

2007年度

オープンソースソフトウェア活用基盤整備事業

第I期 テーマ型（調査）

「Linux ディスク冗長化機能の適用評価と  
最適な適用方法の調査」

— ソフトウェア RAID 評価手順書 —

2008年 1月

独立行政法人 情報処理推進機構

# 目次

<b>1</b>	<b>はじめに</b> .....	<b>1</b>
1.1	目的.....	1
1.2	実施評価概要.....	1
1.2.1	機能評価.....	1
1.2.2	性能評価.....	1
1.2.3	品質評価.....	1
<b>2</b>	<b>機能評価実施手順詳細</b> .....	<b>2</b>
2.1	機能評価環境.....	2
2.2	機能評価テストプログラムインストール手順.....	2
2.3	機能評価実施手順.....	2
<b>3</b>	<b>性能評価実施手順詳細</b> .....	<b>4</b>
3.1	性能評価環境.....	4
3.2	性能評価テストプログラムインストール手順.....	4
3.3	性能評価実施手順.....	4
3.3.1	性能測定.....	4
3.3.2	復旧処理時間測定.....	5
<b>4</b>	<b>品質評価実施手順詳細</b> .....	<b>6</b>
4.1	品質評価環境.....	6
4.2	品質評価ツールインストール手順.....	6
4.2.1	動作環境.....	6
4.2.2	故障模擬ライブラリのインストール手順.....	6
4.3	品質評価実施手順.....	7
4.3.1	品質評価テストプログラムを用いた品質評価実施手順.....	8
4.3.2	故障模擬ライブラリを直接使用する場合の評価手順.....	10
4.4	品質評価手順（故障ディスク交換評価）.....	13
4.4.1	故障ディスク交換評価環境の構築手順.....	13
4.4.2	故障ディスク交換評価実施手順.....	13
4.4.3	故障ディスク交換評価結果確認手順.....	14
4.5	テストプログラムのマニュアル.....	15
4.5.1	品質評価テストプログラム.....	15
4.5.2	故障模擬ライブラリ.....	16

# 1 はじめに

## 1.1 目的

本文書は「Linux ディスク冗長化機能の適用評価と最適な適用方法の調査」に於ける機能評価、性能評価、品質評価で実際に実施した評価手順について説明する。

## 1.2 実施評価概要

本件で実施したソフトウェア RAID 評価は、大別すると「機能評価」、「性能評価」、「品質評価」がある。本節ではこれら評価種別毎に概要を説明する。次節以降でそれぞれについて、評価環境および手順について詳細を説明する。なお、評価結果については「調査報告書」を参照のこと。

### 1.2.1 機能評価

Linux のソフトウェア RAID 機能について最適な適用方法を調査するために、md (Multiple Devices)および DM (DeviceMapper)において、設定・復旧・管理のために必要な機能がサポートされているか調査し、それらの機能が正常に動作するか評価により検証した。サポート機能の調査結果は、「ソフトウェア RAID 設定手順書」を参照のこと。

### 1.2.2 性能評価

Linux ソフトウェア RAID を様々な状態で性能を検証するため、以下の項目について、性能測定を行う。

- ・ 通常運用時の RAID に対する I/O 性能について RAID 構築なしと比較する
- ・ 縮退状態での I/O 性能について通常運用時と比較する
- ・ 復旧処理中の I/O 性能について通常運用時と比較する
- ・ 復旧処理に要する時間を測定する

性能評価では、今回の調査で開発した性能評価用テストプログラムを使用する。このプログラムは、シーケンシャルに読み取り/書き込み実行時間を測定し、その結果から性能を計算する。評価用ファイルの先頭から末尾まで複数パターンの I/O サイズで read/write した場合の I/O 性能(MB/sec)の平均値を測定するものである。

### 1.2.3 品質評価

Linux のソフトウェア RAID 機能である md と DM に関して、通常運用時、縮退状態、復旧処理中のそれぞれの場合に模擬故障を発生させ、エラー処理部分の評価を行った。また故障ディスクの交換を行う場合を想定して、hotplug を実際に行い、デバイスドライバまで含めたシステムとして正しく動作するか評価を行った。

品質評価では模擬故障を発生させるために故障模擬ライブラリと、これを利用したテストプログラムを使用した。このプログラムは、故障模擬ライブラリで実装されている模擬ディスク故障を各 RAID レベルに対して発生させ状態を確認するものである。評価の内容は以下の通りである。

#### ①スベアディスク付き通常運用時の故障評価

スベアディスク有りの状態でアクティブディスク一台に模擬故障を発生させた場合に、故障が検出できて、RAID アレイとシステムの運用に問題がないことを確認する。

#### ②スベアディスクなし通常運用時の故障評価

スベアディスク無しの状態で、アクティブディスク一台に模擬故障を発生させた場合に故障が検出できて、RAID アレイとシステムの運用に問題がないことを確認する。

#### ③縮退状態での故障評価

縮退状態で、アクティブディスク一台に模擬故障を発生させた場合に故障が検出できて、RAID アレイは運用できなくなるが、システムの運用に問題がないことを確認する。

#### ④スベアディスク付き復旧処理中の故障評価

スベアディスクで復旧処理が動作している状態でスベアディスク一台に模擬故障を発生させた場合に、故障が検出できて、RAID アレイとシステムの運用に問題がないことを確認する。

#### ⑤復旧処理中アクティブディスクでの故障評価

スベアディスクで復旧処理が動作している状態でアクティブディスク一台に模擬故障を発生させた場合に、故障が検出できて、RAID アレイは運用できなくなるが、システムの運用に問題がないことを確認する。

## 2 機能評価実施手順詳細

本章では ソフトウェア RAID 機能評価について実施環境と実施手順について説明する。

Linux のソフトウェア RAID 機能について最適な適用方法を調査するために、MD (Multiple Devices)、DeviceMapper に設定・復旧・管理のための必要な機能がサポートされているか調査し、それらの機能が正常に動作するか評価により検証した。サポート機能の調査結果は、「ソフトウェア RAID 設定手順書」参照。

機能評価項目は以下の 3 種類に大別される。

- ・ 設定手順
- ・ 復旧手順
- ・ 管理手順

各項目に対して、以下の環境を使用して、それぞれの機能が正常に動作しているか確認した。また、RAID アレイ上で一般的なファイル操作関連のシステムコールが正常に動作することを確認するシステムコールテストプログラムを開発し評価した。

以下ではシステムコールテストプログラムを利用したシステムコール動作確認について、評価実施手順について説明する。設定手順評価、復旧手順評価、管理手順評価については調査報告書を参照のこと。

### 2.1 機能評価環境

表 1 は、ディスク種別とソフトウェア RAID 種別ごとの評価環境を一覧にしたものである。

表 1 機能評価環境

		ソフトウェア RAID 種別	
		MD	Device Mapper
ディスク種別	SATA デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	Intel Matrix Storage Manager を用いて RAID1 で dmraid 構築と、LVM2 の RAID アレイ構築機能を用いて RAID1 構築
	SCSI デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	LVM2 の RAID アレイ構築機能を用いて RAID1 構築

評価環境の詳細については「調査報告書を参照」のこと。

### 2.2 機能評価テストプログラムインストール手順

ソースに同梱する Makefile にて make を行う。

評価用ファイルシステムを構築、マウント後、マウントしたファイルシステムの任意のディレクトリにコピーし実行する。

### 2.3 機能評価実施手順

システムコールテストプログラム `syscall_test` は、引数で動的に作成されるテストファイルの大きさを指定して、`open` システムコールのフラグを変えながら、`open`、`write`、`lseek`、`read`、`fstat`、`close` の各システムコールが正しく動作するかを確認する。

以下に実行例を示す。

(例) テストファイルのサイズに 256 キロバイトを指定して実行する場合

```
# ./syscall_test 256k
```

本プログラムは、何かエラーを検出した場合のみ、その旨を出力する。正常時は何も出力は無い。エラーを検出した場合の出力内容は以下の通りである。システムコールが失敗した場合、システムエラーメッセージを出力するが、「`~:<flags>:file_size=<size>:`」各システムエラーメッセージ」の書式で出力する。

`<flags>` は、`NONE/O_DIRECT/O_SYNC` のいずれかである。`<size>` は、`54/user_size` のいずれかであり、

`user_size` は、コマンド引数で指定されたファイルサイズである。

～部分の各エラーメッセージの説明は、以下の通りである。

- `open failed` …ファイルのオープン失敗
- `stat failed` …ファイルの新規作成失敗
- `write failed` …ファイルの書き込み失敗
- `random_open failed` …ファイルに書き込む/`dev/random` のオープン失敗
- `random_read failed` …ファイルに書き込む/`dev/random` からの読み込み失敗
- `random_close failed` …ファイルに書き込む/`dev/random` のクローズ失敗
- `lseek failed` …ファイルのオフセット位置変更失敗
- `read failed` …ファイルからの読み込み失敗
- `fstat failed` …ファイルの状態が不正
- `close failed` …ファイルのクローズ失敗
- `unlink failed` …ファイルの削除失敗

例えば、`open` のフラグが `O_DIRECT` で、小さいファイルサイズ(54 バイト)のファイルに対しての `open` 処理でエラーが発生した場合、下記のように出力される。

```
open failed:O_DIRECT:file_size=54: No such file or directory
```

上記以外のエラーメッセージは、「～:<flags>:file\_size=<size>」の書式で出力する。

<flags>は `NONE/O_DIRECT/O_SYNC` のいずれかである。

<size>は `54/user_size` のいずれかである。`user_size` はコマンド引数で指定されたファイルサイズである。

～部分の各エラーメッセージの説明は、以下の通りである。

- `invalid write size` …ファイルに書き込んだサイズが不正
- `invalid random_read size` …ファイルに書き込む/`dev/random` から読み込んだサイズ不正
- `invalid write_memcmp` …ファイルに書き込んだ内容が不正
- `invalid read size` …ファイルから読み込んだサイズ不正
- `invalid read_memcmp` …ファイルから読み込んだ内容が不正
- `invalid fstat_mode` …ファイルのアクセス保護が不正
- `invalid fstat_nlink` …ファイルのハードリンク数が不正
- `invalid fstat_uid` …ファイルの所有者のユーザ ID が不正
- `invalid fstat_size` …ファイルのサイズが不正

例えば、`open` のフラグが `O_SYNC` で、小さいファイルサイズ(54 バイト)のファイルに対しての `write` 処理でサイズ不正のエラーが発生した場合、下記のように出力される。

```
invalid write size:O_SYNC:file_size=54
```

### 3 性能評価実施手順詳細

本章では ソフトウェア RAID 性能評価について実施環境と実施手順について説明する。

性能評価は以下の 3 種類に大別される。

- ①通常運用時:RAID が正常に運用している状態
  - ・各 RAID レベルの通常運用時に RAID の性能を測定する。
- ②縮退状態: RAID からディスクの 1 つを切り離れた状態
  - ・各 RAID レベルの縮退状態での RAID の性能を測定する。
- ③復旧処理中: 縮退している RAID をスペアディスクで復旧処理中の状態
  - ・各 RAID レベルの復旧処理中に RAID の性能を測定する。
  - ・無負荷状態での各 RAID レベルの復旧時間を測定する。
  - ・負荷状態での各 RAID レベルの復旧時間を測定する。

以下に性能評価テストプログラムを利用した性能評価実施手順を説明する。なお、RAID アレイの個々の操作の詳細については、「ソフトウェア RAID 設定手順書」を参照。

#### 3.1 性能評価環境

それぞれの運用種別について、表 2 に示す性能評価環境を使用して評価を実施した。

表 2 性能評価環境

		ソフトウェア	
		MD (Multiple Devices)	Device Mapper
ディスク種別	SATA デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	LVM2 の RAID 構築機能を用いて RAID1 構築
	SCSI デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	LVM2 の RAID 構築機能を用いて RAID1 構築

評価環境の詳細については「調査報告書」を参照のこと。

#### 3.2 性能評価テストプログラムインストール手順

ソースに同梱する Makefile にて make を行う。

評価用ファイルシステムを構築、マウント後、マウントしたファイルシステムの任意のディレクトリにコピーし実行する。

#### 3.3 性能評価実施手順

3.1 項の環境を使用して、通常運用時、縮退状態、復旧処理中の性能評価、および復旧処理時間測定を実施するための具体的な評価方法を説明する。なお、各 RAID を通常運用時、縮退状態、復旧処理中の構成にするための個々の操作の詳細については、「ソフトウェア RAID 設定手順書」を参照。

##### 3.3.1 性能測定

性能測定では、性能評価用テストプログラムを使用して、通常運用時、縮退状態、復旧処理中のそれぞれの状態でブロックデバイスおよびファイルシステムのファイルに対してシーケンシャルに read/write を行い、read/write の性能(Mbyte/sec)の測定を実施する。

ファイルのオープンモードに依って以下の 3 通りを実施する。

指定なしの場合:

```
# ./srwbench -f testfile -s 64 -e 262144 -m 512
```

ダイレクト I/O の場合:

```
# ./srwbench -f testfile -s 64 -e 262144 -m 512 -i
```

同期 I/O の場合:

```
# ./srwbench -f testfile -s 64 -e 262144 -m 512 -y
```

性能評価テストプログラムによる測定の結果は、ツールが整形して表示する。表示内容は、測定 I/O サイズに於ける 1 秒あたりの read 速度(MB/sec)、1 秒当たりの write 速度(MB/sec)である。以下に実行例を示す。

(例)

```
# ./srwbench -f Testfile1 -s 128 -e 16384 -m 16
```

I-O size	READ (MB/sec)	WRITE (MB/sec)
128Kbyte	388.538125	45.276783
256Kbyte	410.077658	57.726722
512Kbyte	409.217627	57.293662
1Mbyte	407.851134	57.260036
2Mbyte	411.882819	57.434130
4Mbyte	396.196513	57.612606
8Mbyte	416.189783	57.103904
16Mbyte	408.966593	57.069279

### 3.3.2 復旧処理時間測定

復旧処理時間測定では、スペアディスクを含む RAID を構築し、RAID ディスクのひとつに不良マークを付与して復旧処理中の状態にし、復旧処理が完了する時間の測定を実施する。  
無負荷時、負荷時の 2 パターンの測定を実施する。

## 4 品質評価実施手順詳細

本章では ソフトウェア RAID 性能評価について実施環境と実施手順について説明する。  
品質評価は以下の 4 種類に大別される。

- ・ 通常運用時の故障
- ・ 縮退状態の故障
- ・ 復旧処理中の故障
- ・ 故障ディスク交換

以下に性能評価テストプログラムを利用した品質評価実施手順を説明する。

なお、RAID アレイに対する個々の操作の詳細については、「ソフトウェア RAID 設定手順書」を参照。

### 4.1 品質評価環境

表 3 は、ソフトウェア RAID 種別ごとの評価環境を一覧にしたものである。

表 3 品質評価環境

		ソフトウェア	
		MD (Multiple Devices)	Device Mapper
ディスク種別	SATA デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	LVM2 の RAID 構築機能を用いて RAID1 構築
	SCSI デバイス	mdadm を用いて RAID1、RAID5、RAID6、RAID10 構築	LVM2 の RAID 構築機能を用いて RAID1 構築

評価環境の詳細については「調査報告書」を参照のこと。

本評価では、RAID アレイに対して模擬故障を発生させるため、使用した装置が実際にハードウェア的な問題を引き起こすわけではない。しかし、Linux ドライバやカーネルの観点では故障が起きた際の動作をすることになり、ソフトウェア的に RAID アレイの内容が破損する恐れがある。従って、本評価は評価用に用意した RAID アレイ環境上で実施するべきである。

### 4.2 品質評価ツールインストール手順

#### 4.2.1 動作環境

品質評価ツールは下記 OS 環境で動作する。

- ・ Red Hat Enterprise Linux 4 Update 5 (カーネルバージョン:2.6.9-55.ELsmp)
- ・ Red Hat Enterprise Linux 4 Update 5 上で動作するコミュニティカーネル linux-2.6.22.6

#### 4.2.2 故障模擬ライブラリのインストール手順

最初に、故障模擬ライブラリの導入手順を説明する。故障模擬ライブラリは品質評価テストプログラムで模擬故障を発生させるために使用する。

##### (1) SystemTap の導入

本評価では Red Hat Enterprise Linux 4 Update 5 をベースとして利用するため、前提として、Red Hat Enterprise Linux 4 Update 5 をインストールする。また、インストール時に SystemTap パッケージを選択して OS インストールを実施する。その上で、以下の手順が必要となる。

##### A) Red Hat Enterprise Linux 4 Update 5 のカーネルを使用する場合

Red Hat Enterprise Linux 4 Update 5 に含まれるカーネルバージョン:2.6.9-55.ELsmp に対応する kernel-debuginfo パッケージを導入する。

##### B) Red Hat Enterprise Linux 4 Update 5 上で動作するコミュニティカーネル linux-2.6.22.6 を使用する場合

- ・ Linux2.6.22.6 カーネルのコンパイルする。

Red Hat Enterprise Linux 4 Update 5 上で一般的なカーネルコンパイルの手順に従って、カーネ

ルをコンパイルする。Systemtap を使う上で必要なカーネル設定は以下の通りである。なお、カーネルバージョン差分が大きいため、本件では Red Hat Enterprise Linux 4 Update 5 の.config ファイルをベースとせず、Fedora7 に含まれる.config ファイルを利用してカーネルコンパイルを実施した。

Loadable module support

Enable loadable module support [Y]

Module unloading [Y]

General setup

Kernel->user space relay support [Y]

Instrumentation Support

Kprobes [Y]

Kernel hacking

Kernel debugging [Y]

Compile the kernel with debug info [Y]

- ・ SystemTap の検索パスにデバッグカーネルへの symlink を追加する。

```
# mkdir -p /usr/lib/debug/lib/modules/2.6.22.6/
```

```
# ln -s /lib/modules/2.6.22.6/kernel /usr/lib/debug/lib/modules/2.6.22.6/
```

```
# ln -s /lib/modules/2.6.22.6/build/vmlinux /usr/lib/debug/lib/modules/2.6.22.6/
```

## (2) 品質評価テストプログラムの展開

品質評価テストプログラム一式を任意のディレクトリに展開する。

## (3) 品質評価テストプログラムのコンパイル

ソースに同梱する Makefile にて make を行う。

## (4) 故障模擬ライブラリの展開

故障模擬ライブラリを品質評価テストプログラムと同じディレクトリに展開する。使用する OS 種別、RAID 種別に従って、適切な故障模擬ライブラリを選択する必要がある。

### A) Red Hat Enterprise Linux 4 Update 5 のカーネルを使用する場合

rhel4 配下のもを使用する。md RAID5/6 使用時には、rhel4/rhel4\_raid56 以下を展開する。md RAID5/6 以外を使用している場合には、rhel4/rhel4 以下を展開する。

### B) Red Hat Enterprise Linux 4 Update 5 上で動作するコミュニティカーネル linux-2.6.22.6 を使用する場合

upstream 配下のもを使用する。さらに、RAID 種別、ディスク種別に従って適切なものを選択する必要がある。upstream 配下に以下の 4 つのサブディレクトリが存在するが、使い分けは以下の通りとなる。

upstream\_sata: SATA ディスクで md RAID5/6 以外を使用する場合

upstream\_scsi: SCSI ディスクで md RAID5/6 以外を使用する場合

upstream\_scsi\_md\_raid56 : SATA ディスクで md RAID5/6 を使用する場合

upstream\_scsi\_md\_raid56 : SATA ディスクで md RAID5/6 を使用する場合

## 4.3 品質評価実施手順

以下に性能評価ツールを利用した品質評価実施手順を説明する。いずれも root 権限を必要とする。

以下に実際に実施した品質評価の手順を説明する。品質評価の方法には大きく分けて、品質評価テストプログラムを用いる方法と故障模擬ライブラリを直接用いる方法の二通りがある。品質評価テストプログラムは自動実行に適しているが、RAID の種類によっては故障を正しく発生できないパターンが存在する。そこで評価に当たっては両方の方法を組み合わせて評価を実施する。まず、品質評価テストプログラムを用いて評価を実施し、詳細を調べる必要のある項目については故障模擬ライブラリを直接用いて評価を行う。

### 4.3.1 品質評価テストプログラムを用いた品質評価実施手順

以下では、品質評価テストプログラムにより RAID アレイを評価するための手順を示す。本プログラムを用いて RAID アレイに対する評価を行うためには、5 台以上のディスクを用いて、各ディスク上に 8GB 以上の容量を持つパーティションを作成する。品質評価テストプログラムを実行する際には、これらのパーティションをプログラムの引数として指定する。

なお、以下の場合は品質評価テストプログラムでは正しく評価ができない可能性があるため、後述する故障模擬ライブラリを直接使用する方法を併用する必要がある。

- md RAID5/RAID6  
SystemTAP で仕掛けた故障模擬ライブラリによる模擬故障投入が成功しない場合がある
- DM (LVM2)  
故障発生後の RAID アレイの状態が、LVM2 の管理ツールでは正確に取得できない場合がある
- その他、バグやパニックが発生し自動実行が中断する場合

#### (1) 品質評価テストプログラムの実行

パーティション /dev/sdb1, /dev/sdc1, /dev/sdd1, /dev/sde1, /dev/sdf1 を用いて評価を実施する場合は、以下のコマンドにより評価対象とする RAID アレイの種類と発生させるタイプに相当する模擬故障発生スクリプトを選択して `redundancy_test.sh` を実行して完了を待つ。

```
# redundancy_test.sh -t <スクリプト名> -p /home/test -V VG1 -L LV1 -M /dev/md0 -I 5 -I 6 /dev/sd[bcdef]1
```

/home/test はテスト用に RAID アレイをマウントするディレクトリであり、テストプログラム内部でディレクトリを作成するため、存在しないディレクトリを指定する必要がある。<スクリプト名>は評価するディスク故障パターンに応じて、以下の中から選択する。

temporary_rerr.stp	read 単発エラー
sector_rerr.stp	write によって訂正可能な read エラー
disk_rwerr.stp	read/write 不可エラー
disk_rerr.stp	read 不可エラー
temporary_werr.stp	write 単発エラー
r_timeout.stp	read 単発無応答
w_timeout.stp	write 単発無応答
rw_timeout.stp	read/write 無応答

これにより、評価対象の RAID アレイに対する特定の種類の模擬故障について、アレイ構成を自動で変化させて通常運用時、縮退状態、復旧処理中の評価が一通り実施される。結果は、コンソールログ、syslog 出力から各状況における模擬故障発生でどのようなエラー処理が発生したか判断する。

#### (2) コンソール出力の確認

コンソールログで確認すべき点は、タイムスタンプ、使用する故障模擬ライブラリのスクリプト名、RAID 種別、開始時構成、故障模擬ライブラリによる故障発生通知、模擬故障投入の結果、終了時構成である。

このうち、評価項目はタイムスタンプ、使用する故障模擬ライブラリのスクリプト名、RAID 種別、開始時構成から判断する。また故障模擬ライブラリによる故障発生通知、模擬故障投入の結果、終了時構成により、評価結果がわかる。

コンソール出力は以下の内容の繰り返しとなっている。ツールは自動的に構成を変更して評価を連続実行しているが、各項目の区切りはタイムスタンプで判断する。

##### 1. タイムスタンプ

評価日時が評価項目毎に一度だけ表示される。ログの区切りを付けるために使用する。

(例) 2007 年 12 月 19 日 水曜日 16:22:47 JST

##### 2. 使用する故障模擬ライブラリのスクリプト

引数で与えたスクリプト名が実行開始時に表示される。"Target faultscript"の文字列で識別する。

(例) Target faultscript:[disk\_rwerr.stp]

##### 3. RAID 種別

md の場合、RAID 種別が mdadm -D コマンド出力中に含まれる。"Raid Level"の文字列で識別する。

(例) Raid Level : raid1

#### 4. 開始時構成

md の場合、評価開始時の構成が `mdadm -D` コマンド出力中に含まれる。識別は以下の方法で行う。

- ・ 通常運用時 (スペア有り)

”spare <デバイス名>”という文字列がある場合。

(例) `spare /dev/sde1`

- ・ 通常運用時 (スペア無し)

“active sync <デバイス名>”という文字列の後に”faulty ”や”spare”という文字列が無い場合。

(例)

```
Number Major Minor RaidDevice State
  0      8    17      0    active sync  /dev/sdb1
  1      8    33      1    active sync  /dev/sdb1
  1      8    33      1    active sync  /dev/sdc1
  2      8    49      2    active sync  /dev/sdd1
mke2fs 1.35 (28-Feb-2004)
```

- ・ 縮退状態

“faulty <デバイス名>”もしくは”removed”という文字列がある場合。

(例) `faulty /dev/sdb1`

- ・ 復旧処理中

”spare rebuilding <デバイス名>”という文字列がある場合。

(例) `spare rebuilding /dev/sdd1`

#### 5. 故障模擬ライブラリによる故障発生通知

模擬故障が発生した場合、故障模擬ライブラリが出力するメッセージにより、模擬故障が正しく投入できたか否かを確認する。

- ・ 応答のある場合ディスク故障パターン(read 単発エラー、write によって訂正可能な read エラー、read/write 不可エラー、read 不可エラー、write 単発エラー)の場合”scsi\_decide\_disposition”で始まる行が出力されていた場合、模擬故障が投入されたと判断できる。

(例) `scsi_decide_disposition : major=8 minor=32...`

- ・ 応答の無い場合ディスク故障パターン(read 単発無応答、write 単発無応答、read/write 無応答)の場合、”scsi\_add\_timer:”で始まる行が出力されていた場合、模擬故障が投入されたと判断できる。

(例) `scsi_add_timer: inode=...`

#### 6. 模擬故障投入の結果

縮退状態での模擬故障発生の場合では、故障発生の契機となった I/O が失敗する場合があります、その際にはエラーメッセージが表示されることがある。

- ・ read 失敗の場合、”Input/output error” というメッセージが表示される。

(例) `cat: xor.h: Input/output error`

- ・ write 失敗の場合、”journal commit I/O error” というメッセージが表示される場合があります。ここで、write 失敗の場合でも、この表示が無い可能性もあることに注意。

(例) `XXX kernel: journal commit I/O error`

- ・ I/O エラーにはならなかったものの、I/O の結果データ化けやデータ欠けが発生していた場合、`redundancy_test.sh` の中で実行される `cmp` コマンドが”differ:” という文字列を出力する。

(例) `test_data_before tast_data_after differ: char 1, line 1`

#### 7. 終了時構成

模擬故障発生、OS によるエラー処理が完了した後で、RAID の状態を判断する。開始時構成の”active sync <デバイス名>”の行と、終了時構成の”active sync <デバイス名>”の行を比較し、”active sync <デバイス名>”の行が減っていれば array が縮退したと判断できる。スペアあり通常運用時は”active sync <デバイス名>”の行が減っていなかった場合でも、デバイス名に違いがあれば、一旦縮退されてスペアが RAID アレイに組み込まれたことを意味する。

### (3) syslog 出力の確認

(2)で説明したコンソール出力のタイムスタンプと syslog に記録されているタイムスタンプを比較し、特定の評価項目における故障発生時の OS の動作を判断する。先に説明した通り、コンソールログでは各評価項目で最初に 1 回だけタイムスタンプを出力するため、タイムスタンプの間に発生した現象がその評価項目で発生した現象である。

- ・ read/write ともにエラーを SCSI より上位層が認識した場合、"end\_request: I/O error, <デバイス名>" が出力される。  
(例) end\_request: I/O error, dev sdc, sector 57407
- ・ ディスクを RAID アレイから切り離す場合、"<RAID 種別> Disk failure on <デバイス名> disabling device." が出力される。  
(例) raid1: Disk failure on sdc1, disabling device. Operation continuing on 3 devices
- ・ スペアディスクがある場合には"md: recovery of RAID array <デバイス名>"という出力により、切り離された後に自動的に復旧が開始されたことがわかる。  
(例) md: recovery of RAID array md0
- ・ read 単発エラーや write によって訂正可能な read エラーが発生した場合、write による修復を試みて成功した場合、"read error corrected"が出力される。(2.6.22.6 カーネルを使用している場合のみ)

#### 4.3.2 故障模擬ライブラリを直接使用する場合の評価手順

以下では故障模擬ライブラリを直接使用する場合の評価の実施方法を説明する。

以下で説明する故障模擬ライブラリを直接利用した評価方法は品質評価すべての項目について適用可能である。

ソフトウェア RAID 種別としては md の RAID1、RAID5、RAID6、RAID10、及び DM (LVM2)に対応している。故障種別は通常運用時故障、縮退状態での故障、復旧処理中の故障に対応している。また、ディスク故障パターンは read 単発エラー、write によって訂正可能な read エラー、read/write 不可エラー、read 不可エラー、write 単発エラー、read 単発無応答、write 単発無応答、read/write 無応答に対応している。

前節で説明した品質評価テストプログラムと異なり、故障模擬ライブラリを直接利用する方法では、一つずつの RAID 構成、RAID 状態と、故障タイプの組合せ毎に個別にテストを実施する必要がある。

以下の説明では、OS は sda にのみインストールされ、sdb、sdc、sdd、sde、sdf に評価用のワークディスクが接続された評価環境を用いるものとして説明する。ディスク種別は SCSI もしくは SATA とする。

sda		システムディスク
sdb		work disk1
	sdb1	work partition1 (4GB)
sdc		work disk2
	sdc1	work partition2 (4GB)
sdd		work disk3
	sdd1	work partition3 (4GB)
sde		work disk4
	sde1	work partition4 (4GB)
sdf		work disk5
	sdf1	work partition5 (4GB)

##### (1) 評価対象となる RAID アレイの構築

ソフトウェア RAID を以下の手順で構築する

- ・ RAID1 (スペアディスク有り)  
# mdadm -C /dev/md0 -l1 -n2 -x1 /dev/sd[bcd]1
- ・ RAID1 (スペアディスク無し)  
# mdadm -C /dev/md0 -l1 -n2 /dev/sd[bc]1
- ・ RAID5 (スペアディスク有り)  
# mdadm -C /dev/md0 -l5 -n3 -x1 /dev/sd[bcde]1
- ・ RAID5 (スペアディスク無し)  
# mdadm -C /dev/md0 -l5 -n3 /dev/sd[bcd]1
- ・ RAID6 (スペアディスク有り)  
# mdadm -C /dev/md0 -l6 -n4 -x1 /dev/sd[bcdef]1
- ・ RAID6 (スペアディスク無し)  
# mdadm -C /dev/md0 -l6 -n4 /dev/sd[bcde]1
- ・ RAID10 (スペアディスク有り)  
# mdadm -C /dev/md0 -l10 -n4 -x1 /dev/sd[bcdef]1
- ・ RAID10 (スペアディスク無し)  
# mdadm -C /dev/md0 -l10 -n4 /dev/sd[bcde]1

- ・ DM (LVM2)
 

```
# pvcreate/dev/sd[bcd]1
# vgcreate VG1 /dev/sd[bcd]1
# lvcreate -m 1 -L 2G VG1
```

(2) 評価するソフトウェア RAID 状態に応じて、以下の手順で RAID アレイの状態を変更する。

- ・ 通常運用時 (スペアディスク有り)
 

各 RAID レベルをスペアディスク有りの構成で構築した RAID アレイを用いる。
- ・ 通常運用時 (スペアディスク無し)
 

各 RAID レベルをスペアディスク無しの構成で構築した RAID アレイを用いる。
- ・ 縮退状態 (md の RAID1, RAID5, RAID10 の場合)
 

各 RAID レベルをスペアディスク無しの構成で構築した RAID アレイで、以下のコマンドを実行する。

```
# mdadm /dev/md0 -f /dev/sdb1
```
- ・ 縮退状態 (md の RAID6 の場合)
 

RAID6 をスペアディスク無しで構成した RAID アレイで、以下のコマンドを実行する。

```
# mdadm /dev/md0 -f /dev/sdd1
# mdadm /dev/md0 -f /dev/sde1
```
- ・ 縮退状態 (DM (LVM2)の場合)
 

DM (LVM2)で構成した RAID アレイを用いて、以下の手順にて disk\_rerr.stp を用いて一度模擬故障を発生させた RAID アレイを用いる。

RAID アレイが縮退していることは、md の場合 mdadm コマンドで、DM (LVM2)の場合は dmsetup コマンドを利用して確認できる。確認手順については「ソフトウェア RAID 設定手順書」の故障ディスクの特定の項を参照のこと。

- ・ 復旧処理中
 

上記、縮退状態の状態を作成した状態で、以下を実施

```
# mdadm /dev/md0 -r /dev/<故障の発生したディスク>
# mdadm /dev/md0 -a /dev/<故障の発生したディスク>
```

<故障の発生したディスク>は cat /proc/mdstat コマンドの出力で sdxx の右隣に(F)マークのついているディスクである。

(例) ディスク sdc で故障が発生した場合

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [raid1]
md0 : active raid1 sdd1[2] sdc1[3] (F) sdb1[0]
```

(3) 評価するディスク故障パターンに応じて、使用する故障模擬ライブラリのスクリプトを選択する。各ファイルと故障パターンの対応は以下の通りである。

temporary_rerr.stp	read 単発エラー
sector_rerr.stp	write によって訂正可能な read エラー
disk_rwerr.stp	read/write 不可エラー
disk_rerr.stp	read 不可エラー
temporary_werr.stp	write 単発エラー
r_timeout.stp	read 単発無応答
w_timeout.stp	write 単発無応答
rw_timeout.stp	read/write 無応答

(4) 模擬故障投入対象ファイルの設定

故障模擬ライブラリによる模擬故障発生の契機として利用する対象ファイルを決める。評価対象の RAID アレイ上にテストファイルを作成し、ls -hil コマンドにより、そのファイルの inode 番号を取得する。

```
# touch test.txt
# echo "test file" >> test.txt
# ls -hil
合計 28K
11 drwx----- 2 root root 16K 2007-12-21 18:22 lost+found
```

```
12 -rw-r--r-- 1 root root 21 2007-12-21 19:39 test.txt
```

(5) ページキャッシュ破棄

これまでの操作により、模擬故障を発生させる対象ファイルが作成されているが、ページキャッシュに乗っているため、この状態で対象となるファイルにリードアクセスしたとしてもディスクへアクセスが発生しない可能性がある。そこでページキャッシュを確実にフラッシュするために、システムを再起動する。

```
# reboot
```

(6) 故障模擬ライブラリの実体である SystemTap スクリプトの起動

模擬故障発生させるために、次のコマンドで対象となる模擬故障発生用 SystemTap スクリプトを起動する。

```
# stap ./<スクリプト名> <major> <minor1> <minor2> 1 <inode> -g -l ./fault_injection_common/ -v
```

<スクリプト名>は(3)で選択した故障に応じたスクリプト名を指定する。<major>は8固定である。

<minor1>と<minor2>は以下の表の値を用いる。

<inode>は(4)で調べた値を用いる。

(凡例：故障模擬ライブラリに与える引数 <minor1>, <minor2>)

	md (RAID1)	md (RAID5)	md (RAID6)	md (RAID10)	DM (LVM2)
通常運用時 (スペア有・無共通)	17, 33	17, 49	17, 65	17, 65	17, 33
縮退状態	33, 33	33, 49	17, 33	33, 65	17, 33
復旧処理中(スペア故障)	49, 49	65, 65	81, 81	81, 81	—
復旧処理中(アクティブ故障)	33, 33	33, 49	17, 33	33, 65	—

(7) SystemTap スクリプト準備完了待ち

stap コマンド投入直後に模擬故障投入のためのフックがかかるわけではなく、stap コマンドが準備完了を示す” Pass 5: starting run.”の文字列を表示するのを待ち合わせる必要がある。待ち合わせにかかる時間は環境依存だが、概ね数十秒程度である。

(8) 模擬故障発生のための RAID アレイへのアクセス

stap コマンドで設定した模擬故障フックで模擬故障を発生させる契機となるアクセスを、模擬故障投入デバイスに対して呼び出す。

A) read アクセスで模擬故障を発生させる場合

対象デバイスの対象位置から read アクセスを発生させる。inode 指定の場合は、(4)項で確認した inode を持つファイルを cat コマンドで read する。

```
# cat test.txt
```

故障を発生させたことで、コマンドがエラーとなるか確認する。以下はエラーとなった例である。

```
# cat test.txt
cat: test.txt: Input/output error
```

B) write アクセスで模擬故障を発生させる場合

対象デバイスの対象位置に write アクセスを発生させる。inode 指定の場合は、(4)で確認した inode を持つファイルに対して echo コマンドをリダイレクトし、続けて sync コマンドを実行することで write する。

```
# echo " test write" >> test.txt
# sync
```

#### (9) 模擬故障発生の確認

模擬故障を発生させるアクセスの結果、正しく故障が起きたか、故障模擬ライブラリが出力するメッセージから確認する。故障が発生した場合は、`scsi_decide_disposition` か、`scsi_add_timer` で始まる行がコンソールに出力される。

#### (10)stap コマンド停止

`stap` コマンドは明示的に止めない限り動作し続けるため、模擬故障が発生し、OS によるエラー処理が完了したと考えられる時点で、`Ctrl-C` で `stap` コマンドを止める。応答のある模擬故障の場合、模擬故障発生を確認してから数秒後に止める。無応答故障の場合、OS 内部でコマンドリトライが発生し応答を待つため、10 分待った後で止める。

#### (11)syslog の確認

OS が実施したエラー処理の内容が `syslog` に記録されている場合があり、処理を確認する。例えば、縮退状態の `md (RAID6)` アレイ内の `test.txt` に対して `read` 不可エラーを投入した場合、`cat` が失敗し I/O エラーとなると同時に、`sdb` にエラーが発生し、それを `md` ドライバが認識して切り離すことがわかる。`RAID6` の場合でも `raid5` と表示されるのは 2.6.22.6 の `md` の仕様である。

```
kernel: sd 0:0:1:0: [sdb] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE, SUGGEST_OK
kernel: sd 0:0:1:0: [sdb] Sense Key : Medium Error [current]
kernel: sd 0:0:1:0: [sdb] Add. Sense: Unrecovered read error - auto reallocate failed
kernel: end_request: I/O error, dev sdb, sector 2754823
kernel: raid5: Disk failure on sdb1, disabling device. Operation continuing on 1 devices
kernel: RAID5 conf printout:
```

#### (12)RAID アレイ 構成情報の確認

エラー処理後の `RAID` アレイの状態を確認する。確認方法については「ソフトウェア `RAID` 設定手順書」を参照。例えば縮退状態の `md (RAID6)` アレイ内の `test.txt` に対して `read` 不可エラーを投入した場合、`sdb` にエラーが発生し、`md` ドライバが認識して切り離したことを `mdadm` コマンドからも確認できる。

```
# cat /proc/mdstat
Personalities : [raid5] [raid6] [raid10]
md0 : active raid6 sde1[4] (F) sdd1[5] (F) sdc1[0] sdb1[6] (F)
      3919616 blocks level 6, 64k chunk, algorithm 2 [4/1] [_U_]
```

## 4.4 品質評価手順（故障ディスク交換評価）

### 4.4.1 故障ディスク交換評価環境の構築手順

ディスクのホットプラグに対応した装置上で、4.2 項の手順に従い、故障模擬ライブラリ動作環境を構築する。

### 4.4.2 故障ディスク交換評価実施手順

以下に故障ディスク評価操作手順および期待値の確認方法について説明する。`RAID` アレイの個々の操作詳細については「ソフトウェア `RAID` 設定手順書」、故障模擬ライブラリの操作方法については 4.3 項を参照。`LVM2` は機能未サポートのため評価対象外とする。

運用中の `RAID` アレイに故障が発生した場合に実施すべきホットプラグの手順を踏んでディスク交換を実施する。特に、運用中の故障発生ならびにディスク交換であることを想定して、あらかじめ対象となる `RAID` アレイに負荷を与える。具体的には以下の手順となる。

- (1) 対象となる `RAID` アレイを構築する。
- (2) 擬似的な高負荷環境を作るために、対象となる `RAID` アレイ上で読み書きを繰り返す操作を実施する。
- (3) 故障模擬ライブラリを用いて、対象の `RAID` アレイでディスク切り離しを引き起こす模擬故障を発生させる。故障の種類は `Read/Write` 不可エラーを用いる。
- (4) `mdadm -r` コマンドを用いて、対象の `RAID` アレイから模擬故障の発生したディスクを切り離す。
- (5) 以下の `SCSI proc` インターフェースを用いて故障の発生したディスクを OS から切り離す。

```
# echo "scsi remove-single-device 交換対象ディスクの位置情報" > /proc/scsi/scsi
```

ここで、交換対象ディスクの位置情報とは、空白で区切られた4つの数字で、これらは  
“cat /proc/scsi/scsi”コマンドで表示される、切り離し対象とするディスクに対する Host、  
Channel, Id, Lun 番号である。

- (6) 実際に筐体からディスクを抜き取り、保守部品と交換する。
- (7) 以下の SCSI proc インターフェースを用いて交換したディスクを OS に組み込む  

```
# echo "scsi add-single-device 交換対象ディスクの位置情報" > /proc/scsi/scsi
```

ここで、交換対象ディスクの位置情報は、上記手順中の切り離しの際に使用したものと同一値を用いる。
- (8) mdadm -a コマンドを用いて、対象の RAID アレイに交換したディスクを組み込む
- (9) ディスク交換後の RAID アレイに対して読み書きを実施する。  
read アクセスについては、アレイから cat コマンドでファイルが読めることを確認する。write アクセスについては、アレイ上のファイルに echo コマンド結果をリダイレクトし、ファイルに書き込みができることを確認する。
- (10) OS を再起動し、上記の手順で実施した、交換後の RAID アレイ上での書き込み結果が残っていることを、cat コマンドで確認する。

#### 4.4.3 故障ディスク交換評価結果確認手順

各評価パターンにおいて確認する点はディスク故障が発生したことの確認、継続運用確認、交換後構成情報確認、交換後のディスクの動作確認である。

ディスク故障発生の確認方法については、故障模擬ライブラリで模擬故障を発生させた場合の評価方法に準ずる。継続運用、交換後のディスクの動作確認については、交換したディスクに実際にアクセスを発生させて読み書きができることを確認する。構成情報については RAID アレイの種類に応じた方法で確認する。

## 4.5 テストプログラムのマニュアル

### 4.5.1 品質評価テストプログラム

書式

```
redundancy_test.sh [-t faultscript].. -p mountpoint -V vgname -L lvname devices..  
redundancy_test.sh [-t faultscript].. -p mountpoint -M mddev [-l mdlevel].. devices..  
redundancy_test.sh [-t faultscript].. -p mountpoint -V vgname -L lvname -M mddev [-l mdlevel].. devices..
```

説明

RAID アレイを構築し、故障模擬ライブラリを使用してディスク故障を擬似的に発生させ、ディスク故障時の動作を確認するプログラムである。LVM2 mirror、md-RAID1、md-RAID5、md-RAID6、md-RAID10 の評価をサポートしている。

本テストプログラムにより、指定した RAID アレイに対して、指定した種別のディスク故障を、md についてはスペアディスク付き通常運用時、スペアディスク無し通常運用時、縮退状態中、復旧処理中スペアディスク、復旧処理中アクティブディスクに対して投入できる。LVM2-mirror に対しては、通常運用時、縮退状態での模擬故障の投入ができる。なお、評価で用いるの RAID アレイは本テストプログラム内で動的に作成する。

**devices** には RAID アレイの構成要素となるデバイス名を指定する。

すべてのレベルの RAID アレイに対して評価を行うには5台以上のデバイス名を指定する必要がある。**mdadm** の RAID1 アレイのみの評価であれば3台以上のデバイス名で評価を行うことが可能である。ここで指定する RAID アレイを構成するデバイスは、8GB 以上の容量を持つディスクパーティションである。

-t faultscript	引数で指定した名前の故障模擬ライブラリのスクリプト(～.stp)で故障を発生させる。
-p mountpoint	引数で指定した名前のディレクトリを作成し、RAIDアレイをマウントする。テストプログラム内部でディレクトリを作成するため、存在しないディレクトリを指定しなければならない。
-V vgname	引数で指定した名前のボリュームグループを作成する。
-L lvname	引数で指定した名前の論理ボリュームを作成する。
-M mddev	引数で指定した RAID アレイのデバイスを作成する。
-l mdlevel	RAIDレベルを指定する。引数で"1"、"5"、"6"、"10"が指定された場合、それぞれ RAID1、RAID5、RAID6、RAID10のアレイを構築して評価を行う。-lオプションを複数回指定した場合、各RAIDアレイの評価を連続して実施する(実行例を参照)。-lオプションを指定しなかった場合、RAID 1、RAID 5、RAID 6、RAID 10のすべてのアレイの評価が実施される。-V、-Lと同時に-Mオプションも指定した場合には、mdadmで構築したRAIDアレイの評価が行われた後に、連続してLVM2で構築したRAID1の評価が行われる。-V、-Lまたは-Mのいずれのオプションも指定されなかった場合、エラーとなる。

実行例 1: read/write 不可エラーを md RAID5, md RAID6, LVM2 に対して投入する場合

上述の通り、md RAIDアレイに対しては、スペアディスク有り/無しそれぞれの場合の通常運用時、縮退状態、復旧処理中に対する評価を実施し、LVM2 mirror アレイに対しては通常運用時、縮退状態に対する評価を実施する。

- ①. デバイス/dev/sdb1、/dev/sdc1、/dev/sdd1、/dev/sde1、/dev/sdf1 を使用して mdadm で/dev/md0 を RAID5 アレイとして構築し評価を行い、その RAID5 アレイを削除する。
- ②. ①と同じデバイス群を使用して mdadm で/dev/md0 を RAID6 アレイとして構築し評価を行い、その RAID6 アレイを削除する。
- ③. ①と同じデバイス群を使用して LVM2 で論理ボリューム VG1/LV1 を RAID1(mirror)アレイとして構築し評価を行い、その論理ボリュームを削除する。

```
# redundancy_test.sh -t disk_rwerr.stp -p /home/test -V VG1 -L LV1 -M /dev/md0 -l 5 -l 6 /dev/sd[bcd]1
```

実行例 2: read 不可エラーを LVM2 に投入する場合)

```
# ./redundancy_test.sh -t disk_rerr.stp -p /home/test -V VG1 -L LV1 /dev/sd[bcd]1
```

## 4.5.2 故障模擬ライブラリ

### 書式

```
stap script_name major minor-min minor-max inode/LBAフラグ 値(inode/LBA) -g -l common-lib  
-path -v
```

### 説明

`script_name` で指定した模擬故障発生用のフックを `SystemTap` を用いて仕掛ける。模擬故障投入対象となるデバイスを `major/ minor` 番号で指定する。( `major, minor-min` ) から、( `major, minor-max` ) の範囲が投入対象となる。投入対象となるディスク上の場所は `LBA` を直接指定するか、対象ディスク上の `inode` を指定することができる。

`SystemTap` スクリプトが起動した後、最初に該当デバイスの該当場所に該当種類のアクセス(`read` もしくは `write`) が発生した場合に模擬故障を発生させる。`SystemTap` スクリプトの起動は、`SystemTap` の仕様により、”`Pass 5: starting run`” という文字列が出力されたことで確認できる。この表示以降のアクセスが模擬故障発生投入の対象となる。なお、スクリプトは `stap` コマンドを明示的に停止するまで持続する。

<code>script_name</code>	ディスク故障パターンとして以下の*.stpファイルのいずれかを指定する。
<code>temporary_rerr.stp</code>	read単発エラー
<code>sector_rerr.stp</code>	writeによって訂正可能なreadエラー
<code>disk_rwerr.stp</code>	read/write不可エラー
<code>disk_rerr.stp</code>	read不可エラー
<code>temporary_werr.stp</code>	write単発エラー
<code>r_timeout.stp</code>	read単発無応答
<code>w_timeout.stp</code>	write単発無応答
<code>rw_timeout.stp</code>	read/write無応答
<code>major</code>	blockデバイスのmajor番号。HDDの場合、8を指定する。
<code>minor-min</code>	模擬故障投入対象とするデバイスの最小minor番号を指定する。
<code>minor-max</code>	模擬故障投入対象とするデバイスの最大minor番号を指定する。
<code>inode/LBAフラグ</code>	後続の値がLBAを示すのか、inodeを示すのか判断するために使う。LBA指定を使用する場合0、inode指定を使用する場合1とする。
<code>値(inode/LBA)</code>	上記inode/LBAフラグの値に従って、inodeかLBAの値を10進で入力する。
<code>common-lib-path</code>	故障模擬ライブラリ共通部へのパス。通常、ディスク故障パターン用 *.stpファイルの直下の <code>fault_injection_common</code> を指定する。 (例) <code>./rhel4/fault_injection_common</code>

### 模擬故障投入対象範囲についての注意事項

- `minor-min` から `minor-max` の範囲内のデバイスを模擬故障発生の対象とする。この範囲に含まれる最初にアクセスの発生したデバイスにのみ擬似故障を発生させる。(例: `0 49` を範囲として指定した場合、`sda, sdb, sdc, sdd` が対象となる)
- `disk_rerror.stp` 等、引数で指定した範囲のデバイス/block にアクセスがあった際に、何度でも 模擬故障が発生するタイプのスクリプトにおいて、2 度目以降の模擬故障は最初に模擬故障が発生したデバイスのみで発生する。(例: `sdb` と `sdc` を範囲として指定、`inode 13` を指定して `disk_rerr.stp` 実行した場合、対象は `sdb` の `inode13` か `sdc` の `inode13` への `read` アクセスとなる。先に `sdc` の `inode 13` に `read` アクセスがあった場合、`sdc` の `inode 13` の `read` に模擬故障を発生させる。このあと続けて `sdb` の `inode 13` に `read` アクセスが発生したとしても 模擬故障は発生させない。一方、`sdc` の `inode 13` に再度 `read` アクセスが発生した場合、再度模擬故障を発生させる。)

### 実行例 1: read 単発エラー

`/dev/sdb, /dev/sdc` に `inode` 指定(`5005747`)で単発の `read` 模擬故障フックを投入する

```
stap ./temporary_rerr.stp 8 17 33 1 5005747 -g -l ./upstream/fault_injection_common/
```

### 実行例 2: read 不可エラー

`/dev/sda, /dev/sdb/, /dev/sdc, /dev/sdd` に `block` 指定で `LBA = 0x269ed9d (= 40496541)` に対して `read` 不可エラー発生フックを投入する。ここで、上記の注意事項でも説明した通り、模擬故障発生対象となる場所は `disk_rerr.stp` を投入後 (`sda,LBA 40496541`), (`sdb,LBA 40496541`), (`sdc,LBA 40496541`),(`sdd,LBA 40496541`)のうち、最初にアクセスした一つである。

```
stap ./read_rerr.stp 8 7 49 0 40496541 -g -l ./rhel4/fault_injection_common/
```