Empowered by Innovation NEC

device-mapper 解説

日本電気株式会社
野村　淳一
2009年9月18日

device-mapper 解説

- はじめに
- device-mapper とは
- 各種 device-mapper ターゲットの機能
- device-mapper の概念と構造
- device-mapper を使う
- Request-based device-mapper の紹介
- さいごに

はじめに

はじめに

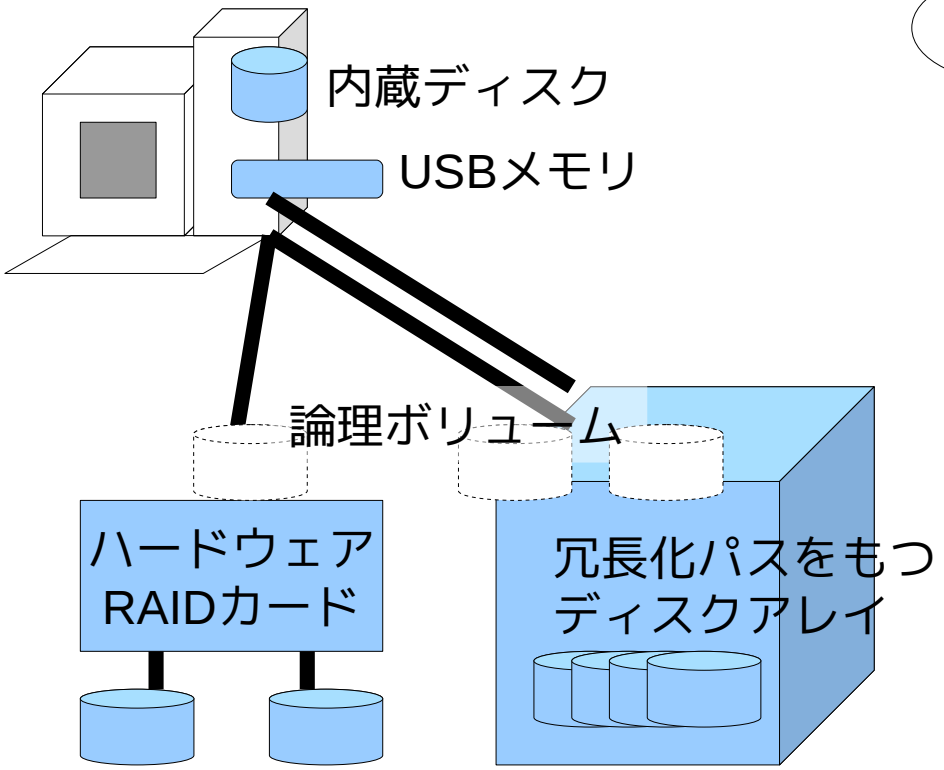
- ハードディスクなどのストレージデバイスは高速化・大容量化の一途をたどっていますが、拡張性や耐障害性、暗号化、バックアップなど機能面での重要性も増しています。
- Linux では、このような機能拡張をソフトウェア的に行なうために device-mapper という仕組みが 2.6 カーネルから取り込まれています。
- このチュートリアルでは、device-mapper について、その背景や機能、内部構造について解説します。

device-mapper とは

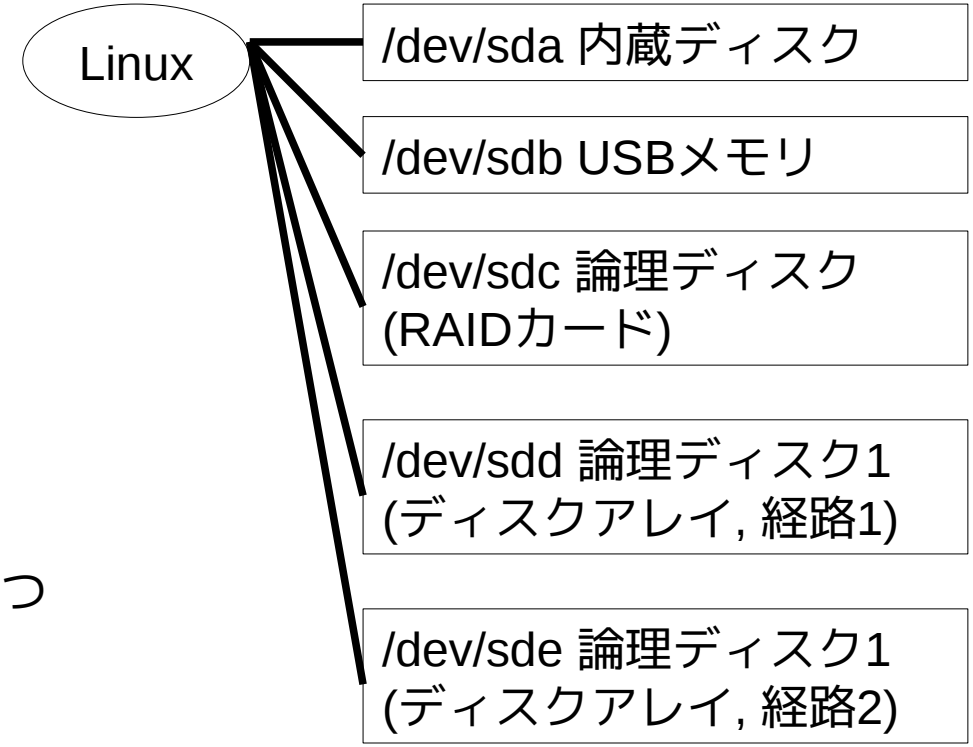
Linux におけるストレージデバイス

ストレージデバイスは「ブロックデバイス」として抽象化

例えば以下のようなデバイスが

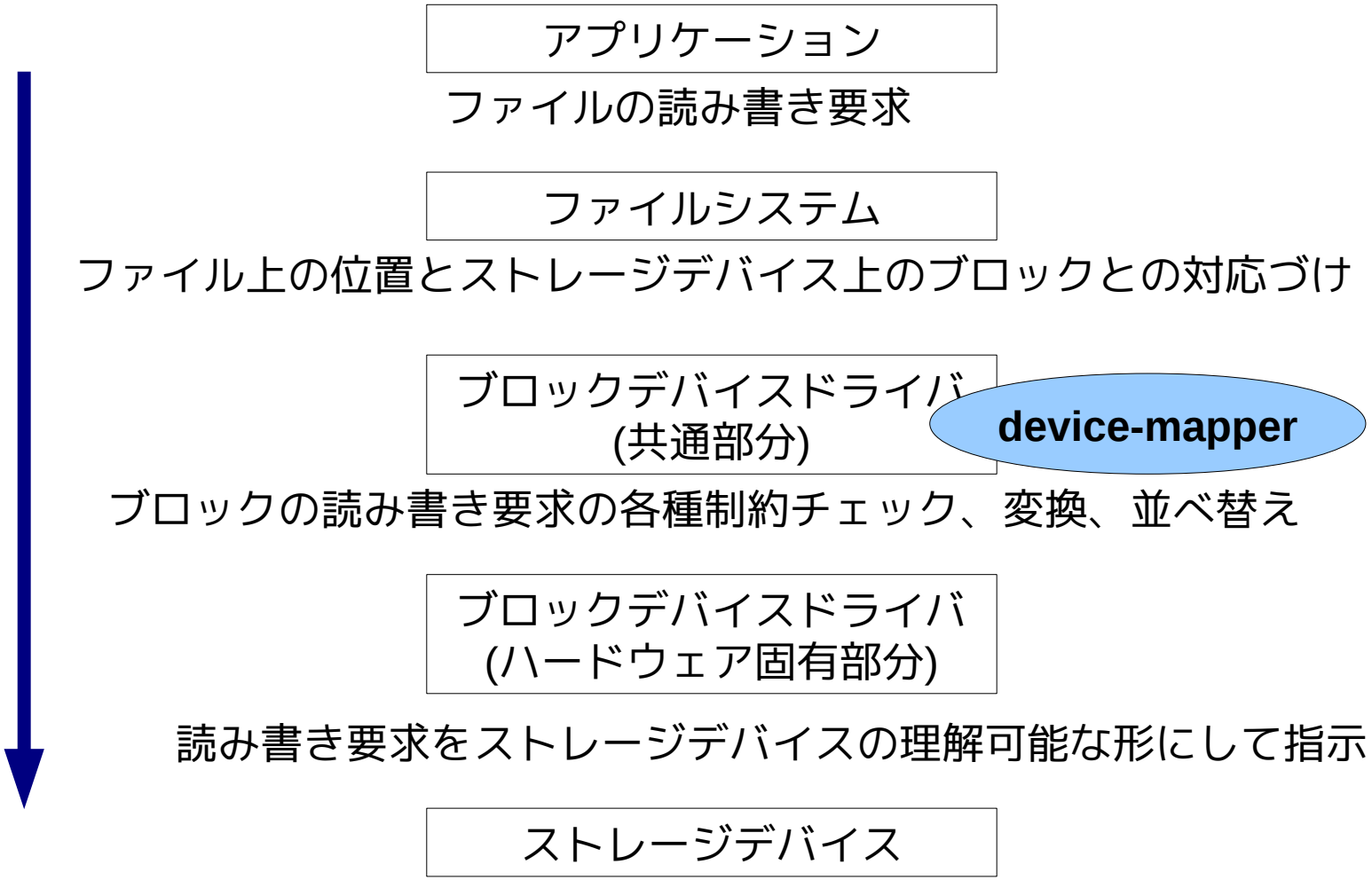


Linux上では以下のように



ストレージデバイスへのI/O

- ストレージデバイスへの I/O は概ね以下のような流れになる



device-mapper とは

- device-mapper は、2.6 カーネルから取り込まれた Linux のブロックデバイスドライバおよびそれをサポートするライブラリ群 (“dm”, “DM” と略される)
- ブロックデバイスの共通部分にプラグインし、ブロックの読み書き要求に対して様々な変換を加えることができる
- 開発された経緯
  - 2.6 カーネル開発時、論理ボリュームの管理機構として、EVMS と LVM2 という二つの機能が競合していた。
  - 多機能化しコードが複雑化する中で、論理ボリュームに対するI/Oを物理デバイスに対するI/Oにマッピングする機能を双方のカーネル内共通部分として括り出し、論理ボリュームの構成管理のような固有の処理はユーザスペースで行なうようにしたのが、device-mapper。

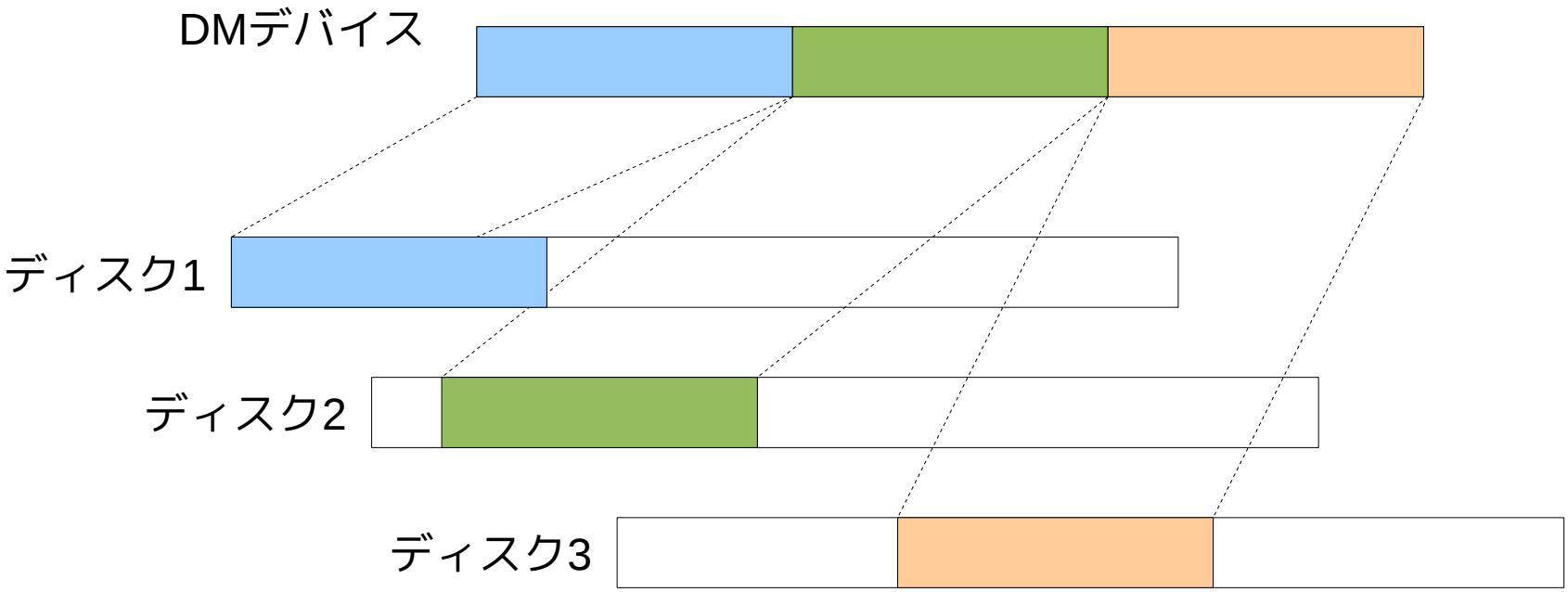
device-mapper の機能と利用例

- 一つまたは複数のブロックデバイスへの I/O をとりまとめ変換を加える機能を提供する
  - Linear (ブロック位置のオフセット変更)
  - Striped (I/O 要求を一定サイズで分割して複数のデバイスへ発行)
  - Mirror (I/O 要求を複数のデバイスへ発行)
  - Multipath (I/O 要求をいずれか一つのデバイスに発行)
  - Snapshot (ブロック内容変更時に旧内容をバックアップにコピー)
  - Crypt (ブロック単位での暗号化・復号)
- 使用アプリケーション
  - LVM2 (論理ボリュームマネージャ)
  - multipath-tools (冗長パス管理ツール)
  - dmraid (ソフトウェア RAID 管理ツール)
  - cryptsetup (暗号化デバイス管理ツール)

各種 device-mapper ターゲットの機能

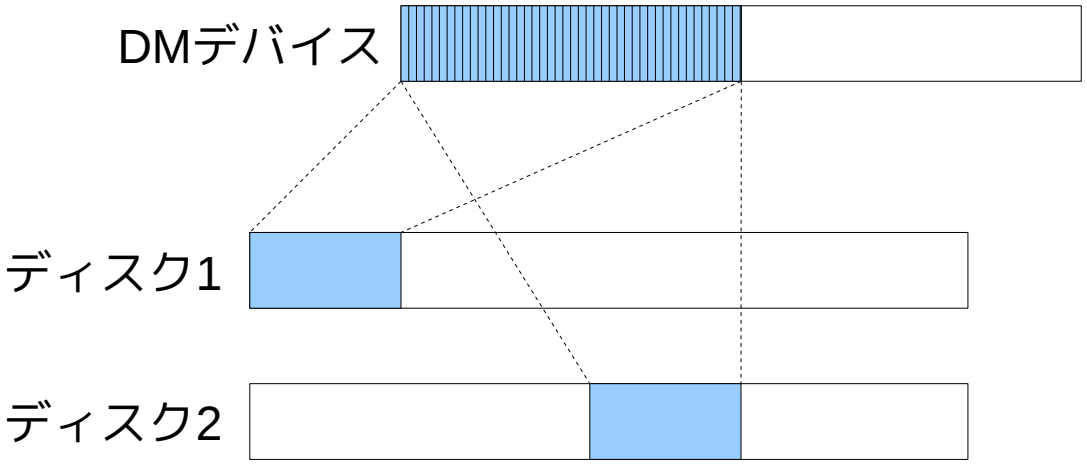
linear

- 最も基本的なターゲット
- 指定オフセットを加えて他のブロックデバイスにI/Oを転送



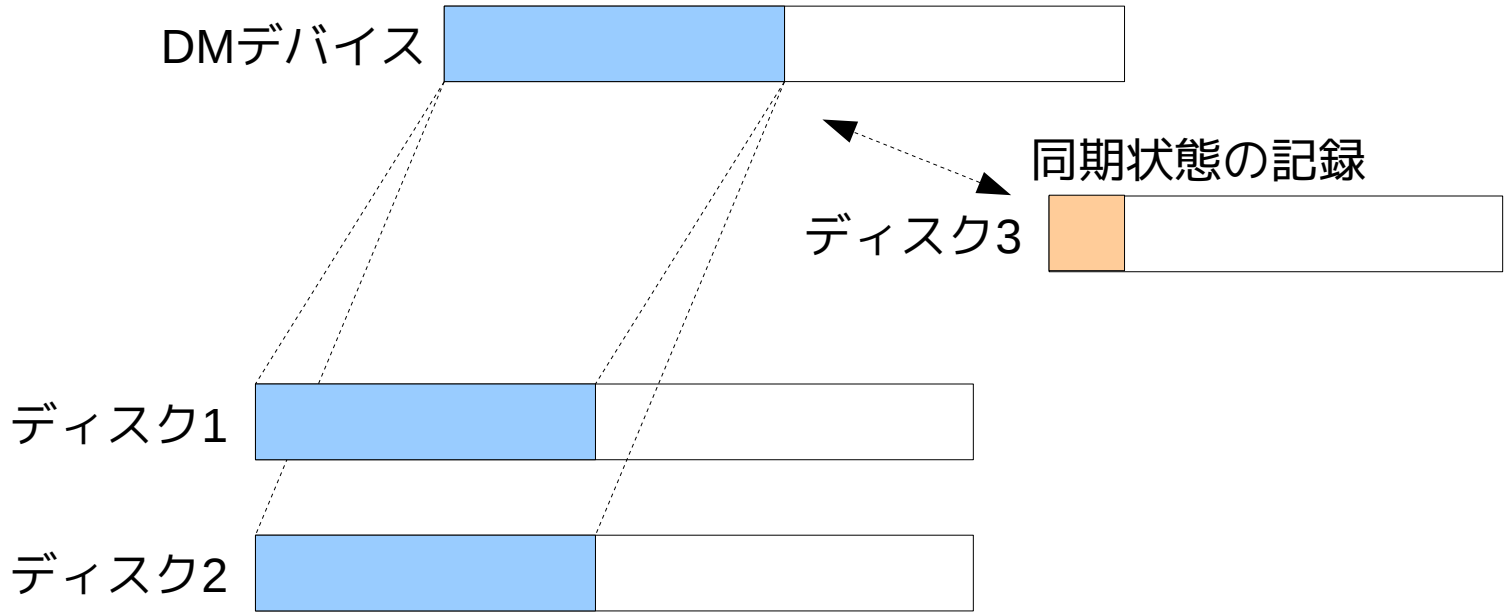
striped

- 指定のストライプサイズ境界で、複数のデバイスにI/Oを振り分ける (RAID0 相当)



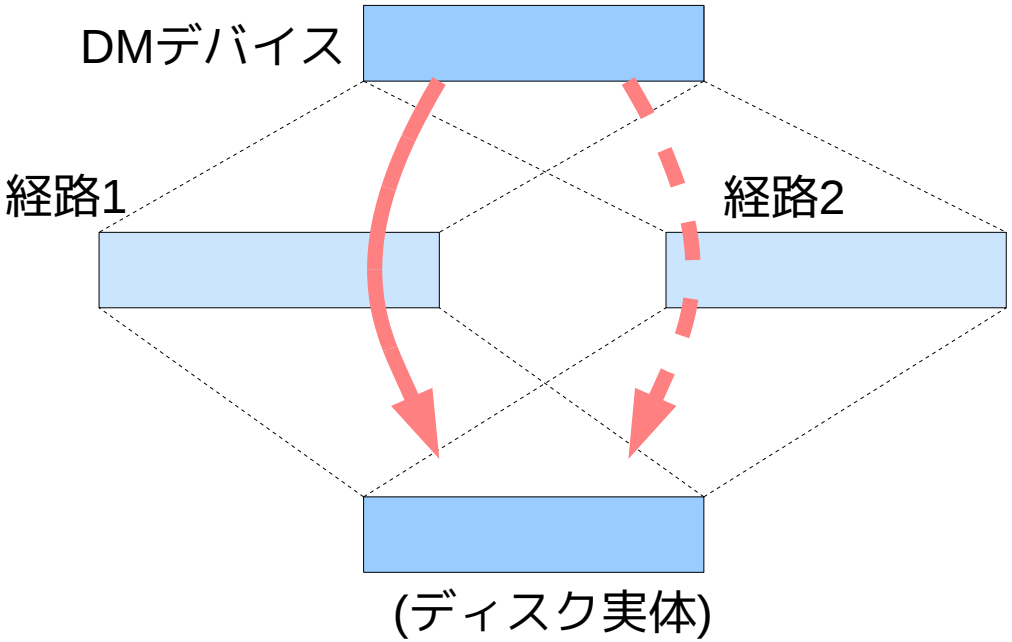
mirror

- 書き込みは配下の全デバイスに、読み込みはいずれか一つのから(RAID1相当)
- ディスク上の各位置について複数のデバイス間で内容が一致しているかどうかを随時記録・更新するログデバイスを別途指定可能
- ディスクの新規追加や突発的な電源断による再起動など、同期状態にずれがある場合、ディスク間でコピーを行なう



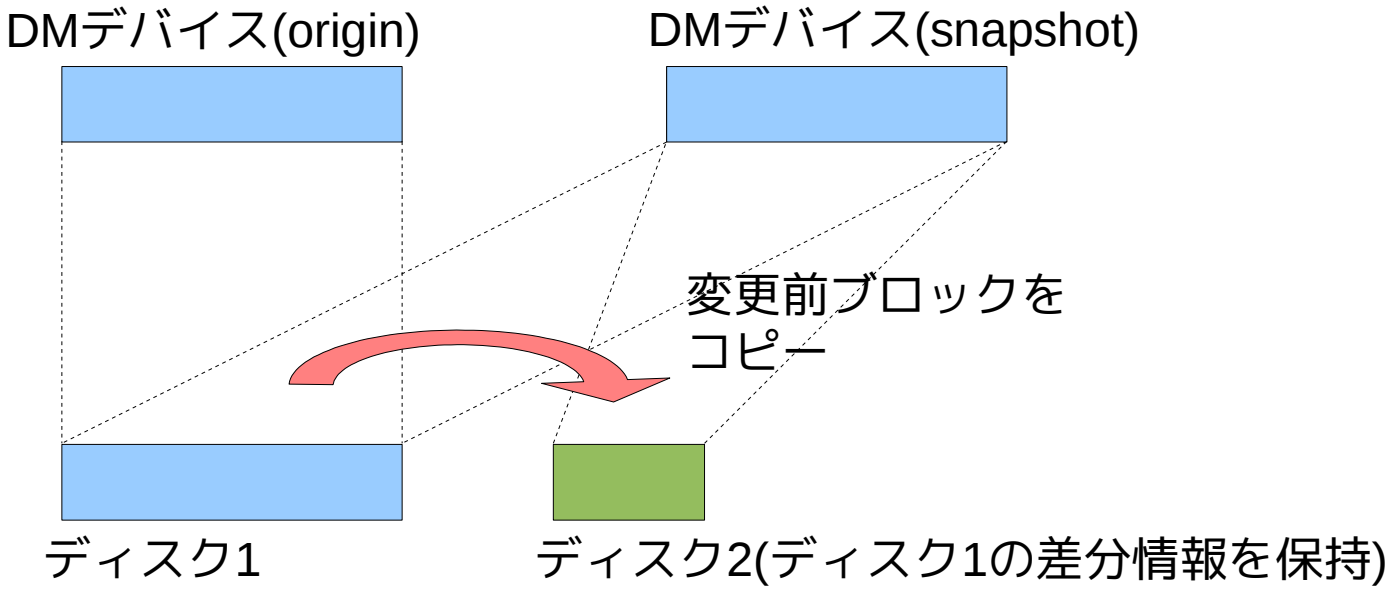
multipath

- ディスク実体に対して複数のパス(アクセス経路)がある場合、Linuxでは各パスについて別々のブロックデバイスが割り当てられる
- multipath ターゲットでは、それらのパスを一つの DM デバイスにまとめ、I/Oを振り分けたり、パスが切断された場合に別パスでリトライしたりする



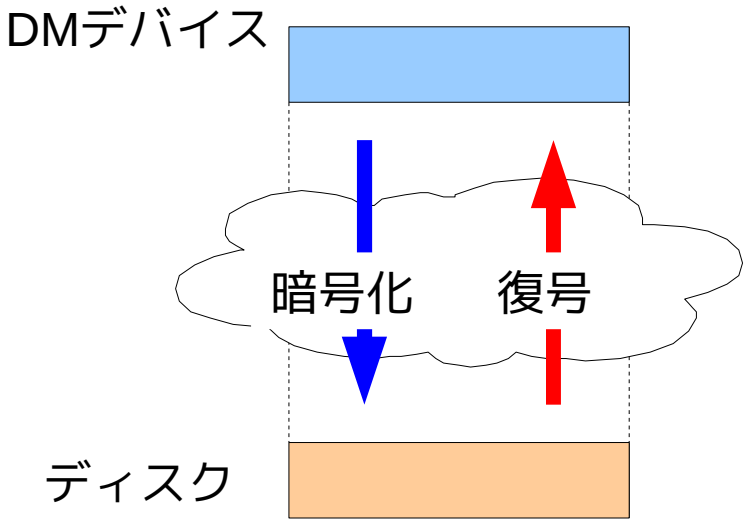
snapshot

- snapshot-origin と snapshot の2種類のターゲットドライバから構成される
- snapshot-origin は linear マッピングと同様だが、書き込みがあった場合、snapshot ターゲットに通知する
- snapshot は通知を受けると古い内容をコピーする。読み込み時、コピーがあればそれを、なければoriginの内容を返す



crypt

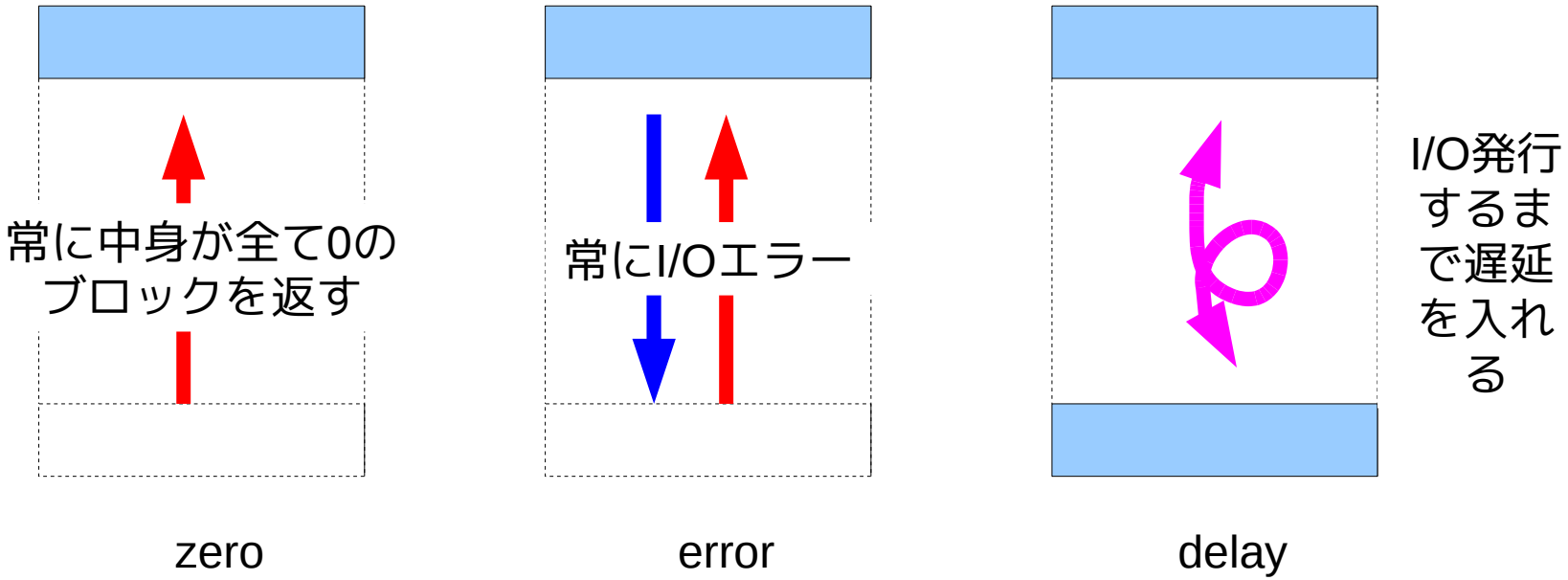
- テーブルのパラメータで指定された方法で、ブロック内容を暗号化・復号する



zero, error, delay

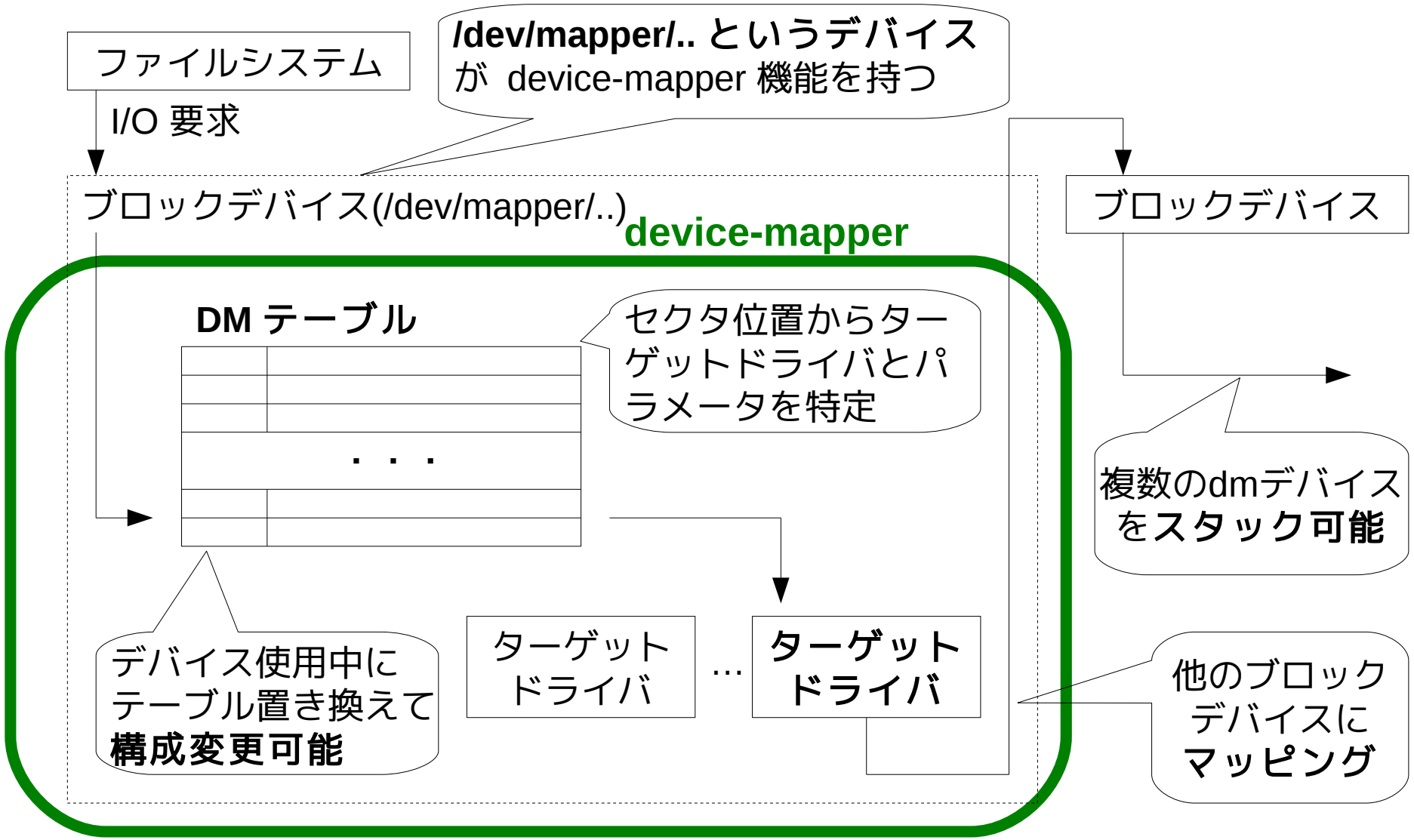
テストやデバッグで便利なターゲット

- zero: 読み込んだブロックの中身は全て 0
- error: 全ての読み書きが I/O エラー
- delay: 一定時間の遅延後、下位のブロックデバイスにI/O発行

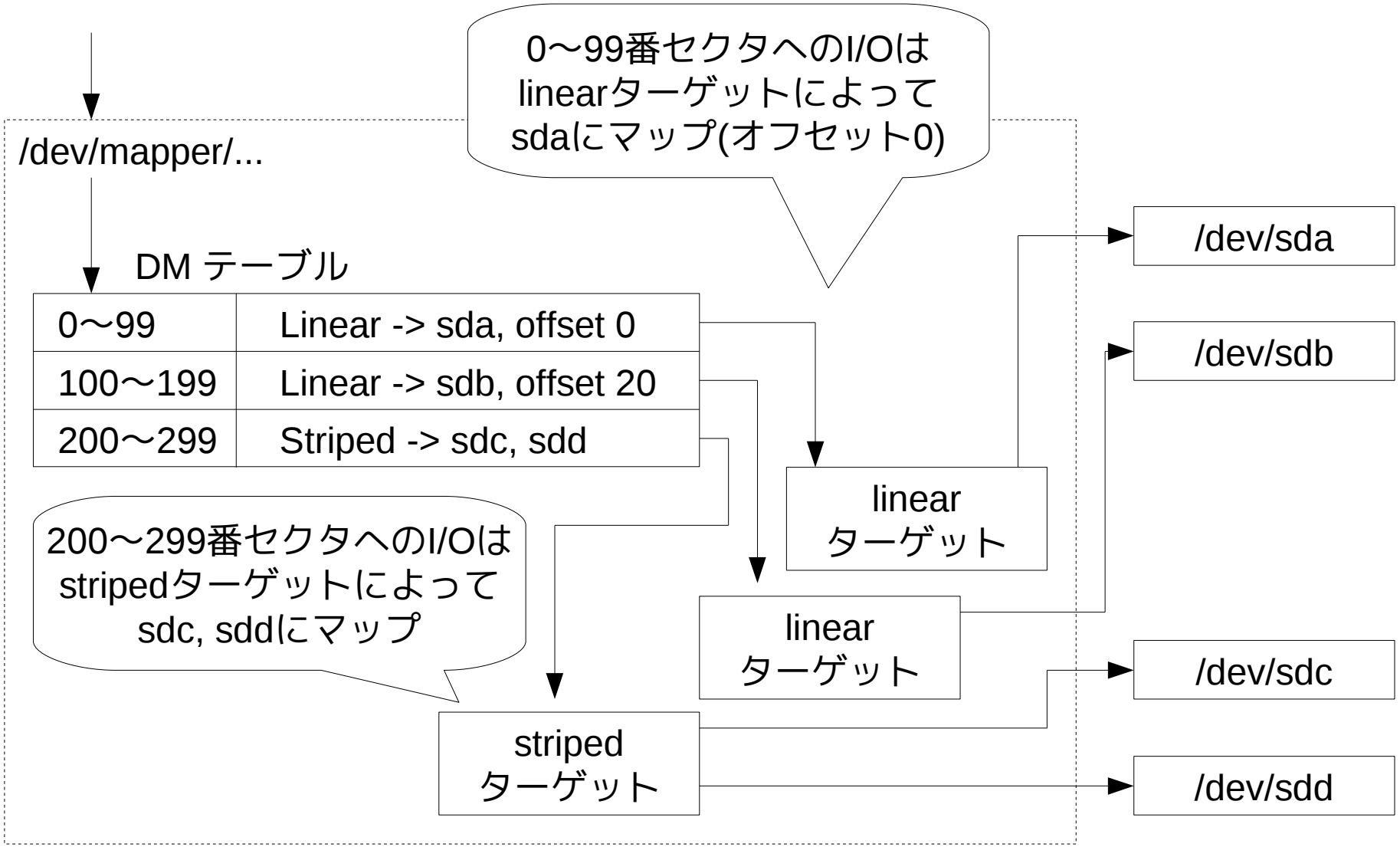


device-mapper の概念と構造

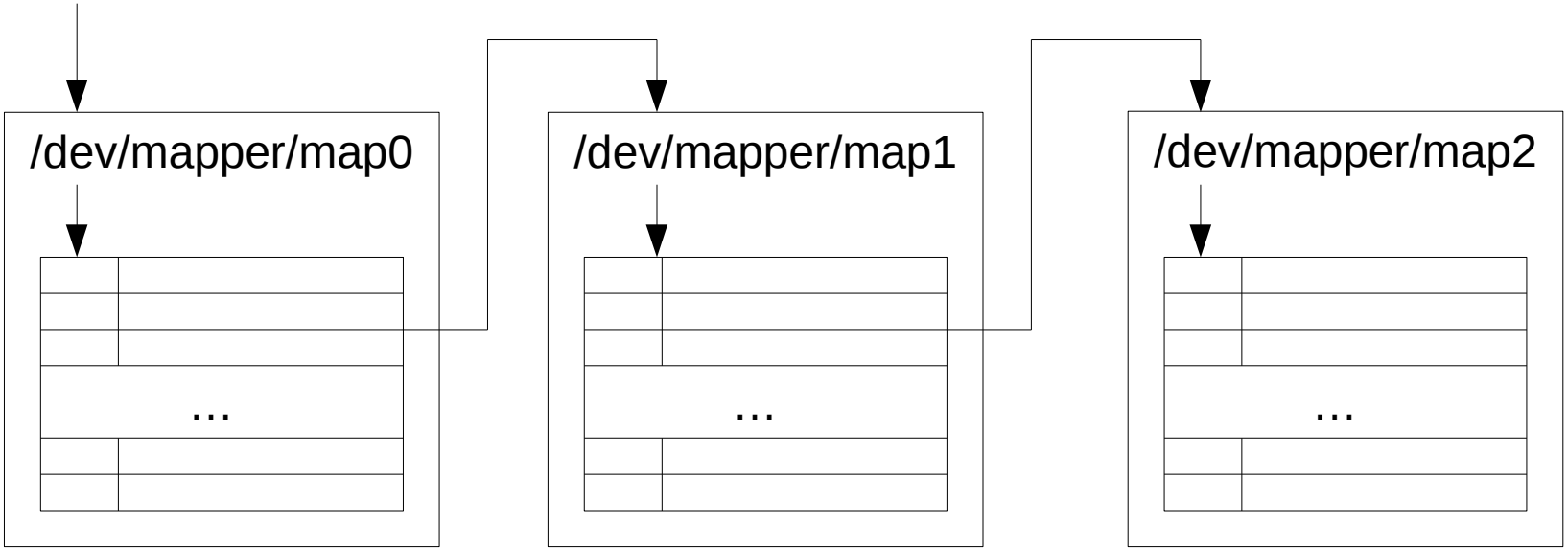
device-mapper の構成概念と構造



device-mapper のマッピング例

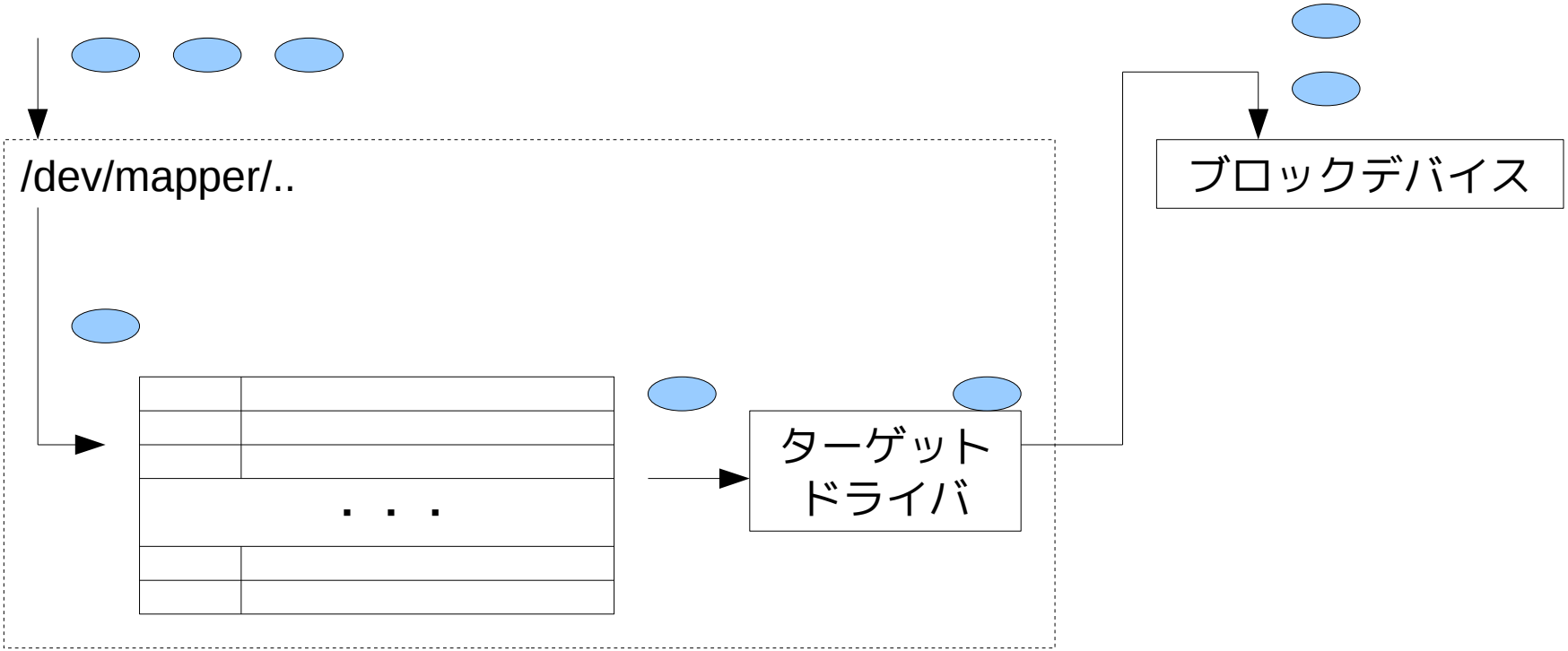


device-mapper のスタッキング



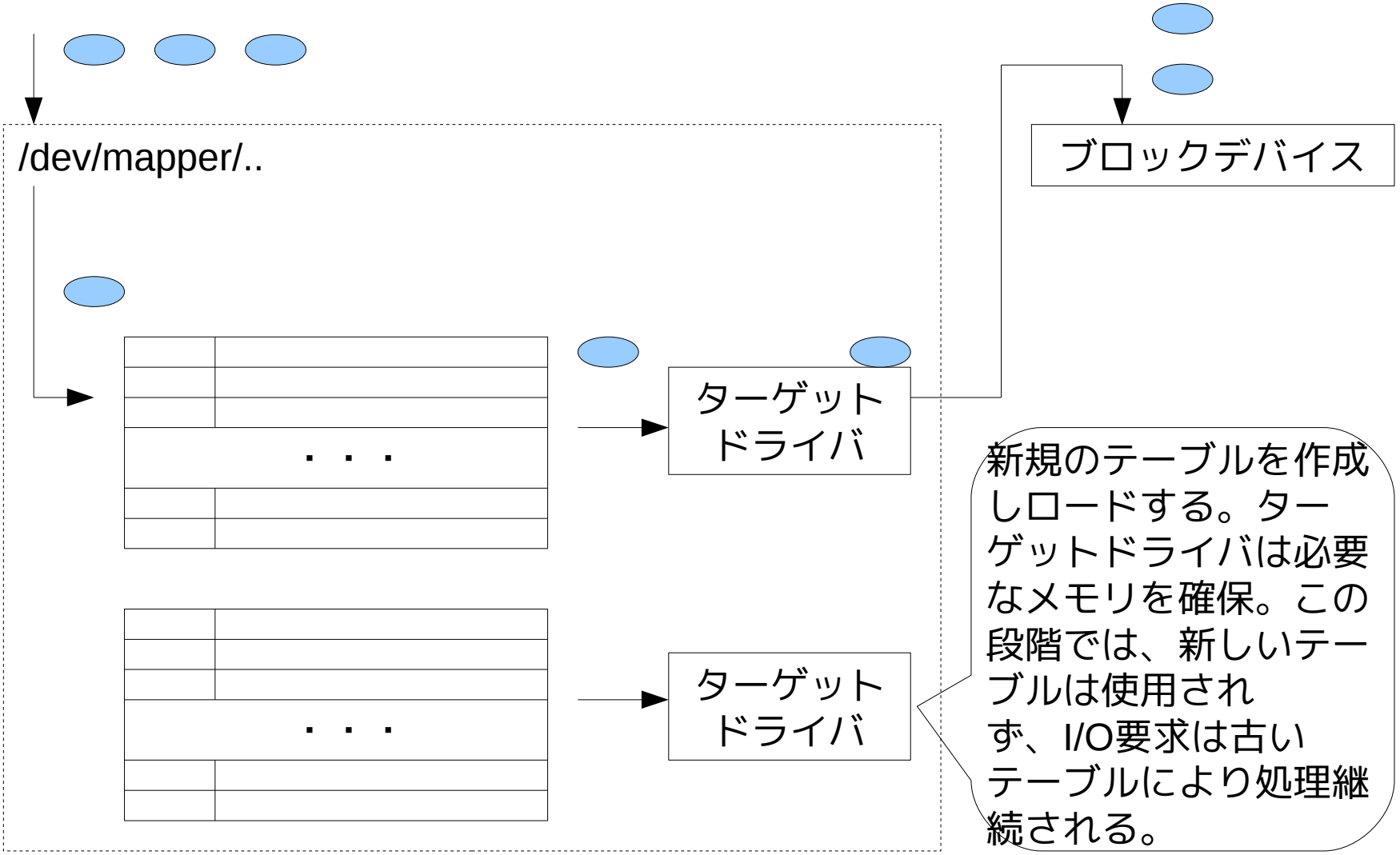
- device-mapper デバイス (/dev/mapper/..) 自身もブロックデバイスなので、各種のマップをスタックすることが可能

device-mapper の構成変更(1)

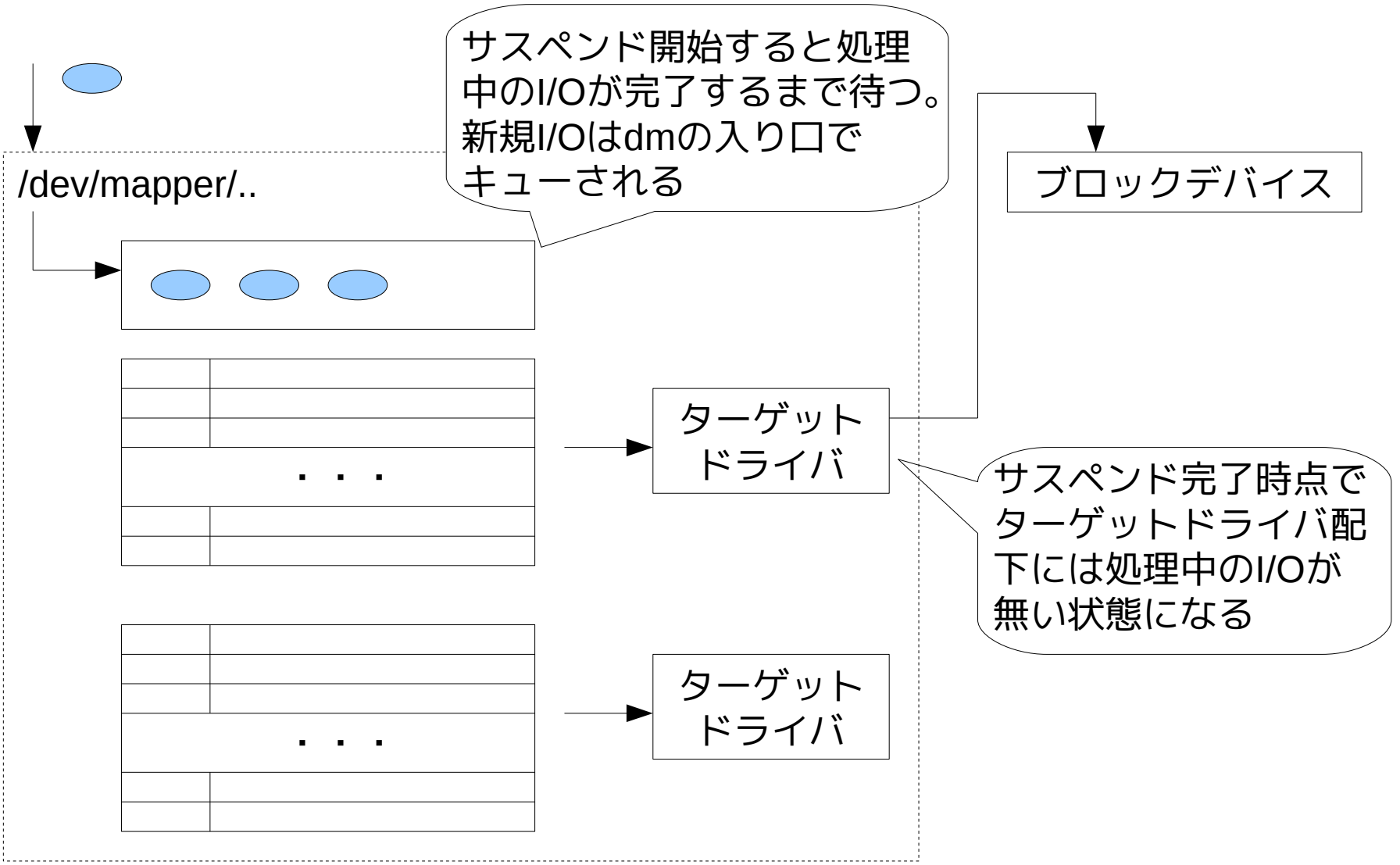


- device-mapper の特徴の一つは、ブロックデバイスを使用中に構成変更できること。上図中、●が各コンポーネントで処理されているI/O要求とした場合、どのように構成変更を行なうか、内部の動作を見てみる。

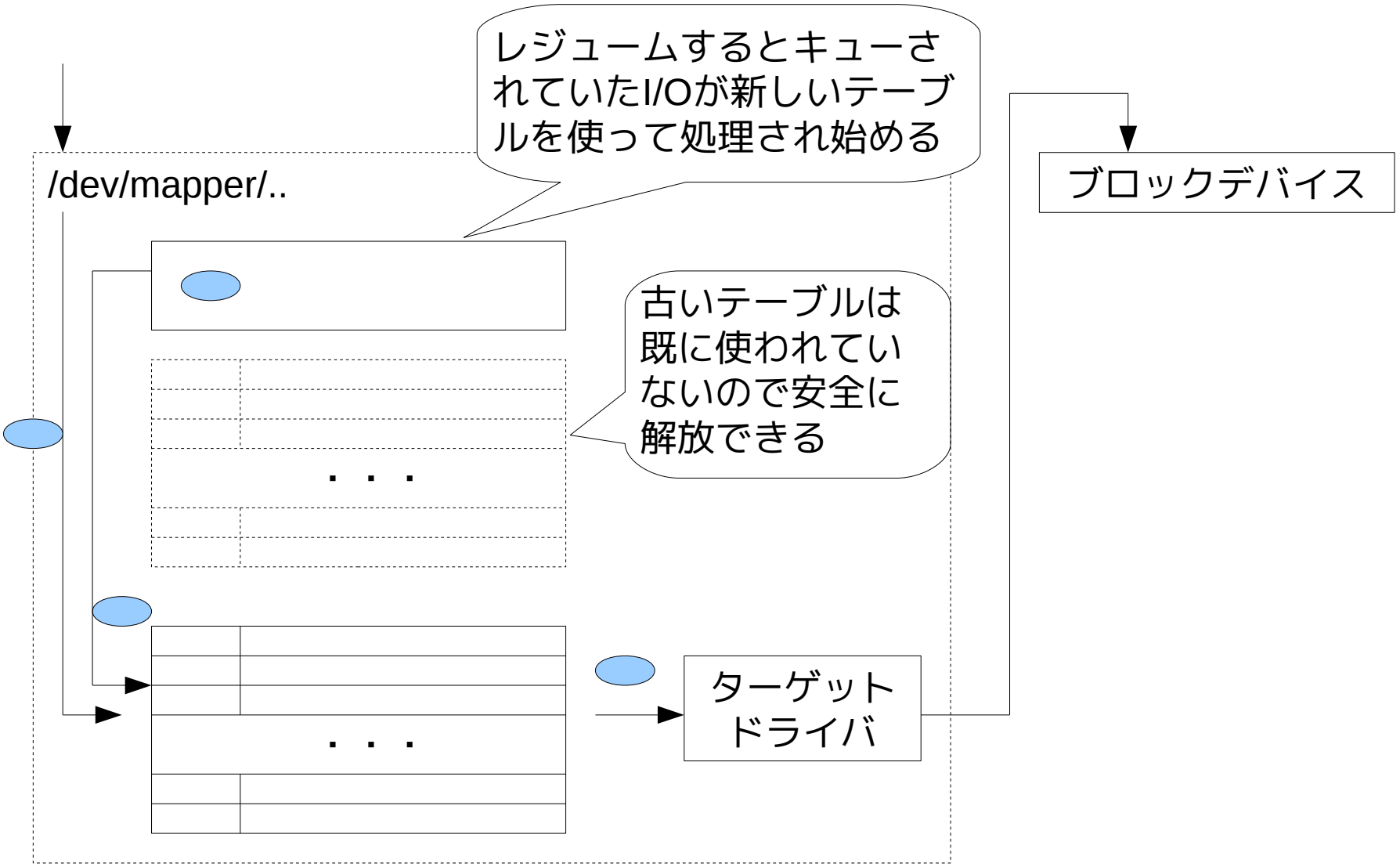
device-mapper の構成変更(2) テーブルのロード



device-mapper の構成変更(3) サスペンド



device-mapper の構成変更(4) レジューム



device-mapper を使う

device-mapper の基本

- デバイスファイル名: /dev/mapper と /dev/dm-*
  - /dev/mapper/<マップ名> を使うことが推奨される
  - /dev/dm-<番号> はカーネル内でのブロックデバイス共通の名前管理のための便宜的なもの。マップを作成する順番により変動するので、通常使うべきではない。
    - sysfs 上では dm-<番号> を参照せざるを得ないが、
      /sys/block/dm-<番号>/dm/name に <マップ名> が格納されている
  - DMアプリケーションによって他の名前(通常 /dev/mapper/.. へのシンボリックリンク)が用意されることもある
    - multipath: /dev/mpath/<デバイス名>
    - LVM2: /dev/<ボリュームグループ名>/<ボリューム名>

dmsetup

- device-mapper の API にほぼ1:1対応した操作を行なえる
  - 情報表示
    - dmsetup ls [--tree] : DMデバイス一覧(またはツリー表示)
    - dmsetup table: DMテーブルの表示
    - dmsetup info [-c]: DMデバイスの状態
  - 作成・構成変更・削除
    - dmsetup create <マップ名> : デバイス作成
    - echo “<DMテーブル>” | dmsetup load <マップ名> : テーブルのロード
    - dmsetup resume <マップ名>: ロードしたテーブルに置き換え
    - dmsetup remove <マップ名>: デバイス削除

  ただし、dmsetupを直接使うのは主にデバッグやテストなど
- 通常は各DMアプリケーション (LVM2 や multipath-tools) を使う
  - 作成から使用まで (DMアプリケーション間でほぼ共通)
    - 論理デバイス作成 (アプリケーション固有の構成管理情報作成)
    - 論理デバイスの activation (DMデバイス作成)
    - 以降、状態表示や deactivation、論理デバイス削除は随時

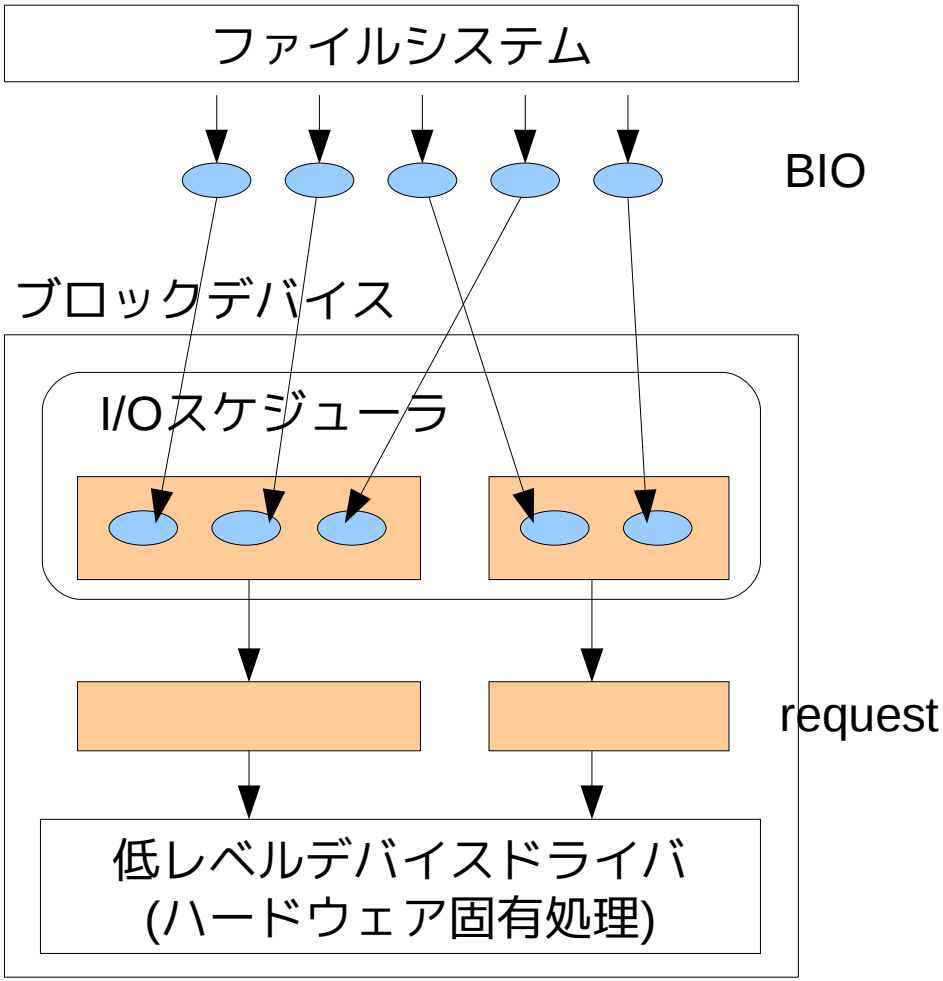
DMアプリケーション

各DMアプリケーションの使い方概要

	LVM2	multipath-tools	dmraid	cryptsetup
新規作成	lvcreate	multipath.conf に装置情報を記載	装置専用のBIOSユーティリティなどでRAIDメタデータ作成	cryptsetup luksFormat
Activation	lvchange -ay	multipathd 起動	dmraid -ay	cryptsetup luksOpen
Deactivation	lvchange -an	multipath -f	dmraid -an	cryptsetup luksClose
構成変更	lvresize, lvconvert, pvmove	multipath.conf の設定変更	装置専用のBIOSユーティリティなどから行なう	cryptsetup luksAddKey など
削除	lvremove	multipath.conf でブラックリスト指定	dmraid -E	(特になし)
状態表示	lvs	multipath -ll	dmraid -c	cryptsetup luksDump

Request-based device-mapper の紹介

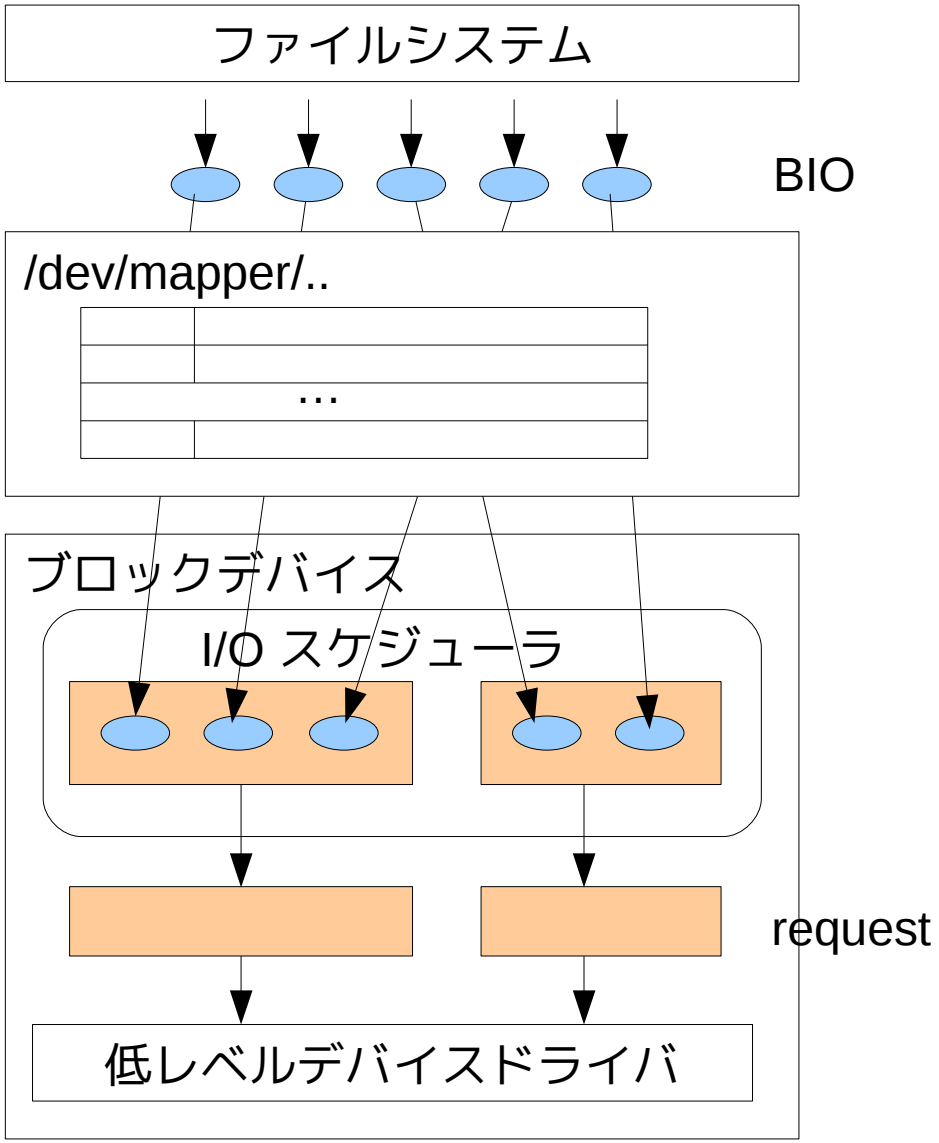
Linux ブロックデバイスの内部構造



- BIOはファイルシステムなどからブロックデバイスに対するI/O要求単位
- ブロックデバイス内部ではBIOをrequestという「箱」に詰め込む
- requestはI/Oスケジューラのキューに置かれ、対象ディスク領域が連続しているBIOは一つのrequestにまとめられる
- 低レベルデバイスドライバには、requestがI/O単位として発行される

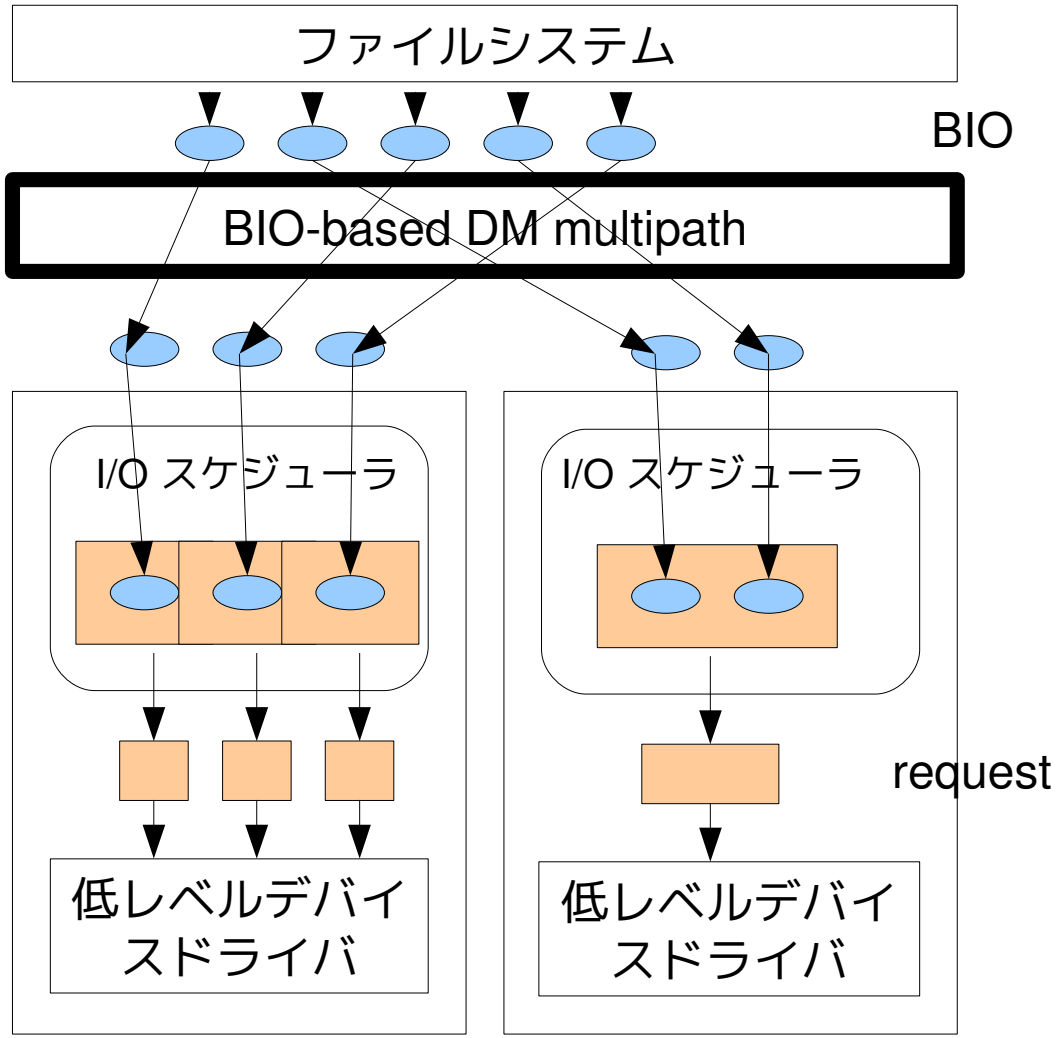
従来の BIO-based device-mapper の位置づけ

- I/O スケジューラより上の層に位置づけられ、BIO が処理対象



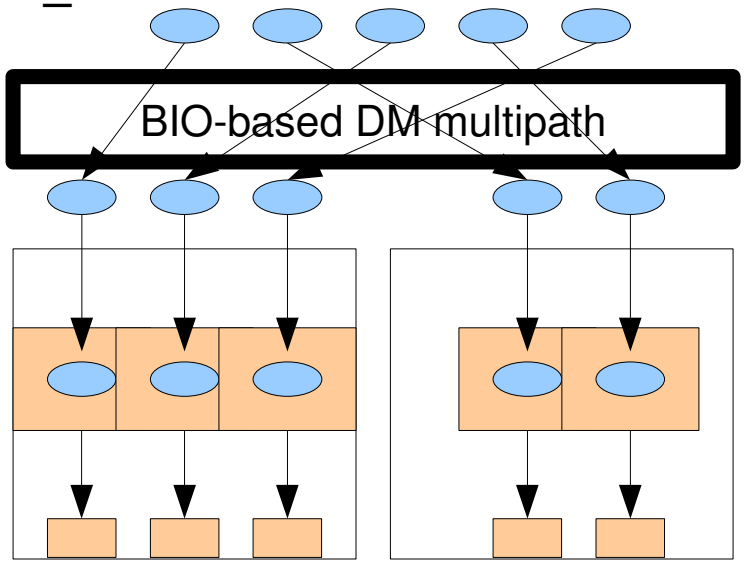
BIO-based device-mapper における multipath の動作

- BIO何個毎にパス切り替えの判断を行なうかをパラメータ(rr_min_io)で指定する
- 実際に低レベルデバイスドライバに発行されるI/Oは、request単位なので、I/Oスケジューラの動作によっては期待した負荷分散にならない

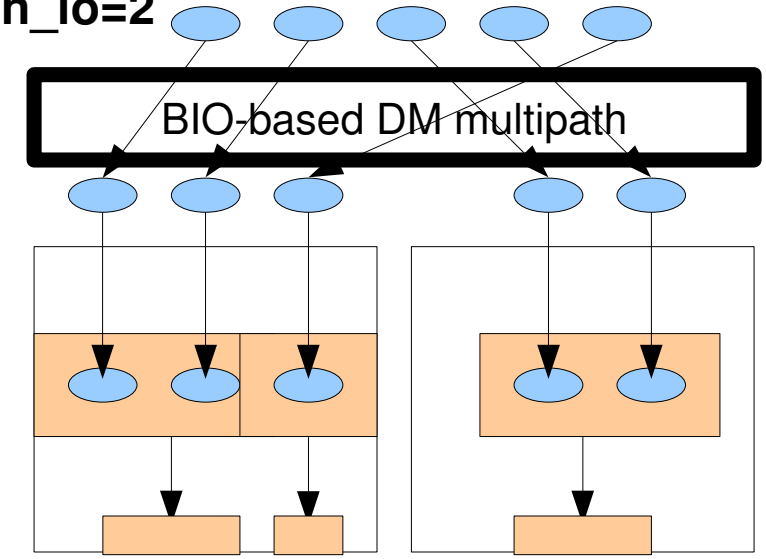


パス切り替え間隔(rr_min_io)による動作の違い

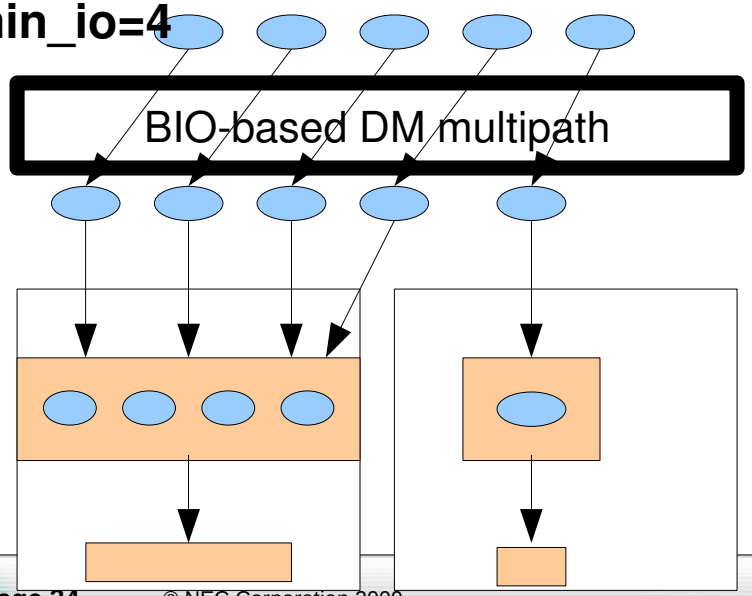
rr_min_io=1



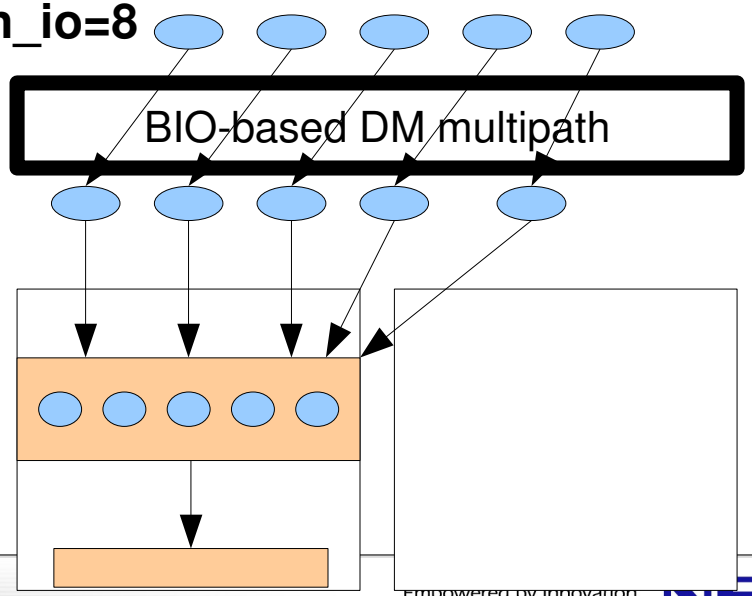
rr_min_io=2



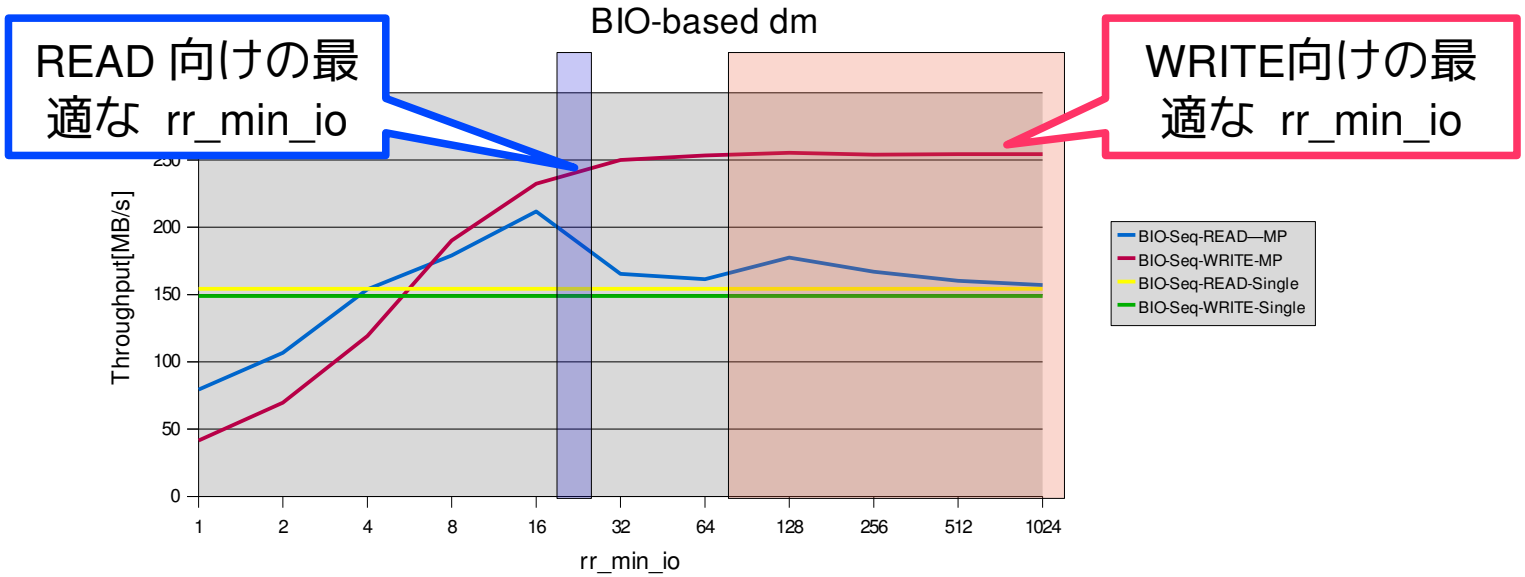
rr_min_io=4



rr_min_io=8



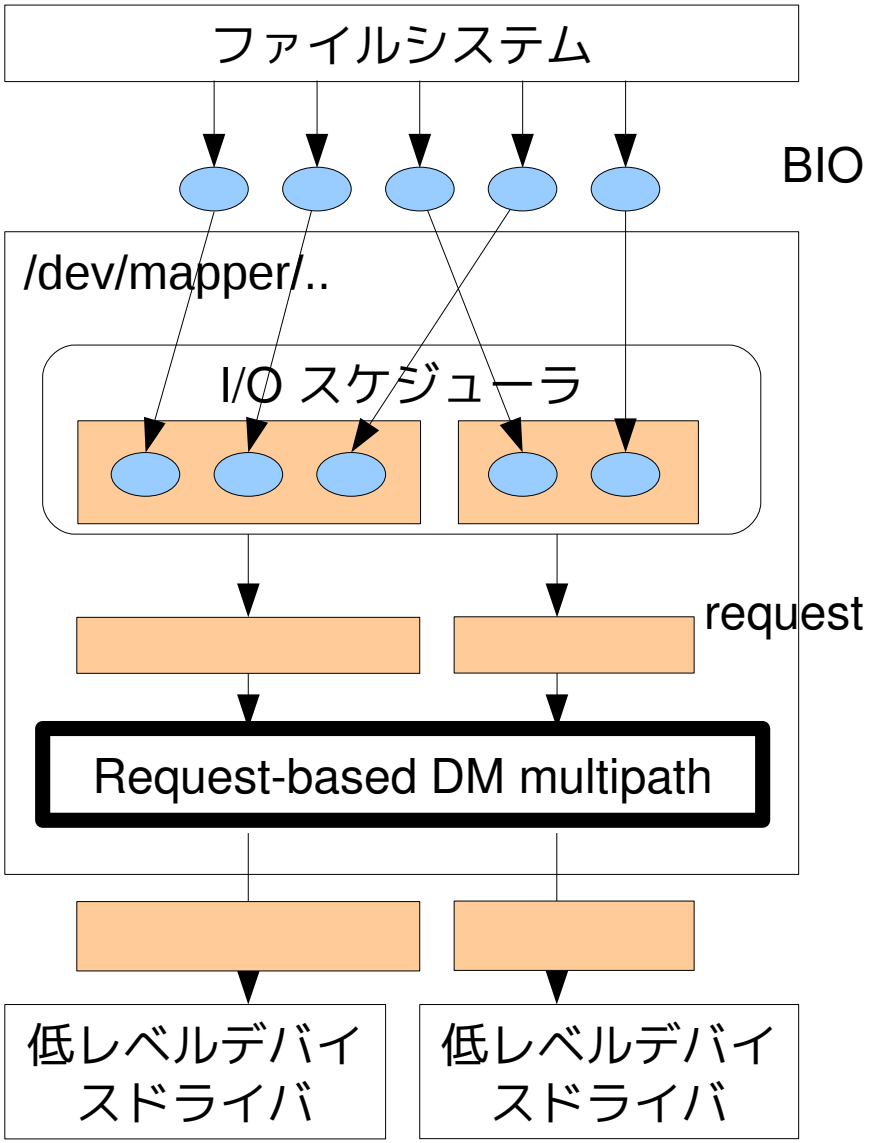
BIO-based multipath での問題点



- I/O パターンを予想して最適なパス切り替え間隔を見つける必要あり
  - WRITE はある程度のまとまった量の要求として出される場面が多く、I/Oスケジューラでマージされる可能性も高い。その場合には、rr_min_io は大きめの方が良い。
  - READ は応答性が要求されるため細かなサイズでマージされずに出される場面が多く、rr_min_io は小さい方が良い。

request-based device-mapper

- I/O スケジューラより下の層にも device-mapper の機能を追加
- request を単位として機能するため、multipath の負荷分散で効果を発揮
- 2.6.31 カーネルより使用可能
- BIO-based か request-based かはターゲットによって使い分け (現状、multipath のみ request-based)



さいごに

device-mapper に関する情報

- Device-mapper Resource Page
  - http://sources.redhat.com/dm/
- dm-devel メーリングリスト
  - device-mapper のカーネル側機能に関する議論、device-mapper multipath に関する議論
  - https://www.redhat.com/mailman/listinfo/dm-devel
- lvm-devel メーリングリスト
  - device-mapper のユーザスペースライブラリに関する議論はこちらでも行なわれることあり
  - https://www.redhat.com/mailman/listinfo/lvm-devel

Empowered by Innovation