu'il reçoive chaque paquet. Pour configurer RFS, veuillez utiliser les réglages suivants :

/proc/sys/net/core/rps_sock_flow_entries

Ce réglage contrôle le nombre maximal de sockets ou de flux que le noyau peut diriger vers n'importe quel CPU spécifié. La limite définie est globale et partagée.

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

Ce réglage contrôle le nombre maximal de sockets ou flux que le noyau peut diriger vers une file de réception spécifiée (***rx-N***) située sur une NIC (***ethX***). Remarquez que la somme de toutes les valeurs par file de ce réglage sur toutes les NIC devrait être égale ou inférieure à **/proc/sys/net/core/rps_sock_flow_entries**.

Contrairement à RPS, RFS permet à la file de réception et à l'application de partager le même CPU lors du traitement des flux de paquets. Dans certains cas, cela se traduit par de meilleures performances. Cependant, ces améliorations dépendent de facteurs tels que la hiérarchie des caches, la charge de l'application, et ainsi de suite.

Prise en charge de getsockopt pour les flux de type « thin-stream » TCP

Thin-stream est un terme utilisé pour caractériser les protocoles de transport avec lesquels les applications envoient des données à des débits si bas que les mécanismes de retransmission du protocole ne sont pas complètement saturés. Les applications qui utilisent des protocoles « thin-stream » effectuent habituellement leur transport via des protocoles fiables, comme TCP ; dans la plupart des cas, ce type d'applications offre des services urgents (comme des opérations en bourse, jeux en ligne et systèmes de contrôle).

Sur les services urgents, la perte de paquets peut se révéler dévastatrice pour la qualité du service. Pour vous aider à éviter cela, l'appel **`getsockopt`** a été amélioré afin de prendre en charge deux options supplémentaires :

TCP_THIN_DUPACK

Ce booléen active le déclenchement dynamique des retransmissions après un « dupACK » pour les « thin-streams ».

TCP_THIN_LINEAR_TIMEOUTS

Ce booléen active le déclenchement dynamique des délais d'expirations linéaires pour les « thin-streams ».

Ces deux options sont spécifiquement activées par l'application. Pour obtenir des informations supplémentaires sur ces options, veuillez consulter **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/ip-sysctl.txt`**. Pour obtenir des informations supplémentaires sur les « thin-streams », veuillez consulter **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/tcp-thin.txt`**.

Prise en charge des TProxy (proxys transparents)

Le noyau peut maintenant gérer des sockets UDP et TCP IPv4 non-localement liés pour prendre en charge des proxys transparents. Pour activer cela, vous devrez configurer iptables en conséquence. Vous devrez aussi activer et configurer la stratégie de routage correctement.

Pour obtenir des informations supplémentaires sur les proxys transparents, veuillez consulter **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/tproxy.txt`**.

8.2. PARAMÈTRES RÉSEAU OPTIMISÉS

L'optimisation des performances est habituellement effectuée de manière préventive. Le plus souvent, nous ajustons les variables connues avant d'exécuter une application ou de déployer un système. Si le réglage se révèle inefficace, nous tentons d'ajuster d'autres variables. La logique derrière ce raisonnement est que *par défaut*, le système n'opère pas à son niveau de performance optimale. Ainsi, nous *pensons* devoir régler le système en conséquence. Dans certains cas, nous le faisons en calculant les risques.

Comme mentionné plus tôt, la pile réseau s'optimise principalement elle-même. En outre, l'optimisation du réseau requiert une complète compréhension non-seulement de la manière dont la pile réseau fonctionne, mais aussi des conditions spécifiques requises des ressources réseau du système

spécifique. Une configuration incorrecte des performances peut mener à une performance dégradée.

Par exemple, prenez en considération le *problème bufferfloat*. Augmenter les profondeurs des files d'attente de la mémoire-tampon se traduit par des connexions ayant des fenêtres de congestion plus grandes que le lien aurait normalement autorisé (dû à la mémoire-tampon profonde). Cependant, ces connexions ont aussi d'énormes valeurs RTT puisque les trames passent tant de temps en file d'attente. Ceci provoquera en retour une sortie sous-optimale car il sera impossible de détecter la congestion.

Lorsqu'il s'agit des performances réseau, il est recommandé de conserver les paramètres par défaut, *à moins* qu'un problème de performance particulier ne devienne apparent. De tels problèmes incluent les pertes de trames, un débit réduit de manière significative, et ainsi de suite. Même ainsi, la meilleure solution sera souvent celle résultant d'une étude méticuleuse du problème, plutôt que de simplement augmenter les paramètres de réglage (augmenter les longueurs des files d'attente et de la mémoire-tampon, réduire la latence des interruptions, etc).

Pour correctement diagnostiquer un problème de performance, veuillez utiliser les outils suivants :

netstat

Utilitaire sur la ligne de commande qui imprime des connexions réseau, tables de routage, statistiques d'interface, connexions fictives et adhésions à des multidiffusions. Il récupère des informations sur le sous-système « networking » à partir du système de fichiers **`/proc/net/`**. Ces fichiers incluent :

- **`/proc/net/dev`** (informations sur le périphérique)
- **`/proc/net/tcp`** (informations sur le socket TCP)
- **`/proc/net/unix`** (informations sur le socket du domaine Unix)

Pour obtenir des informations supplémentaires sur **`netstat`** et ses fichiers référencés depuis **`/proc/net/`**, veuillez consulter la page man **`netstat`** : **`man netstat`**.

dropwatch

Utilitaire de surveillance qui suit les paquets laissés par le noyau. Pour obtenir davantage d'informations, veuillez consulter la page man **`dropwatch`** : **`man dropwatch`**.

ip

Utilitaire pour gérer et surveiller les itinéraires, les périphériques, les routages basés sur stratégie et les tunnels. Pour obtenir davantage d'informations, veuillez consulter la page man **`ip`** : **`man ip`**.

ethtool

Utilitaire pour afficher et modifier les paramètres de NIC. Pour obtenir davantage d'informations, veuillez consulter la page man **`ethtool`** : **`man ethtool`**.

/proc/net/snmp

Fichier affichant des données ASCII nécessaires pour les bases d'informations de gestion IP, ICMP, TCP et UDP pour un agent **`snmp`**. Il affiche aussi des statistiques UDP allégées en temps réel.

Le *Guide du débutant SystemTap* contient plusieurs exemples de script que vous pouvez utiliser pour profiler et surveiller les performances réseau. Ce guide est disponible sur http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Après avoir collecté les données pertinentes à un problème de performances réseau, vous devriez être en mesure de formuler une théorie, et probablement une solution.[5] Par exemple, une augmentation des erreurs d'entrées UDP dans **/proc/net/snmp** indique qu'une ou plusieurs des files de réception des sockets sont pleines lorsque la pile réseau tente de mettre des nouvelles trames dans la file d'attente d'un socket d'application.

Ceci indique que les paquets sont congestionnés sur *au moins* une file d'attente de socket, ce qui signifie que soit la file vide les paquets trop lentement, soir le volume de paquets est trop important pour cette file. S'il s'agit de la seconde possibilité, alors veuillez rechercher des données perdues dans les journaux de toute application utilisant le réseau de manière intensive. Pour résoudre ceci, vous devrez optimiser ou reconfigurer l'application fautive.

Taille du tampon de réception du socket

Les tailles des sockets d'envoi et de réception sont ajustées dynamiquement. Ainsi, elles doivent rarement être modifiées manuellement. Si des analyses supplémentaires, comme celle présentée dans l'exemple du réseau SystemTap, **sk_stream_wait_memory.stp**, suggèrent que le taux de vidage de la file du socket est trop lent, alors vous pouvez augmenter la profondeur de la file du socket de l'application. Pour ce faire, augmentez la taille des tampons utilisés par les sockets en configurant l'une des valeurs suivantes :

rmem_default

Paramètre du noyau qui contrôle la taille *par défaut* des tampons de réception utilisés par les sockets. Pour le configurer, veuillez exécuter la commande suivante :

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

Remplacez ***N*** par la taille de tampon souhaitée, en octets. Pour déterminer la valeur de ce paramètre du noyau, affichez **/proc/sys/net/core/rmem_default**. N'oubliez pas que la valeur de **rmem_default** ne doit pas être supérieure à **rmem_max** (**/proc/sys/net/core/rmem_max**) ; si besoin est, augmentez la valeur de **rmem_max**.

SO_RCVBUF

Option de socket contrôlant la taille *maximale* d'un tampon de réception d'un socket, en octets. Pour obtenir davantage d'informations sur **SO_RCVBUF**, veuillez consulter la page man : **man 7 socket**.

Pour configurer **SO_RCVBUF**, veuillez utiliser l'utilitaire **setsockopt**. Vous pouvez récupérer la valeur **SO_RCVBUF** actuelle avec **getsockopt**. Pour obtenir davantage d'informations sur l'utilisation de ces deyx utilitaires, veuillez consulter la page man **setsockopt** : **man setsockopt**.

8.3. APERÇU DES RÉCEPTIONS DE PAQUETS

Pour mieux analyser les congestions réseau et problèmes de performance, il faut comprendre comment la réception de paquets fonctionne. La réception de paquets est importante dans les réglages des performances réseau car le chemin de réception est souvent l'endroit où les trames sont perdues. Les trames perdues dans le chemin de réception peuvent causer un handicap significatif aux performances réseau.

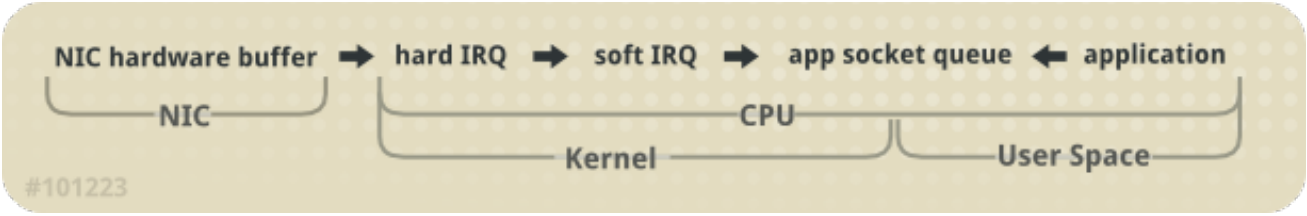


Figure 8.1. Diagramme du chemin de réception réseau

Le noyau Linux reçoit chaque trame et la soumet à un processus composé de quatre étapes :

1. *Réception du matériel* : la *carte d'interface réseau* (ou NIC) reçoit la trame sur le fil. Selon la configuration de son pilote, la carte réseau transfère la trame à une mémoire-tampon de matériel interne ou à un tampon en anneau spécifié.

2. *IRQ matérielle* : la NIC affirme la présence d'une trame net en interrompant le CPU. Ceci cause au pilote NIC de reconnaître l'interruption et de programmer une *opération IRQ logicielle*.

3. *IRQ logicielle* : cette étape implémente le processus de réception de trames et est exécuté dans un contexte **`softirq`**. Cela signifie que l'étape empêche toutes les applications de s'exécuter sur le CPU spécifié, mais permet toujours à des IRQ matérielles d'être affirmées.

   Dans ce contexte (exécution sur le même CPU en tant qu'IRQ matérielle, minimisant ainsi l'en-tête de verrouillage), le noyau supprime la trame des tampons du matériel de la carte réseau et la traite via la pile réseau. À partir de là, la trame est transférée, abandonnée, ou passée à un socket d'écoute cible.

   Lorsque passée à un socket, la trame est ajoutée à l'application propriétaire du socket. Ce processus est effectué de manière itérative jusqu'à ce qu'il ne reste plus de trames au tampon du matériel de la NIC, ou jusqu'à ce que *device weight* (**`dev_weight`**) soit atteint. Pour obtenir davantage d'informations sur « device weight », veuillez consulter la Section 8.4.1, « Mémoire tampon matérielle de la carte d'interface réseau (NIC) ».

4. *Réception de l'application* : l'application reçoit la trame et la sort de toute file d'attente de sockets possédés via des appels POSIX standards (**`read`**, **`recv`**, **`recvfrom`**). À ce moment, les données reçues sur le réseau n'existent plus sur la pile réseau.

CPU/cache affinity

Pour maintenir un haut débit sur le chemin de réception, il est recommandé de conserver le cache L2 *à chaud*. Comme décrit précédemment, les tampons réseau sont reçus sur le même CPU que les IRQ qui ont signalés leurs présences. Cela signifie que les données de tampon seront sur le cache L2 de ce CPU de réception.

Pour tirer profit de ceci, veuillez placer les affinités de processus sur les applications censées recevoir le plus de données sur la carte réseau qui partage le même cœur que le cache L2. Cela maximisera les chances d'accès au cache et améliorera les performances par la même occasion.

8.4. RÉSOUDRE DES PROBLÈMES COMMUNS DE PERTE DE FILE OU DE TRAME

La cause la plus commune pour une perte de trame est le *débordement de file d'attente*. Le noyau définit une limite à la longueur de la file et dans certains cas, la file se remplit plus vite qu'elle ne se vide. Lorsque ceci se produit trop longtemps, des trames finissent par être perdues.

Comme illustré dans Figure 8.1, « Diagramme du chemin de réception réseau », il existe deux files majeures dans le chemin de réception : la file de la mémoire tampon matérielle de la NIC et la file du socket. Ces deux files doivent être conformément configurées pour pouvoir être protégées des débordements de files.

8.4.1. Mémoire tampon matérielle de la carte d'interface réseau (NIC)

La NIC remplit sa mémoire tampon avec des trames ; la mémoire tampon est ensuite vidée par les interruptions **softirq**, ce que la carte réseau affirme via une interruption. Pour interroger le statut de cette file, veuillez utiliser la commande suivante :

```
ethtool -S ethX
```

Remplacez ***ethX*** par le nom du périphérique correspondant de la carte réseau. Ceci affichera combien de trames ont été supprimées dans ***ethX***. Souvent, les suppressions se produisent lorsque la file ne possède plus d'espace dans la mémoire tampon pour stocker les trames.

Il existe plusieurs manières de répondre à ce problème, principalement :

Trafic des entrées

Vous pouvez aider à empêcher les débordements de files en ralentissant le trafic des entrées. Ceci peut être effectué en filtrant, en réduisant le nombre de groupes de multidiffusion rejoints, en réduisant le trafic de diffusion, et ainsi de suite.

Longueur des files

Alternativement, vous pouvez aussi augmenter la longueur des files. Ceci implique l'augmentation des tampons dans une file spécifiée au maximum autorisé par le pilote. Pour ce faire, modifiez les paramètres d'anneau **rx**/**tx** de ***ethX*** en utilisant :

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

Ajoutez les valeurs **rx** ou **tx** appropriées à la commande mentionnée ci-dessus. Pour obtenir davantage d'informations, veuillez consulter **man ethtool**.

« Device weight » (Poids du périphérique)

Vous pouvez aussi augmenter la vitesse à laquelle une file est vidée. Pour ce faire, veuillez ajuster *device weight* en conséquence. Cet attribut fait référence ay nombre maximal de trames que la carte réseau peut recevoir avant que le contexte **softirq** ne doive céder au CPU et se reprogrammer. Il est contrôlé par la variable **/proc/sys/net/core/dev_weight**.

La plupart des administrateurs ont tendance à choisir la troisième option. N'oubliez pas, cependant, que des conséquences en résulteront. Augmenter le nombre de trames pouvant être reçues en provenance d'une carte d'interface réseau en une seule itération implique des cycles de CPU supplémentaires, pendant lesquels aucune application ne peut être programmée sur ce CPU.

8.4.2. File de sockets

Tout comme la file du matériel de la carte d'interface réseau, la file des sockets est remplie par la pile réseau en provenance du contexte **softirq**. Les applications vident ensuite les files de leurs sockets correspondants via des appels **read**, **recvfrom** et ainsi de suite.

Pour surveiller le statut de cette file, veuillez utiliser l'utilitaire **netstat** ; la colonne **Recv-Q** affiche la

taille de la file. De manière générale, les débordements dans la file des sockets sont gérés de la même façon que les débordements de mémoire tampon de matériel de carte d'interface réseau (voir la Section 8.4.1, « Mémoire tampon matérielle de la carte d'interface réseau (NIC) ») :

Trafic des entrées

La première option est de ralentir le trafic des entrées en configurant la vitesse à laquelle la file se remplit. Pour ce faire, filtrez les trames ou supprimez-les de manière préventive. Vous pouvez aussi ralentir le trafic des entrées en rabaissant le poids du périphérique de la carte d'interface réseau[6].

Profondeur de file

Vous pouvez aussi éviter les débordements de files de sockets en augmentant la profondeur des files. Pour ce faire, augmentez la valeur du paramètre de noyau **`rmem_default`** ou de l'option de socket **`SO_RCVBUF`**. Pour obtenir davantage d'informations sur les deux, veuillez consulter la Section 8.2, « Paramètres réseau optimisés ».

Fréquence des appels d'application

Lorsque possible, veuillez optimiser l'application afin qu'elle effectue des appels plus fréquemment. Cela implique la modification ou la reconfiguration de l'application réseau pour effectuer des appels POSIX (comme **`recv`**, **`read`**) plus fréquemment. En conséquence, cela permettra à une application de vider la file plus rapidement.

Pour de nombreux administrateurs, augmenter la profondeur de la file est une solution préférable. Il s'agit de la solution la plus facile, mais elle ne fonctionne pas toujours sur le long-terme. Comme les technologies de réseau sont de plus en plus rapides, les files de sockets continueront à se remplir plus vite. Sur le long-terme, cela signifie qu'il faudra réajuster la profondeur de file de manière conséquente.

La meilleure solution est d'améliorer ou de configurer l'application afin qu'elle vide les données du noyau plus rapidement, même si cela signifie mettre les données en file d'attente dans l'espace de l'application. Cela permet aux données d'être stockées de manière plus flexible puisqu'elles peuvent être mises en swap et remises en page comme nécessaire.

8.5. CONSIDÉRATIONS SUR LA MULTIDIFFUSION

Lorsque de multiples applications écoutent un groupe de multidiffusion, le code du noyau qui gère les trames de multidiffusion doit, de par sa conception, dupliquer les données réseau pour chaque socket individuel. Cette duplication prend du temps et se produit dans le contexte **`softirq`**.

Ainsi, l'ajout de multiples écouteurs sur un seul groupe de multidiffusion a un impact direct sur le temps d'exécution du contexte de **`softirq`**. L'ajout d'un écouteur à un groupe de multidiffusion implique que le noyau doit créer une copie supplémentaire pour chaque trame reçue pour ce groupe.

L'effet de ceci est minimal avec des faibles volumes de trafic et faibles nombres d'écouteurs. Cependant, lorsque de multiples sockets écoutent un groupe de multidiffusion à trafic volumineux, le temps d'exécution plus élevé du contexte **`softirq`** peut conduire à des pertes de trame sur les files de la carte réseau et des sockets. Un temps d'exécution **`softirq`** plus élevé se traduit par une réduction des possibilités pour l'exécution des applications sur des systèmes lourdement chargés. Ainsi, la vitesse à laquelle les trames de multidiffusion sont perdues augmente au même rythme que le nombre d'applications écoutant un groupe de multidiffusion volumineux augmente.

Résolvez cette perte de trame en optimisant les files des sockets et mémoires tampon de matériel NIC, comme le décrit la Section 8.4.2, « File de sockets » ou la Section 8.4.1, « Mémoire tampon matérielle de la carte d'interface réseau (NIC) ». Alternativement, vous pouvez optimiser l'utilisation des sockets

d'une application. Pour ce faire, configurez l'application pour contrôler un seul socket et rapidement disséminer les données de réseau reçues vers d'autres processus de l'espace utilisateur.

---

[4] S'assurer des affinités du cache entre un CPU et une NIC signifie les configurer pour qu'ils partagent le même cache L2. Pour obtenir des informations supplémentaires, veuillez consulter la Section 8.3, « Aperçu des réceptions de paquets ».

[5] La Section 8.3, « Aperçu des réceptions de paquets » contient un aperçu du voyage du paquet, ce qui devrait vous aider à trouver et mapper les zones prônes à une certaine congestion dans la pile réseau.

[6] Le poids d'un périphérique (« Device weight ») est contrôlé via **`/proc/sys/net/core/dev_weight`**. Pour obtenir davantage d'informations sur le poids d'un périphérique et les implications liées à son ajustement, veuillez consulter la Section 8.4.1, « Mémoire tampon matérielle de la carte d'interface réseau (NIC) » .

ANNEXE A. HISTORIQUE DES VERSIONS

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

Version 4.0-22.2 **Fri Jun 28 2013** **Sam Friedmann**
Fichiers de traduction synchronisés avec les sources XML 4.0-22.2

Version 4.0-22.1 **Thu Apr 18 2013** **Chester Cheng**
Fichiers de traduction synchronisés avec les sources XML 4.0-22

Version 4.0-22 **Fri Feb 15 2013** **Laura Bailey**
Publication pour Red Hat Enterprise Linux 6.4.

Version 4.0-19 **Wed Jan 16 2013** **Laura Bailey**
Corrections mineures pour une meilleure consistance (BZ#868404).

Version 4.0-18 **Tue Nov 27 2012** **Laura Bailey**
Publication pour Red Hat Enterprise Linux 6.4 Bêta.

Version 4.0-17 **Mon Nov 19 2012** **Laura Bailey**
Ajout des commentaires des experts de domaine fonctionnel concernant la section numad (BZ#868404).

Version 4.0-16 **Thu Nov 08 2012** **Laura Bailey**
Ajout de la section du brouillon sur numad (BZ#868404).

Version 4.0-15 **Wed Oct 17 2012** **Laura Bailey**
Application des commentaires des experts de domaine fonctionnel pour bloquer les discussions annulées et déplacement de la section sous « Options de montage » (BZ#852990).
Mise à jour des descriptions de profils de performance (BZ#858220).

Version 4.0-13 **Wed Oct 17 2012** **Laura Bailey**
Mise à jour des descriptions de profils de performance (BZ#858220).

Version 4.0-12 **Tue Oct 16 2012** **Laura Bailey**
Amélioration de la navigation du livre (BZ#854082).
Correction de la définition de ***file-max*** (BZ#854094).
Correction de la définition de ***threads-max*** (BZ#856861).

Version 4.0-9 **Tue Oct 9 2012** **Laura Bailey**
Ajout de la recommandation FSTRIM au chapitre « Systèmes de fichiers » (BZ#852990).
Mise à jour de la description du paramètre ***threads-max*** selon les commentaires des clients (BZ#856861).
Mise à jour de la remarque sur les améliorations de la gestion de la fragmentation GFS2 BZ#857782).

Version 4.0-6 **Thu Oct 4 2012** **Laura Bailey**
Ajout d'une nouvelle section sur l'utilitaire numastat (BZ#853274).

Version 4.0-3 **Tue Sep 18 2012** **Laura Bailey**
Ajout d'une remarque sur les capacités des nouvelles performances (BZ#854082).
Correction de la description du paramètre file-max (BZ#854094).

Version 4.0-2 **Mon Sep 10 2012** **Laura Bailey**
Ajout de la section sur BTRFS et d'une introduction de base sur les systèmes de fichiers (BZ#852978).
Remarque sur l'intégration Valgrind avec GDB (BZ#853279).

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

Ajout et mise à jour de descriptions de profils tuned-adm (BZ#803552).

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

Mise à jour des descriptions des paramètres threads-max et file-max (BZ#752825).

Mise à jour de la valeur par défaut du paramètre slice_idle (BZ#785054).

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

Restructuration et ajout de détails sur le taskset et la liaison de CPU et sur l'allocation de mémoire avec numactl sur la Section 4.1.2, « Régler les performances CPU » (BZ#639784).

Correction de l'utilisation des liens internes (BZ#786099).

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

Corrections mineures apportées à la Section 5.3, « Utiliser Valgrind pour établir un profil de l'utilisation de mémoire » (BZ#639793).

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

Consistance assurée entre les liens hypertexte internes et externes (BZ#752796).

Ajout de la Section 5.3, « Utiliser Valgrind pour établir un profil de l'utilisation de mémoire » (BZ#639793).

Ajout de la Section 4.1.2, « Régler les performances CPU » et restructuration de Chapitre 4, *CPU* (BZ#639784).

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

Publication pour mise à disponibilité générale de Red Hat Enterprise Linux 6.2.