cgroup

Présentation

Les **Control Groups** fournissent un mécanisme pour agréger/partitionner des jeux de tâches, et tous les futures enfants, dans des groupes hiérarchiques.

Un **cgroup** associe un jeu de tâches avec un jeu de paramètres pour un ou plusieurs sous-systèmes. un sous-système est un module qui utilise les facilités de groupement de tâches pour traiter ces groupes de tâche d'une manière particulière. un sous-système est typiquement un contrôleur de ressource.

Une hiérarchie est un jeu de **cgroups** arrangée en arborescence, de manière à ce que chaque tâche dans le système soit dans exactement un **cgroup** dans la hiérarchie. Chaque hiérarchie a une instance du **cgroup** de système de fichier virtuel avec lui.

Chaque tâche dans le système a un pointeur de référence vers un **css_set**. Un css_set contient un jeu de pointeur de référence vers des objets **cgroups_subsys_state**, un pour chaque **cgroup** enregistré dans le système.

Un système de fichier hiérarchique de **cgroup** peut être monté pour le voir et le manipuler depuis l'espace utilisateur. On peut lister toutes les tâches attachées à un cgroup.

Quand un cgroup est démonté, s'il y a des cgroups enfants, cette hiérarchie reste active. S'il n'y a pas de cgroups enfant, la hiérarchie est désactivé.

Chaque tâche sous **/proc** a un fichier nommé **cgroup** affichant, pour chaque hiérarchie active, les noms des sous systèmes et le nom du cgroup.

Chaque cgroup est représenté par un répertoire contenant les fichiers suivants :

tasks liste des tâches attachées au cgroup.

cgroup.procs liste des tgid dans ce cgroup.

notify_on_release Permet lorsque la dernière tâche d'un cgroup se termine et que le dernier cgroup enfant est supprimé, que le kernel lance une commande spécifiée dans le fichier release_agent

release_agent Le chemin à utiliser pour les notifications de release (existe dans le top cgroup uniquement)

cgroup.clone_children (0 ou 1) Tous les cgroups créés vont appeler post_clone pour chaque sous-système du nouveau cgroup. ce callback copie les valeurs du parent.

cgroup.event_control (0 ou 1) active l'exécution du release_agent

cpu planifie l'accès du cpu aux cgroup

cpu.shares Valeur entière de part relative du temps CPU disponible pour les tâches

cpu.rt_runtime_us Période en microsecondes contiguë la plus longue autorisé relative à rt_period_us

cpu.rt_period_us Période de temps en microsecondes de référence pour rt_runtime_us

cpuset Assigne des cpu et des nœud mémoire à des cgroups

cpuset.cpu_exclusive (0 ou 1) spécifie si des cpusets autres que celui-ci, des ses parents ou enfants peuvent partager les cpu spécifiés

cpuset.cpus Spécifie à quels CPU les tâches dans ce cgroup peuvent accéder

cpuset.mem_exclusive (0 ou 1) Spécifie si d'autre cpusets peuvent partager les nœuds de mémoire

cpuset.mem_hardwall (0 ou 1) Spécifie si les allocations au noyau de pages mémoire et données tampon sont restreintes aux nœuds mémoire spécifiés pour ce cpuset

cpuset.memory_migrate (0 ou 1) spécifie si une page en mémoire devrait migrer vers un nouveau nœud si les valeurs dans cpuset.mems changent

cpuset.memory_pressure Contient une moyenne de sollicitation de la mémoire des processus de ce cpuset. Est mis à jours lorsque memory_pressure_enabled est activé

cpuset.memory_pressure_enabled (0 ou 1) Spécifie si le système doit calculer la sollicitation mémoire (memory_pressure)

cpuset.memory_spread_page (0 ou 1) Spécifie si les tampons de système de fichier doivent être placés de manière régulières sur les nœuds mémoire alloués à ce cpuset

cpuset.memory_spread_slab (0 ou 1) Spécifie si les cache slab du noyau pour les opérations I/O doivent être placés de manière régulières sur le cpuset

cpuset.mems Spécifie les nœuds de mémoire auxquels les tâches de ce cgroup ont accès

cpuset.sched_load_balance (0 ou 1) Spécifie si le noyau va équilibrer les charges sur les CPU dans ce cpuset

cpuset.sched_relax_domain_level entier entre -1 et une valeur positive, représente la largeur de l'étendue des cpu sur laquelle le noyau va essayer d'équilibrer les charges. les valeurs sont :

- **-1** Utilise la valeur par défaut du système pour l'équilibrage de charges
- **0** Pas d'équilibrage de charge
- **1** Équilibre immédiatement les charges sur les threads du même coeur
- **2** Équilibre immédiatement les charges sur les coeurs du même chip
- **3** Équilibre immédiatement les charges sur les CPU du même nœud ou la même lame
- **4** Équilibre immédiatement les charges sur les CPU de même architecture NUMA
- **5** Équilibre immédiatement les charges sur tous les CPU sur architecture NUMA

cpuacct Rapports automatiques sur les cpu utilisés

cpuacct.stat Rapporte le nombre de cycles CPU en unité USER_HZ pris par les tâches de ce cgroup et ses enfants dans le mode système

cpuacct.usage Rapporte le nombre de cycles CPU total en nanosecondes pris par les tâches de ce cgroup et enfants

cpuacct.usage_percpu Rapporte le nombre de cycles CPU total en nanosecondes sur chaque CPU pris par toutes les taches de ce cgroup et ses enfants

blkio Surveille et contrôle l'accès des tâches aux entrées/sorites sur des périphériques block

blkio.io_merged Rapporte le nombre de requêtes BIOS fusionnées en requêtes pour des opérations I/O

blkio.io_queued Rapporte le nombre de requêtes en file d'attente pour des opérations I/O

blkio.io_service_bytes Rapporte le nombre d'octets transférés depuis et vers des périphériques spécifiques

blkio.io_serviced Rapporte le nombre d'opérations I/O effectuées sur des périphériques spécifiques

blkio.io_service_time Rapporte le temps total pris par l'envoie de la demande et son achèvement pour les opérations I/O sur des périphériques spécifiques

blkio.io_wait_time Rapporte le temps total pour un service dans les files d'attentes des opération I/O sur des périphériques spécifiques

blkio.reset_stats Réinitialise les statistiques enregistrées

blkio.sectors Rapporte le nombre de secteurs transférés depuis/vers des périphériques spécifiques

blkio.time Rapporte le moment auquel un cgroup avait un accès I/O à des périphériques spécifiques, en ms

blkio.weight Poids relatif d'accès au block I/O (de 100 à 1000)

blkio.weight_device Poids relatif d'accès à des périphériques spécifiques au format majeur :minuer poid

blkio.avg_queue_size Rapporte la taille moyenne des files d'attente pour les opérations I/O

blkio.group_wait_time Rapporte la durée totale (en ns) q'un cgroup attendu une tranche de temps pour l'une de ses files d'attente

blkio.empty_time Rapporte la durée totale (en ns) qu'un cgroup a attendu sans requêtes en attente

blkio.idle_time Rapporte le temps totale (en ns) que le planification a passé à attendre un cgroup dans l'attente d'une meilleur requête que celles déjà dns les files d'attentes ou provenant d'autres cgroups (si CONFIG_DEBUG_BLK_CGROUP=y)

blkio.dequeue Rapporte le nombre de fois que des requêtes d'opérations I/O ont été retirés des files d'attentes par des périphériques spécifiques

net_cls repère les paquets réseau avec un ClassId

net_cls.classid valeur hexa indiquant un descripteur de contrôle du traffic

devices Autorise ou refuse l'accès aux périphériques dans un cgroup

devices.allow Spécifie à quels périphériques les tâches du cgroup peuvent accéder. Chaque entrée possède 4 champs : type (a, b ou c), majeur, mineur et accès (r, w, m : autorise à créer des fichiers de périphériques qui n'existent pas encore)

devices.deny Spécifie les périphériques que le cgroup n'a pas accès

devices.list Rapporte les périphériques pour lesquels les contrôles d'accès ont été définis pour ce cgroup

freezer suspend ou réactive les tâches dans un cgroup

freezer.state FROZEN (les tâches dans le cgroup sont suspendues), FREEZING (le système est en train de suspendre les tâches dans le cgroup), THAWED (les tâches dans le cgroup sont réactivées)

perf_event Monitoring par cpu. (voir perf)

ns permet de grouper les processus dans des espaces de nom séparés

memory

memory.stat reporte des statistiques de mémoire :

cache Cache de la page, inclut tmpfs (shmem), en octets

rss caches swap et anonyme, n'inclut pas tmpfs (shmem), en octets

mapped_file taille des fichiers mappés en mémoire, inclut tmpfs (shmem), en octets

pgpgin nombre de pages chargées en mémoire

pgpgout nombre de pages renvoyées de la mémoire

swap usage swap, en octets

active_anon caches swap et anonyme de la liste des LRU (dernier récemment utilisé) actifs, inclut tmpfs (shmem), en octets

inactive_anon caches swap et anonyme de la liste des LRU (dernier récemment utilisé) inactifs, inclut tmpfs (shmem), en octets

active_file mémoire sauvegardée sur fichier de la liste des LRU actifs, en octets

inactive_file mémoire sauvegardée sur fichier de la liste des LRU inactifs, en octets

unevictable mémoire ne pouvant pas être récupérée, en octets

hierarchical_memory_limit limite de la mémoire pour la hiérarchie contenant le groupe de contrôle memory, en octets

hierarchical_memsw_limit limite de la mémoire plus le swap pour la hiérarchie contenant le groupe de contrôle memory, en octets

memory.usage_in_bytes Rapporte l'utilisation mémoire total actuelle en octets

memory.memsw.usage_in_bytes Rapporte la somme de l'utilisation de la mémoire et de l'espace swap

memory.max_usage_in_bytes Rapporte le montant max e mémoire utilisée

memory.memsw.max_usage_in_bytes Rapporte la somme totale de mémoire et swap utilisé

memory.limit_in_bytes Définis la mémoire max utilisateur incluant le cache (-1 pas de limite)

memory.memsw.limit_in_bytes mémoire et swap max (-1 pas de limite)

memory.failcnt Rapporte le nombre de fois que la limite mémoire est atteinte (memory.limit_in_bytes)

memory.memsw.failcnt Rapporte le nombre de fois que la limite mémoire est atteinte (memory.memsw.limit_in_bytes)

memory.force_empty Vide la mémoire de toutes les pages utilisée lorsque défini à 0. uniquement quand aucune tâche n'est présente

memory.swappiness Définis la tendance du noyau à déloger la mémoire plutôt que de réclamer des pages depuis le cache de page (idem à /proc/sys/vm/swappiness)

memory.use_hierarchie (0 ou 1) spécifie si la mémoire utilisée devrait être prise en comptes sur une hiérarchie de cgroup, à 0, le sous-système ne récupère pas la mémoire d'une tâche enfant.

Les nouveaux cgroups sont créé en utilisant l'appel système mkdir. les propriétés d'un cgroup sont modifiés en écrivant dans le fichier approprié dans ce cgroup.

Utiliser les cgroups

monter une hiérarchie cgroup :
mount -t tmpfs cgroup_root /sys/fs/cgroup
ajouter le sous-système cpuset :

mount -t cgroup cpuset -ocpuset /sys/fs/cgroup/cpuset
ajouter cpuset et memory :
mount -t cgroup -o cpuset,memory hier1 /sys/fs/cgroup/rg1
Changer de jeu de sous-systèmes :
mount -o remount,cpuset,blkio hier1 /sys/fs/cgroup/rg1
Spécifier le release_agent de la hiérarchie :
mount -t cgroup -o cpuset,release_agent="/sbin/cpuset_release_agent" xxx /sys/fs/cgroup/rg1
Changer la valeur du release_agent :
echo "/sbin/new_release_agent" > /sys/fs/cgroup/rg1/release_agent
Attacher une tâche à un cgroup :
echo ‘pidof bash‘ > tasks