ext4 オンラインデフラグ実装に関する研究開発

藤田　朗　　佐藤　尚
ＮＥＣソフトウェア東北株式会社

概要

ext3 は Linux の標準ファイルシステムとして広く利用されてきたが、年々増加傾向にあるデータサイズに対応しきれなくなってきた。ext3 が抱える諸般の問題を解消する目的で、次世代ファイルシステムの ext4 が開発された。ext4 はできるだけ連続した領域のブロックを割り当てるよう考慮されているが、並列書き込み時や空き容量が少ない場合、フラグメントが発生しやすく、システム停止をせずにフラグメントを解消する方法がなかった。そこで我々は、オンラインでフラグメントを解消する方法を考案し、実装した。

本論文では、linux-2.6.31 の rc(リリース候補)版に含まれた ext4 オンラインデフラグ機能の実装とフラグメント解消による性能向上について述べる。

1. はじめに

ファイルシステムを長く運用している場合、それまでの書き込み、削除によりファイルデータのブロックがディスク上離れた領域に格納され、フラグメントが発生する。ファイルの読み込み時に複数箇所のブロックを検索するため、ハードディスクの無駄なシークが発生し、連続した領域に対するアクセスに比べ I/O 性能が悪化する。さらに、度重なるハードディスクのシークにより、駆動部分の耐久性が低下し、ひいてはハードディスク故障によるシステムの信頼性の低下にもつながる。

フラグメントが I/O 性能に及ぼす影響を図１に示す。フラグメントが発生した 1GB のファイルを読み込んだ場合、どのファイルシステム種別でも読み込み性能が劣化している。

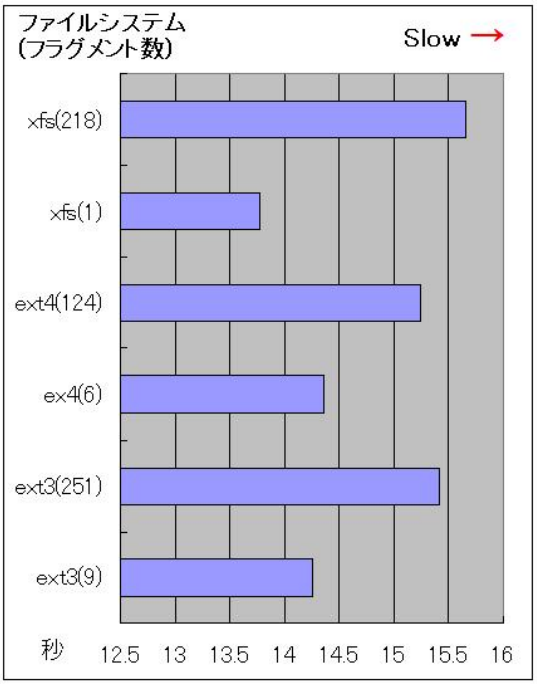

図１　フラグメントによる読み込み性能劣化

評価環境 カーネル: 2.6.31-rc3, アーキテクチャ: i386, CPU: Xeon 3.0GHz, メモリ: 1GB

先進的なファイルシステムでは、発生したフラグメントを解消し性能劣化を改善す

るためデフラグ機能に対応している(表１)。しかし、Linux の標準ファイルシステムである ext3 ではデフラグ機能に対応していない。つまり、ext3 ではシステム停止をせずにフラグメントを解消する方法がなかった。

ext4 がデフラグ機能に対応しない場合、Linux ユーザが ext4 採用を敬遠することも考えられる。さらに、デフラグ機能などの管理ツールが充実していないことは、Windows から Linux への移行を検討する際の障害にもなりかねない。しかし、デフラグ機能を実装し、フラグメントを解消することでハードディスクの I/O 性能が向上し、システム全体のスループットの改善も見込める。そこで我々は、ext4 オンラインデフラグ機能を開発し、Linux コミュニティへ提案した。

表１　主要ファイルシステム(FS)のデフラグ機能対応状況

FS 種別	デフラグ対応状況
ext3	×
ext4	○※
JFS	×
NTFS	○
XFS	○
Btrfs	○

○: 対応 ×: 未対応

上記は執筆時点の最新版である linux-2.6.31-rc3 を参考にした。

※　ext4 は我々が行った開発により、デフラグ機能に対応した。

2. ext4 の研究開発の動向

ext4 は機能強化と諸元拡大する目的として、Theodore Ts'o 氏を中心に開発が進められた。ext4 では、最大ファイルサイズが ext3 の 8 倍になり(表２)、ブロックアロケーション機能の強化がなされている(表３)。

表２　ext3、ext4 の諸元比較

FS 種別	最大ファイルサイズ	最大ＦＳサイズ
ext3	2TB	16TB
ext4	16TB	1EB

表３　ext4 の主要機能

機能	効果
Extents※	・ファイルサイズ、ファイルシステムサイズ拡大 ・ブロックの検索高速化
Delayed allocation	連続ブロック割り当て
Multi-block allocation※	連続ブロックでのブロック割当て効率化
Flex_bg, Uninit_bg	ファイルシステム異常時の復旧時間の短縮化
Journal checksumming	ジャーナルの性能と信頼性の向上

※　ext4 オンラインデフラグに関係する機能

●間接ブロック管理とエクステント管理

ext4 で扱えるファイルサイズが大きくなったのに伴い、新しいファイル管理方式が導入された。

ext4 では ext3 で用いられてきた従来の間接ブロック管理(図２)のほか、標準機能としてエクステント形式でのファイルブロック管理(図３)をサポートしている。

間接ブロック管理では、0-11 のブロックは直接ブロックであり、12,13,14 番目のブロックはそれぞれ 1 段、2 段、3 段の間接ブ

ロックである。ファイルサイズが大きくなるにつれて、多くの間接ブロックを参照する必要があり、大きいファイルを扱うのには向いていない。

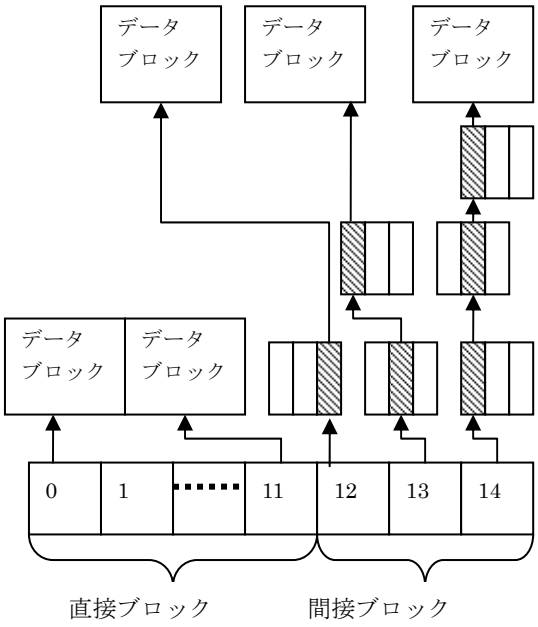

図２　間接ブロック管理

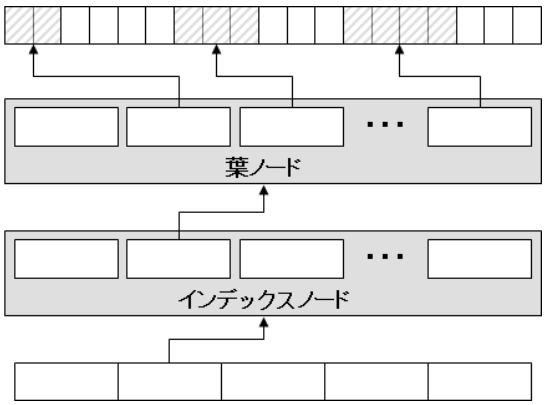

図３　エクステント形式での管理

エクステント形式では連続ブロックを領域として管理する。フラグメントが少ない場合、大きなファイルでもアクセス時間の低下は発生しにくい。ただし、フラグメントが悪化し、エクステント数が増加すると間接ブロック管理と同様に非効率なアクセスとなってしまう。

表４はext3, ext4上に1GBのファイルを作成した際のフラグメント数による読み込み時間の性能劣化を示す。ext3と比べext4のフラグメント数は減少しているが、ブロック管理の方法に関係なく、フラグメントが読み込み速度に影響していることがわかる。

表４　フラグメントによる性能劣化

FS (管理方法)	フラグメント	読み込み(秒)	性能劣化
ext3(間接)	9	14.697	-8.1%
	251	15.894	
ext4(間接)	5	15.554	-10.9%
	123	17.246	
ext4(エクステント)	6	14.901	-3.4%
	125	15.404	

フラグメントの作成方法:ddコマンドを10並列で実行。
読み込み(秒)はcatコマンドを実行した5回分の平均時間。

●フラグメント

ext4ではサポートできる最大ファイルサイズの拡大に伴い、データ書き込み時に連続した領域にデータを格納できるようにするため、以下の２つの機能をサポートし、フラグメントの発生を抑制している。

- Delayed allocation
  ディスクへのデータの書き込みを遅延させ､ブロック割り当てのリクエストをまとめることで連続した領域にデータを書き込む。
- Multi-block allocation
  1 度のブロック割り当て処理で連続した複数のブロックを割り当てる。

実際の運用時には、複数プロセスによる並列書き込み、空き容量の減少によりフラグメントが大量に発生しやすくなる。フラグメントによる I/O 性能の低下は、システム運用において軽視できるものではない。

ext4 は標準で大きなサイズを扱えるエクステント形式のファイルをサポートしている。従来の間接ブロック管理のファイルは互換性を維持するために利用可能となっているが、ファイルサイズの諸元も小さいことから、ext4 ではあまり利用されないことが見込まれる。そのため、ext4 オンラインデフラグ機能は、エクステント形式のファイルをサポートしている。

3. ext4 オンラインデフラグ実装方式

ext4 オンラインデフラグは新規に考案したユーザ空間プログラムの e4defrag コマンドとカーネル空間の EXT4_IOC_MOVE_EXTENT ioctl により実現される機能である。

ext4 オンラインデフラグ機能は 2006 年後半からLinuxコミュニティに提案活動を行ってきた[1] [2]。約 3 年間におよびコミュニティで議論[3] [4] [5] [6]を繰り返した結果、2009 年 6 月にリリースしたver. 2.0 が正式にLinuxに含まれることになった[7] [8]。

開発当初の ext4 オンラインデフラグは、デフラグに特化した ioctl を実装していた。コミュニティとの議論により、ioctl を汎用的にし、カーネル空間で行っていた donor file(データ交換用ファイル)の作成、連続ブロックの割当てをユーザ空間で行うよう変更した。

ここでは、ext4 オンラインデフラグ ver. 2.0 の実装について述べる。

3-1 e4defrag コマンドでのブロック交換

e4defrag コマンドによるユーザ空間からみたブロック交換処理を説明する。

① ユーザ空間で新規に donor_file を作成する。作成した donor_file に unlink システムコールを呼び出し、ファイルのリンク数を 0 にする。

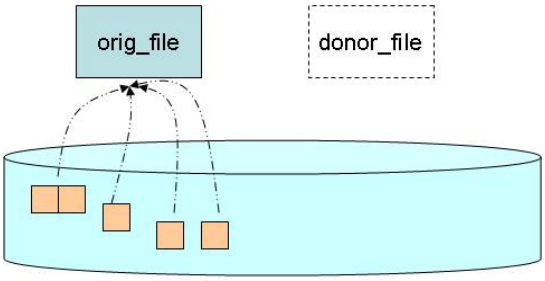

図 4 donor_file の作成

② fallocate システムコール[3] を用いて donor_fileに対して、orig_file(デフラグ対象ファイル)と同じ数の連続したブロックを割り当てる。

[1] Linuxファイルシステム開発用のメーリングリストのアーカイブ,
http://marc.info/?l=linux-fsdevel&r=1&w=2

[2] ext4 開発用のメーリングリストのアーカイブ,
http://marc.info/?l=linux-ext4&r=1&w=2

[3] fallocateシステムコール
開始オフセットからバイト分を指定し、領域を対象ファイルに割り当てるシステムコール。割り当てられた領域にはデータを書き込まないため、高速に処理が終了する。デフラグではこのシステムコールを用い、あらかじめ連続したブロックをファイルに割り当て、後に呼ばれる EXT4_IOC_MOVE_EXTENT でデータを書き込む。

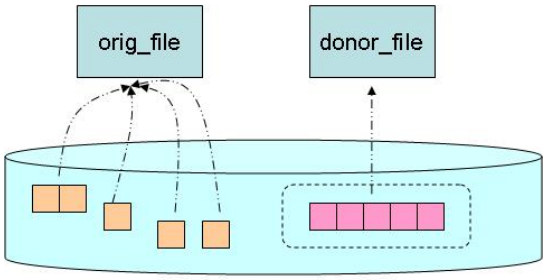

図５ fallocate によるブロック割り当て

③ FS_IOC_FIEMAP ioctl [4]を呼び出し、donor_fileのエクステント数を算出する。②で割り当てたブロックのエクステント数(=フラグメント数)がorig_fileより少ない、つまりデフラグによりフラグメントの改善が見込めることを確認する。

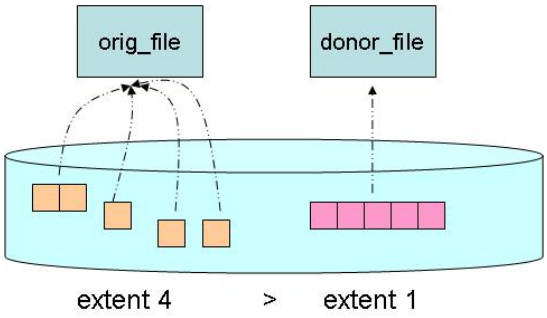

図６ エクステント数の比較

④ donor_file のエクステント 1 個ごとに EXT4_IOC_MOVE_EXTENT ioctl を呼び、orig_file、donor_file 間でデータブロックを交換する。この結果、orig_file と donor_file が指す物理ブロックが交換され、orig_file はフラグメントが少ない連続したブロックを保持することになる。

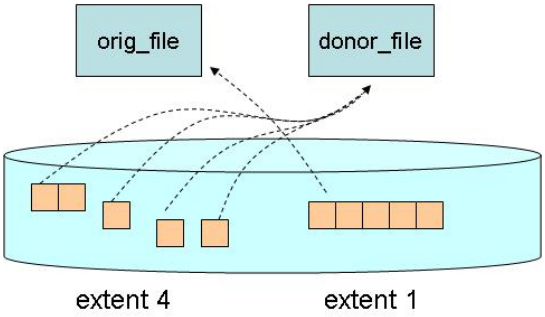

図７ ブロック交換

⑤ ブロック交換後の donor_file は必要ないため、後処理として close システムコールを呼び出す。リンク数が 0 である donor_file は削除される。

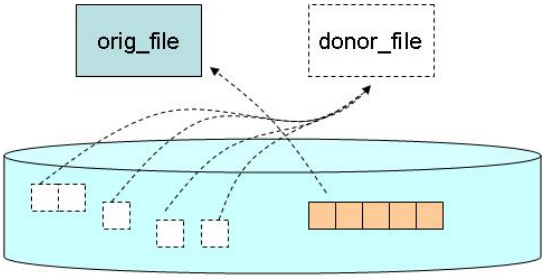

図８ donor_file の削除

3-2 EXT4_IOC_MOVE_EXTENT

EXT4_IOC_MOVE_EXTENT は指定した 2 つのファイル間でブロックを交換する ioctl である。move_extent 構造体(図 9)を入出力引数とし、ファイルの論理オフセット orig_start から len ブロック分をデフラグ対象ファイル(orig_file)とデータ交換用の一時ファイル(donor_file)間で交換する。

[4] FS_IOC_FIEMAP ioctl
指定したファイルの論理オフセット、物理オフセット、ブロック長などのエクステント情報を取得する ioctl。　取得したエクステント数はフラグメント数に等しいため、e4defrag コマンドでは orig_file, donor_file のフラグメント数を比較し、改善が見込める場合のみ処理を継続する。

```
#define EXT4_IOC_MOVE_EXT       _IOWR('f', 15, struct move_extent)
struct move_extent {
        __u32 reserved;         /* should be zero */
        __u32 donor_fd;         /* donor file descriptor */
        __u64 orig_start;       /* logical start offset in block for orig */
        __u64 donor_start;      /* logical start offset in block for donor */
        __u64 len;              /* block length to be moved */
        __u64 moved_len;        /* moved block length */
};
```

図９ EXT4_IOC_MOVE_EXTENT と move_extent 構造体

EXT4_IOC_MOVE_EXTENT ioctl のカーネル内部で行うブロック交換処理について説明する。

① move_extent 構造体に設定された値を元に、orig inode のデータを交換する論理オフセットの範囲を決定する。

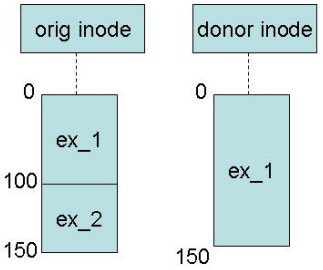

図１０　ブロック交換範囲の設定

※　上記の例では orig inode の先頭ブロックから 150 ブロック目までの 151 ブロックを対象とする。

② 指定された論理オフセットに該当する orig inode のファイルデータをページキャッシュに読み込む。

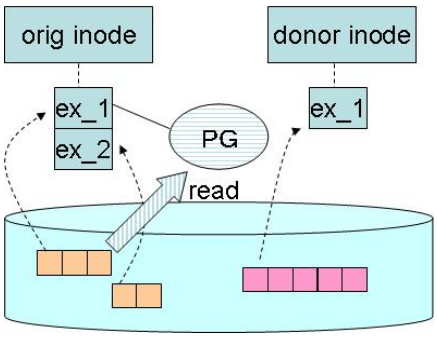

図１１　ページキャッシュ読み込み

③ データブロック交換用に orig inode, donor inode ともにエクステントを複製する。

④ ブロック交換処理はページサイズごとに行うため、複製したエクステントを１ページサイズ分に整形する。

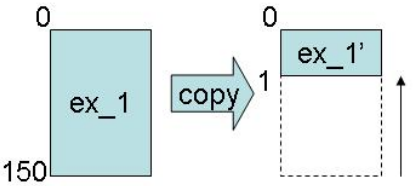

図１２　エクステントの複製とサイズ調整

⑤ 整形したエクステントを orig inode の

同じ論理オフセットに該当するエクステントへ上書きする。これにより、orig のエクステントが指すブロックが変更される。

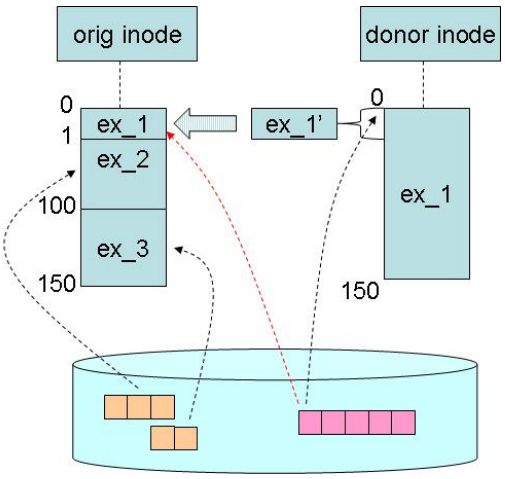

図１３　ブロック交換

⑥ ④、⑤で行った処理を今度は orig inode から donor inode へ実行する。これによりお互いのブロックがページサイズ分交換される。

⑦ ページキャッシュ上の変更されたデータをディスクへ書き出す。orig inode のデータが donor inode のブロックへ書き出される。

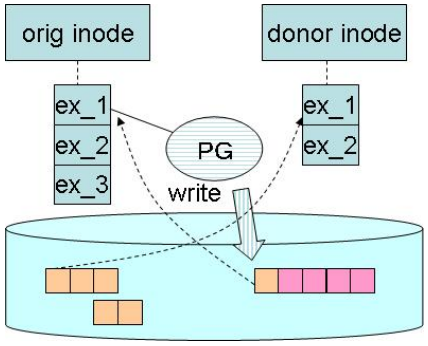

図１４ ページキャッシュ書出し

②-⑤の処理を１ページごとに実施し、①で算出した範囲(例では先頭から 150 ブロック目まで)を実施するまで繰り返す。

3-3 ファイルアクセスの排他

オンラインでデフラグを実施するために、カーネル空間では inode 構造体の mutex ロック(i_mutex)と ext4_inode_info 構造体のセマフォ(i_data_sem)を利用する。

i_mutex は他プロセスからの truncate を防ぐために取得され、カーネル空間での処理が終了するまで保持する。

i_data_sem はブロックを参照する処理では読み込みセマフォ、データブロックの割り当て・削除を行う処理では書き込みセマフォを局所的に取得・解放する。

XFS のデフラグコマンドの xfs_fsr はデフラグ実行中にファイルが変更された場合は処理が失敗となってしまうが、ext4 オンラインデフラグではこれらを用いた排他により、他のプロセスが占有しているファイルに対してもデフラグが実行可能となっている。

3-4 性能改善結果

ext4 上に作成した 2GB のファイルに対し、ext4 オンラインデフラグ実行前後のフラグメント数と読み込み時間の改善具合を表５に示す。デフラグの実行により読み込み時間が約１６％改善している。

表５ デフラグによる性能改善

デフラグ	フラグメント	読み込み(秒)	改善
前	257	39.893	15.6%
後	24	33.653	

評価環境 カーネル: 2.6.31-rc3, アーキテクチャ: i386, CPU: Xeon 3.0GHz, メモリ: 1GB

性能測定方法 : dd コマンドを 10 並列で実行し作成した 2GB のファイルに対し、ext4 オンラインデフラグ実行前後での cat コマンドの読み込み時間を測定。表 5 は 5 回測定した平均値。

4. 今後の課題

ext4 オンラインデフラグ機能を実装することにより、ファイル単位のデフラグが可能となった。今後は、アプリケーションの性能向上、連続空きブロックが少ない状態でのデフラグを可能にするため、以下の 2 つの機能の追加を計画している。

１．指定したディレクトリ配下のファイルを物理的に近い領域に配置する
２．ファイルシステムの空きが少ない状態でも指定したファイルのフラグメントを優先的に解消する

１はアプリケーションが同時にアクセスするファイルを近隣に配置し、アプリケーションの性能向上を図る機能である。

２は連続した空きブロックが少ない状態でも、デフラグ非対象ファイルを別の領域に移動することで連続した空き領域を作成し、そこにデフラグ対象ファイルを格納する機能である。これにより、ext4 上に十分な連続空きブロックがない状態でも、アクセス頻度が高いファイルなどに対し、優先的にデフラグを実行することが可能となる。

これら 2 つの機能をext4 オンラインデフラグに追加するためには、既存のext4 のブロックアロケーションをより柔軟で高機能なものにしなければならない。そのため、我々はLinuxコミュニティへブロックアロケーション制御機能の提案を実施した[9]。

ブロックアロケーション制御機能は、ext4 上の指定した領域への優先的なブロック割り当て、または特定の領域をブロックアロケートから保護することを可能にする。現在、ファーストリリースに対して得られた Linux コミュニティからの意見を反映し、次のリリースに向けての開発を行っている。

5. まとめ

ext4 ファイルシステムのフラグメントを解消する方法として ext4 オンラインデフラグ機能を考案し、実装評価した結果、フラグメント解消による性能向上を実現した。

継続的な提案活動と、Linuxコミュニティの協力もあり、ext4 オンラインデフラグ機能は 2009 年 6 月に、linux-2.6.31-rc1 にマージされた[5]。今後問題がなければ次のバージョンでLinuxの本流へ取りこまれる予定である。また、ユーザ空間プログラムの e4defrag コマンドも 2009 年 7 月に e2fsprogs-1.41.8 に含まれた[6]。

---

[5] **Linuxに取り込まれたパッチのcommit log,**
http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=748de6736c1e482e111f9d1b5a5d5b1787600cad

[6] **e2fsprogsに取り込まれたパッチのcommit log,**
http://e2fsprogs.git.sourceforge.net/git/gitweb.cgi?p=e2fsprogs;a=commit;h=4836459a3fe8ff208a7bf21e28e4648bddc76440

カーネル、コマンドともに正式なバージョンに含まれたことにより、今後多くの利用者からのフィードバックが得られ、より品質やユーザビリティの強化が図られることが期待される。

今後も ext4 への新機能の提案や開発、品質向上の活動を通じて、Linux コミュニティに貢献していきたい。

参考文献

[1] Takashi Sato. ext4 online defragmentation. *In Ottawa Linux Symposium (2007)*
https://ols2006.108.redhat.com/2007/Reprints/sato-Reprint.pdf

[2] Daniel P. Bovet, Marco Cesati.
詳解 LINUX カーネル 第 3 版 (2007), ISBN 978-4873113135

[3] Akira Fujita.
"[RFC][PATCH 0/3] ext4: online defrag (ver 1.0),"
http://marc.info/?l=linux-ext4&m=123329594919001&w=2

[4] Chris Mason.
"Re: [RFC][PATCH 0/3] ext4: online defrag (ver 1.0),"
http://marc.info/?l=linux-ext4&m=123334655401020&w=2

[5] Greg Freemyer.
"Re: [RFC][PATCH 0/3] ext4: online defrag (ver 1.0),"
http://marc.info/?l=linux-ext4&m=123335484515745&w=2

[6] Theodore Ts'o.
"Re: [RFC][PATCH 1/3] ext4: Add EXT4_IOC_DEFRAG ioctl and basic defrag functions,"
http://marc.info/?l=linux-ext4&m=123375929109850&w=2

[7] Akira Fujita.
"[RFC][PATCH 0/3]ext4: online defrag (ver 2.0),"
http://marc.info/?l=linux-ext4&m=124297598014251&w=2

[8] Theodore Ts'o.
"Re: [RFC][PATCH 1/3] Add EXT4_IOC_MOVE_EXT ioctl and related functions,"
http://marc.info/?l=linux-ext4&m=124489932017806&w=2

[9] Akira Fujita.
"[RFC][PATCH 0/7]ext4: Block allocation restriction and inode preferred range of blocks for ext4,"
http://www.spinics.net/lists/linux-ext4/msg13790.html