Device Mapper

<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/logical_volume_manager_administration/device_mapper>

Le mappeur de périphérique

Le Device Mapper est un pilote de noyau qui fournit un cadre pour la gestion des volumes.   
Il fournit un moyen générique de créer des périphériques mappés, qui peuvent être utilisés comme volumes logiques.   
Il ne connaît pas spécifiquement les groupes de volumes ou les formats de métadonnées.

Le Device Mapper fournit la base d'un certain nombre de technologies de niveau supérieur. En plus de LVM, Device-Mapper multipath et la commande dm raid utilisent le Device Mapper.   
L'interface d'application de Device Mapper est l'appel ioctl système. L'interface utilisateur est la commande dmsetup.

Les volumes logiques LVM sont activés à l'aide du Device Mapper. Chaque volume logique est traduit en un périphérique mappé. Chaque segment se traduit par une ligne dans la table de mappage qui décrit le périphérique. Device Mapper prend en charge une variété de cibles de mappage, y compris le mappage linéaire, le mappage par bandes et le mappage d'erreurs. Par exemple, deux disques peuvent être concaténés en un volume logique avec une paire de mappages linéaires, un pour chaque disque. Lorsque LVM crée un volume, il crée un périphérique de mappage de périphérique sous-jacent qui peut être interrogé avec la dmsetupcommande. Pour plus d'informations sur le format des périphériques dans une table de mappage, reportez-vous à la Section A.1, « Mappages de tables de périphériques » . Pour plus d'informations sur l'utilisation de la dmsetupcommande pour interroger un périphérique, reportez-vous à la Section A.2, « La commande dmsetup » .

A.1. Mappages de tables de périphériques

Un périphérique mappé est défini par une table qui spécifie comment mapper chaque plage de secteurs logiques du périphérique à l'aide d'un mappage de table de périphériques pris en charge. Le tableau d'un périphérique mappé est construit à partir d'une liste de lignes de la forme :

start length mapping[ ]

mapping\_parameters...

Dans la première ligne d'un tableau Device Mapper, le startparamètre doit être égal à 0. Les paramètres start+ lengthd'une ligne doivent être égaux startà ceux de la ligne suivante. Les paramètres de mappage spécifiés dans une ligne de la table de mappage dépendent du mappingtype spécifié sur la ligne.

Les tailles dans le Device Mapper sont toujours spécifiées en secteurs (512 octets).

Lorsqu'un périphérique est spécifié en tant que paramètre de mappage dans le Device Mapper, il peut être référencé par le nom du périphérique dans le système de fichiers (par exemple, /dev/hda) ou par les numéros majeurs et mineurs au format major: minor. Le format major:minor est préféré car il évite les recherches de chemin d'accès.

Ce qui suit montre un exemple de table de mappage pour un périphérique. Dans ce tableau, il y a quatre cibles linéaires :

0 35258368 linéaire 8:48 65920

35258368 35258368 linéaire 8:32 65920

70516736 17694720 linéaire 8:16 17694976

88211456 17694720 linéaire 8:16 256

Les 2 premiers paramètres de chaque ligne sont le bloc de départ du segment et la longueur du segment. Le mot-clé suivant est la cible de mappage, qui dans tous les cas de cet exemple est linear. Le reste de la ligne se compose des paramètres d'une linearcible.

Les sous-sections suivantes décrivent le format des mappages suivants :

linéaire

rayé

miroir

snapshot et snapshot-origin

Erreur

zéro

multivoie

crypte

A.1.1. La cible de cartographie linéaire

Une cible de mappage linéaire mappe une plage continue de blocs sur un autre périphérique de bloc. Le format d'une cible linéaire est le suivant :

start length linéaire device offset

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

device

périphérique bloc, référencé par le nom du périphérique dans le système de fichiers ou par les numéros majeurs et mineurs au format major:minor

offset

décalage de départ de la cartographie sur l'appareil

L'exemple suivant montre une cible linéaire avec un bloc de départ dans le périphérique virtuel de 0, une longueur de segment de 1638400, une paire de nombres majeur:mineur de 8:2 et un décalage de départ pour le périphérique de 41146992.

0 16384000 linéaire 8:2 41156992

L'exemple suivant montre une cible linéaire avec le paramètre de périphérique spécifié comme périphérique /dev/hda.

0 20971520 linéaire /dev/hda 384

A.1.2. La cible cartographique rayée

La cible de mappage entrelacé prend en charge l'entrelacement sur les périphériques physiques. Il prend comme arguments le nombre de bandes et la taille du segment de bande suivis d'une liste de paires de nom de périphérique et de secteur. Le format d'une cible rayée est le suivant :

start length rayé #stripes chunk\_size device1 offset1 ... deviceN offsetN

Il existe un ensemble de paramètres deviceet offsetpour chaque bande.

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

#stripes

nombre de bandes pour le périphérique virtuel

chunk\_size

nombre de secteurs écrits sur chaque bande avant de passer à la suivante ; doit être une puissance de 2 au moins aussi grande que la taille de la page du noyau

device

périphérique bloc, référencé par le nom du périphérique dans le système de fichiers ou par les numéros majeurs et mineurs au format major: minor.

offset

décalage de départ de la cartographie sur l'appareil

L'exemple suivant montre une cible rayée avec trois bandes et une taille de segment de 128 :

0 73728 rayé 3 128 8:9 384 8:8 384 8:7 9789824

0

bloc de départ dans le périphérique virtuel

73728

longueur de ce segment

rayé 3 128

bande sur trois appareils avec une taille de bloc de 128 blocs

8:9

majeur:numéros mineurs du premier appareil

384

décalage de départ du mappage sur le premier appareil

8:8

majeur:numéros mineurs du deuxième appareil

384

décalage de départ du mappage sur le deuxième appareil

8:7

majeur:numéros mineurs du troisième appareil

9789824

décalage de départ du mappage sur le troisième appareil

L'exemple suivant montre une cible rayée pour 2 bandes avec des morceaux de 256 Kio, avec les paramètres de périphérique spécifiés par les noms de périphérique dans le système de fichiers plutôt que par les numéros majeurs et mineurs.

0 65536 rayé 2 512 /dev/hda 0 /dev/hdb 0

A.1.3. La cible de mappage miroir

La cible de mappage miroir prend en charge le mappage d'un périphérique logique miroir. Le format d'une cible en miroir est le suivant :

start length miroir log\_type #logargs logarg1 ... logargN #devs device1 offset1 ... deviceN offsetN

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

log\_type

Les types de journaux possibles et leurs arguments sont les suivants :

core

Le miroir est local et le journal du miroir est conservé dans la mémoire centrale. Ce type de journal prend 1 à 3 arguments :

taille des régions [[ no] sync] [ block\_on\_error]

disk

Le miroir est local et le journal du miroir est conservé sur le disque. Ce type de journal prend 2 à 4 arguments :

logdevice regionsize [[ no] sync] [ block\_on\_error]

clustered\_core

Le miroir est mis en cluster et le journal du miroir est conservé dans la mémoire centrale. Ce type de journal prend 2 à 4 arguments :

regionsize UUID [[ no] sync] [ block\_on\_error]

clustered\_disk

Le miroir est mis en cluster et le journal du miroir est conservé sur le disque. Ce type de journal prend 3 à 5 arguments :

logdevice regionsize UUID [[ no] sync] [ block\_on\_error]

LVM tient un petit journal qu'il utilise pour garder une trace des régions synchronisées avec le ou les miroirs. L' argument regionsize spécifie la taille de ces régions.

Dans un environnement en cluster, l' argument UUID est un identifiant unique associé au périphérique de journal miroir afin que l'état du journal puisse être maintenu dans tout le cluster.

L' [no]syncargument optionnel peut être utilisé pour spécifier le miroir comme "in-sync" ou "out-of-sync". L' block\_on\_errorargument est utilisé pour dire au miroir de répondre aux erreurs plutôt que de les ignorer.

#log\_args

nombre d'arguments de journal qui seront spécifiés dans le mappage

logargs

les arguments de journal pour le miroir ; le nombre d'arguments de journal fournis est spécifié par le #log-argsparamètre et les arguments de journal valides sont déterminés par le log\_typeparamètre.

#devs

le nombre de jambes dans le miroir; un appareil et un décalage est spécifié pour chaque jambe

device

périphérique de bloc pour chaque branche miroir, référencé par le nom du périphérique dans le système de fichiers ou par les numéros majeurs et mineurs au format major: minor. Un périphérique de bloc et un décalage sont spécifiés pour chaque jambe de miroir, comme indiqué par le #devsparamètre.

offset

décalage de départ du mappage sur l'appareil. Un périphérique de bloc et un décalage sont spécifiés pour chaque jambe de miroir, comme indiqué par le #devsparamètre.

L'exemple suivant montre une cible de mappage de miroir pour un miroir en cluster avec un journal de miroir conservé sur le disque.

0 52428800 miroir clustered\_disk 4 253:2 1024 UUID block\_on\_error 3 253:3 0 253:4 0 253:5 0

0

bloc de départ dans le périphérique virtuel

52428800

longueur de ce segment

miroir clustered\_disk

cible miroir avec un type de journal spécifiant que le miroir est en cluster et que le journal du miroir est conservé sur le disque

4

4 arguments de journal miroir suivront

253:2

majeur:numéros mineurs du périphérique de journalisation

1024

taille de la région que le journal miroir utilise pour garder une trace de ce qui est synchronisé

UUID

UUID du périphérique de journal miroir pour conserver les informations de journal dans l'ensemble d'un cluster

block\_on\_error

le miroir doit répondre aux erreurs

3

nombre de pieds dans le miroir

253:3 0 253:4 0 253:5 0

majeur:nombres mineurs et décalage pour les dispositifs constituant chaque branche du miroir

A.1.4. Les cibles de mappage d'instantané et d'origine d'instantané

Lorsque vous créez le premier instantané LVM d'un volume, quatre périphériques Device Mapper sont utilisés :

Un périphérique avec un linearmappage contenant la table de mappage d'origine du volume source.

Un périphérique avec un linearmappage utilisé comme périphérique de copie sur écriture (COW) pour le volume source ; pour chaque écriture, les données d'origine sont enregistrées dans le périphérique COW de chaque instantané pour conserver son contenu visible inchangé (jusqu'à ce que le périphérique COW se remplisse).

Un périphérique avec un snapshotmappage combinant #1 et #2, qui est le volume d'instantané visible.

Le volume "original" (qui utilise le numéro de périphérique utilisé par le volume source d'origine), dont la table est remplacée par un mappage "snapshot-origin" du périphérique #1.

Un schéma de nommage fixe est utilisé pour créer ces périphériques. Par exemple, vous pouvez utiliser les commandes suivantes pour créer un volume LVM nommé baseet un volume d'instantané nommé en snapfonction de ce volume.

#

#lvcreate -L 1G -n base volumeGrouplvcreate -L 100M --snapshot -n snap volumeGroup/base

Cela donne quatre appareils, que vous pouvez afficher avec les commandes suivantes :

# dmsetup table|grep volumeGroup

volumeGroup-base-real: 0 2097152 linéaire 8:19 384

volumeGroup-snap-cow: 0 204800 linéaire 8:19 2097536

volumeGroup-snap : 0 2097152 instantané 254:11 254:12 P 16

volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-\*

brw -------- 1 racine racine 254, 11 il y a 29 ans 18:15 /dev/mapper/volumeGroup-base-real

brw------- 1 racine racine 254, 12 29 il y a 18:15 /dev/mapper/volumeGroup-snap-cow

brw------- 1 racine racine 254, 13 29 il y a 18:15 /dev/mapper/volumeGroup-snap

brw------- 1 racine racine 254, 10 29 il y a 18:14 /dev/mapper/volumeGroup-base

Le format de la snapshot-origincible est le suivant :

start length origine-instantané origin

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

origin

volume de base de l'instantané

Le snapshot-originaura normalement un ou plusieurs instantanés basés sur lui. Les lectures seront mappées directement sur le périphérique de support. Pour chaque écriture, les données d'origine seront enregistrées dans le périphérique COW de chaque instantané pour conserver son contenu visible inchangé jusqu'à ce que le périphérique COW se remplisse.

Le format de la snapshotcible est le suivant :

start lengthinstantané P|Norigin COW-devicechunksize

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

origin

volume de base de l'instantané

COW-device

appareil sur lequel les blocs de données modifiés sont stockés

P|N

P (persistant) ou N (non persistant) ; indique si l'instantané survivra après le redémarrage. Pour les instantanés transitoires (N), moins de métadonnées doivent être enregistrées sur le disque ; ils peuvent être conservés en mémoire par le noyau.

chunksize

taille en secteurs des blocs de données modifiés qui seront stockés sur l'appareil COW

L'exemple suivant montre une snapshot-origincible avec un périphérique d'origine de 254:11.

0 2097152 origine de l'instantané 254:11

L'exemple suivant montre une snapshotcible avec un périphérique d'origine de 254:11 et un périphérique COW de 254:12. Ce périphérique d'instantané est persistant à travers les redémarrages et la taille de bloc pour les données stockées sur le périphérique COW est de 16 secteurs.

0 2097152 instantané 254:11 254:12 P 16

A.1.5. L'erreur Mapping Target

Avec une cible de mappage d'erreur, toute opération d'E/S vers le secteur mappé échoue.

Une cible de mappage d'erreurs peut être utilisée pour les tests. Pour tester le comportement d'un périphérique en cas de panne, vous pouvez créer un mappage de périphérique avec un secteur défectueux au milieu d'un périphérique, ou vous pouvez échanger la jambe d'un miroir et remplacer la jambe par une cible d'erreur.

Une cible d'erreur peut être utilisée à la place d'un périphérique défaillant, afin d'éviter les délais d'attente et les nouvelles tentatives sur le périphérique réel. Il peut servir de cible intermédiaire pendant que vous réorganisez les métadonnées LVM en cas d'échec.

La errorcible de mappage ne prend aucun paramètre supplémentaire en plus des paramètres de début et de longueur .

L'exemple suivant montre une errorcible.

0 65536 erreur

A.1.6. La cible de cartographie zéro

La zerocible de mappage est un périphérique de bloc équivalent à /dev/zero. Une opération de lecture sur ce mappage renvoie des blocs de zéros. Les données écrites dans ce mappage sont ignorées, mais l'écriture réussit. La zerocible de mappage ne prend aucun paramètre supplémentaire en plus des paramètres de début et de longueur .

L'exemple suivant montre une zerocible pour un périphérique de 16 To.

0 65536 zéro

A.1.7. La cible de mappage multi-chemins

La cible de mappage multichemin prend en charge le mappage d'un périphérique multichemin. Le format de la multipathcible est le suivant :

start length multipath #features [feature1 ... featureN] #handlerargs [handlerarg1 ... handlerargN] #pathgroups pathgroup pathgroupargs1 ... pathgroupargsN

Il existe un ensemble de pathgroupargsparamètres pour chaque groupe de chemins.

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

#features

Le nombre de fonctionnalités multipath, suivi de ces fonctionnalités. Si ce paramètre est égal à zéro, alors il n'y a pas de featureparamètre et le prochain paramètre de mappage de périphérique est #handlerargs. Actuellement, il existe une fonctionnalité prise en charge qui peut être définie avec l' featuresattribut dans le multipath.conffichier, queue\_if\_no\_path. Cela indique que ce périphérique multichemin est actuellement configuré pour mettre en file d'attente les opérations d'E/S si aucun chemin n'est disponible.

Dans l'exemple suivant, l' no\_path\_retryattribut du multipath.conffichier a été défini pour mettre en file d'attente les opérations d'E/S uniquement jusqu'à ce que tous les chemins aient été marqués comme ayant échoué après un certain nombre de tentatives d'utilisation des chemins. Dans ce cas, le mappage apparaît comme suit jusqu'à ce que tous les vérificateurs de chemin aient échoué au nombre de vérifications spécifié.

0 71014400 multipath 1 queue\_if\_no\_path 0 2 1 round-robin 0 2 1 66:128 \

1000 65:64 1000 round robin 0 2 1 8:0 1000 67:192 1000

Une fois que tous les vérificateurs de chemin ont échoué au nombre de vérifications spécifié, le mappage apparaît comme suit.

0 71014400 multivoies 0 0 2 1 tourniquet 0 2 1 66:128 1000 65:64 1000 \

tournoi à la ronde 0 2 1 8:0 1000 67:192 1000

#handlerargs

Le nombre d'arguments du gestionnaire de matériel, suivi de ces arguments. Un gestionnaire de matériel spécifie un module qui sera utilisé pour effectuer des actions spécifiques au matériel lors du changement de groupe de chemins ou de la gestion des erreurs d'E/S. S'il est défini sur 0, le paramètre suivant est #pathgroups.

#pathgroups

Le nombre de groupes de chemins. Un groupe de chemins est l'ensemble des chemins sur lesquels un périphérique multichemin équilibrera la charge. Il existe un ensemble de pathgroupargsparamètres pour chaque groupe de chemins.

pathgroup

Le prochain groupe de chemins à essayer.

pathgroupsargs

Chaque groupe de chemins comprend les arguments suivants :

pathselector #selectorargs #paths #pathargs device1 ioreqs1 ... deviceN ioreqsN

Il existe un ensemble d'arguments de chemin pour chaque chemin du groupe de chemins.

pathselector

Spécifie l'algorithme utilisé pour déterminer le chemin de ce groupe de chemins à utiliser pour la prochaine opération d'E/S.

#selectorargs

Le nombre d'arguments de sélecteur de chemin qui suivent cet argument dans le mappage multichemin. Actuellement, la valeur de cet argument est toujours 0.

#paths

Le nombre de chemins dans ce groupe de chemins.

#pathargs

Le nombre d'arguments de chemin spécifiés pour chaque chemin dans ce groupe. Actuellement, ce nombre est toujours 1, l' ioreqsargument.

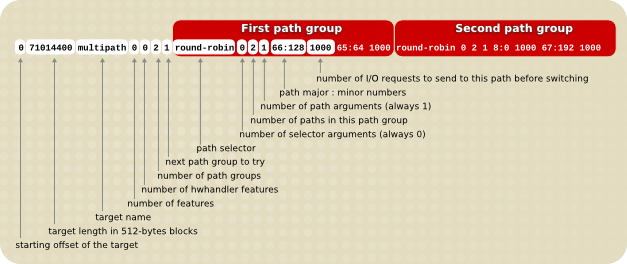
device

Le numéro de périphérique de bloc du chemin, référencé par les numéros majeurs et mineurs au format major:minor

ioreqs

Le nombre de demandes d'E/S à router vers ce chemin avant de passer au chemin suivant dans le groupe actuel.

La Figure A.1, « Cible de mappage multivoies » montre le format d'une cible multivoies avec deux groupes de voies.



Cible de mappage multichemin

Graphique A.1. Cible de mappage multichemin

L'exemple suivant montre une définition de cible de basculement pur pour le même périphérique multichemin. Dans cette cible, il existe quatre groupes de chemins, avec un seul chemin ouvert par groupe de chemins, de sorte que le périphérique multichemin n'utilisera qu'un seul chemin à la fois.

0 71014400 multivoies 0 0 4 1 tourniquet 0 1 1 66:112 1000 \

tournoi à la ronde 0 1 1 67:176 1000 tournoi à la ronde 0 1 1 68:240 1000 \

tournoi à la ronde 0 1 1 65:48 1000

L'exemple suivant montre une définition de cible à diffusion complète (multibus) pour le même périphérique multichemin. Dans cette cible, il n'y a qu'un seul groupe de chemins, qui inclut tous les chemins. Dans cette configuration, multipath répartit la charge uniformément sur tous les chemins.

0 71014400 multivoies 0 0 1 1 tourniquet 0 4 1 66:112 1000 \

67:176 1000 68:240 1000 65:48 1000

Pour plus d'informations sur le multipathing, consultez le manuel DM Multipath .

A.1.8. La cible de mappage de la crypte

La cryptcible crypte les données transitant par le périphérique spécifié. Il utilise l'API Crypto du noyau.

Le format de la cryptcible est le suivant :

start length crypte cipher key IV-offset device offset

start

bloc de départ dans le périphérique virtuel

length

longueur de ce segment

cipher

Le chiffre se compose de cipher[-chainmode]-ivmode[:iv options].

cipher

Les chiffrements disponibles sont répertoriés dans /proc/crypto(par exemple, aes).

chainmode

Utilisez toujours cbc. Ne pas utiliser ebc; il n'utilise pas de vecteur initial (IV).

ivmode[:iv options]

IV est un vecteur initial utilisé pour faire varier le chiffrement. Le mode IV est plainou essiv:hash. Un ivmodede -plainutilise le numéro de secteur (plus le décalage IV) comme IV. Un ivmodeof -essivest une amélioration évitant une faiblesse du filigrane.

key

Clé de cryptage, fournie en hexadécimal

IV-offset

Décalage du vecteur initial (IV)

device

périphérique bloc, référencé par le nom du périphérique dans le système de fichiers ou par les numéros majeurs et mineurs au format major:minor

offset

décalage de départ de la cartographie sur l'appareil

Ce qui suit est un exemple de cryptcible.

0 2097152 crypte aes-plain 0123456789abcdef0123456789abcdef 0 /dev/hda 0