ext3/ext4の開発動向

2007年3月13日
NEC OSS推進センター
NECソフトウェア東北株式会社
佐藤　尚

自己紹介

- 2005/10からext3のサイズ拡大の開発を実施

  改造の基盤となる部分のパッチがLinuxに採用

- 2006/10からext4のオンラインデフラグ機能の開発を実施

  現在プロトタイプ版のパッチをコミュニティに提案し、実装に関して議論中

Index

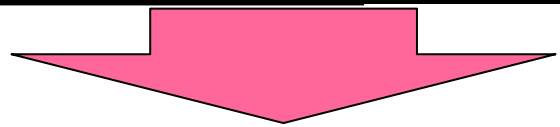
1. ext3/ext4サイズ拡大の動向
2. 私が実施したext3サイズ拡大
3. ext3/ext4のブロックアロケーション改造の動向
4. 私が実施したext4のオンラインデフラグ開発
5. 2007 Linux Storage & Filesystem Workshop
6. OSS開発を通して

1. ext3/ext4サイズ拡大の動向

ext3と他ファイルシステムとの比較

(Linux 2.6.9)

FS種別	32bit環境		64bit環境	
	最大FS サイズ	最大 ファイル サイズ	最大FS サイズ	最大 ファイル サイズ
XFS	16TB	16TB	8EB	8EB
JFS	16TB	16TB	32PB	4PB
ReiserFS	16TB	2TB	1EB	16TB
ext3	**8TB**	**2TB**	**8TB**	**2TB**



ext3の諸元は他ファイルシステムに比べて小さい。

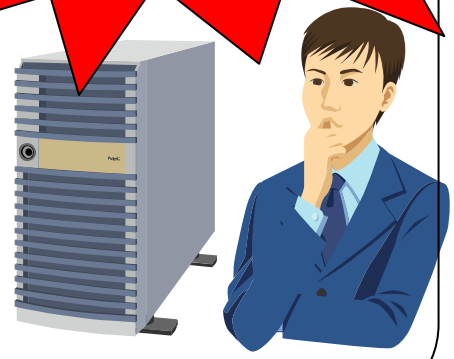
ディスクストレージ容量の増大

過去2年間でディスクストレージの出荷容量は約2倍に増大。
（調査機関調べ）
このペースが継続すると・・・

現状3TBのext3で
ファイルサーバを運用

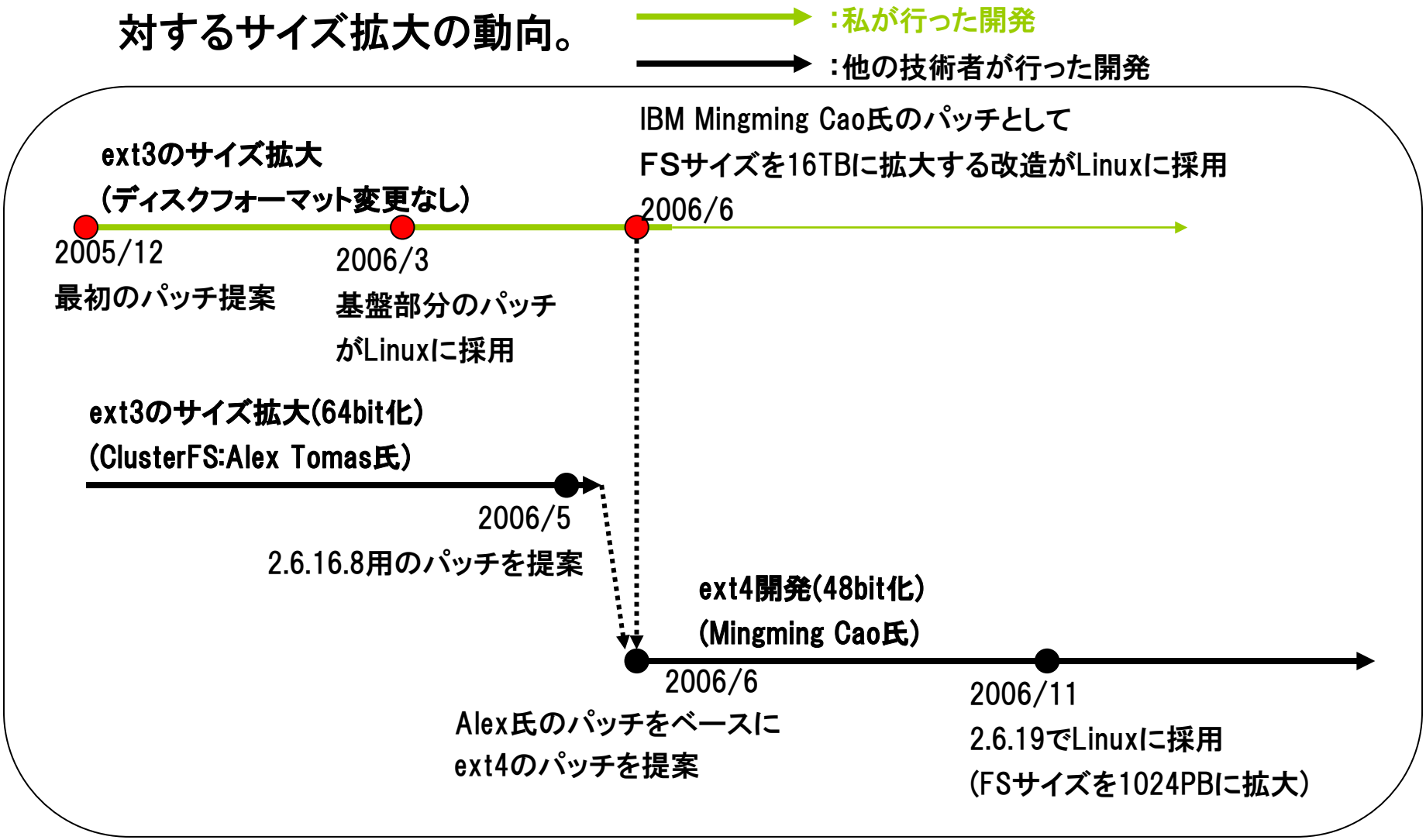
4年後

ディスクが12TBとなり、
ext3で運用できない！



ext3/ext4サイズ拡大の改造

ext3(最大ファイルシステムサイズ:8TB、最大ファイルサイズ:2TB)に対するサイズ拡大の動向。



2. 私が実施したext3のサイズ拡大

ブロックサイズ拡大

ext3の実装としてはページサイズまでのブロックサイズを使用可能。
そこで、ページサイズまでのブロックサイズを許可するようにマウント時のチェック処理を改造。

ファイルシステムサイズ拡大

現在符号付の4バイトで宣言されているブロック用変数の型を
符号なし4バイトに変更。これにより、最大ブロック数が2倍となった。
2G-1個→4G-1個

ファイルサイズ拡大

ディスクinodeのi_blocksをセクタ単位(512バイト)からFSブロック単位に変換したブロック数を格納することにより、最大ファイルサイズが約2倍となった。 2TB→4TB(4KBブロックサイズの場合)

ファイルサイズ、ファイルシステムサイズの拡大結果

FS種別	OS種別	32bit環境		64bit環境	
		最大FSサイズ	最大ファイルサイズ	最大FSサイズ	最大ファイルサイズ
XFS	Linux	16TB	16TB	8EB	8EB
JFS	Linux	16TB	16TB	32PB	4PB
Reiser4	Linux	16TB	2TB	1EB	16TB
ext3(オリジナル)	Linux	8TB	2TB	8TB	2TB
		2倍	2倍	32倍	128倍
ext3(拡張版)	Linux	16TB	4.1TB	256TB	256TB

パッチの動向

ファイルサイズ拡大のパッチ

- A. メモリinodeのi_blocks 8バイト化
- B.ディスクinodeのi_blocksをFSブロック単位のブロック数に変更

ファイルシステムサイズ拡大のパッチ

- C. ファイルシステムブロック数拡大
- D. ブロックサイズ拡大

パッチA,CがLinuxに採用。
その後、サイズ拡大の主流はext4開発へと移行。

最新のLinuxにおけるext3のサイズ

パッチがLinuxに採用されたことによりFSサイズが2倍に拡大。
(RHEL5 のβ版がすでに公開されダウンロード可能になって
おり、ext3のパッチはその中に含まれている)

FS種別	32ビット環境		64ビット環境	
	最大FSサイズ	最大ファイルサイズ	最大FSサイズ	最大ファイルサイズ
ext3(パッチ採用前)	8TB	2TB	8TB	2TB
	2倍		2倍	
ext3(パッチ採用後)	16TB	2TB	16TB	2TB

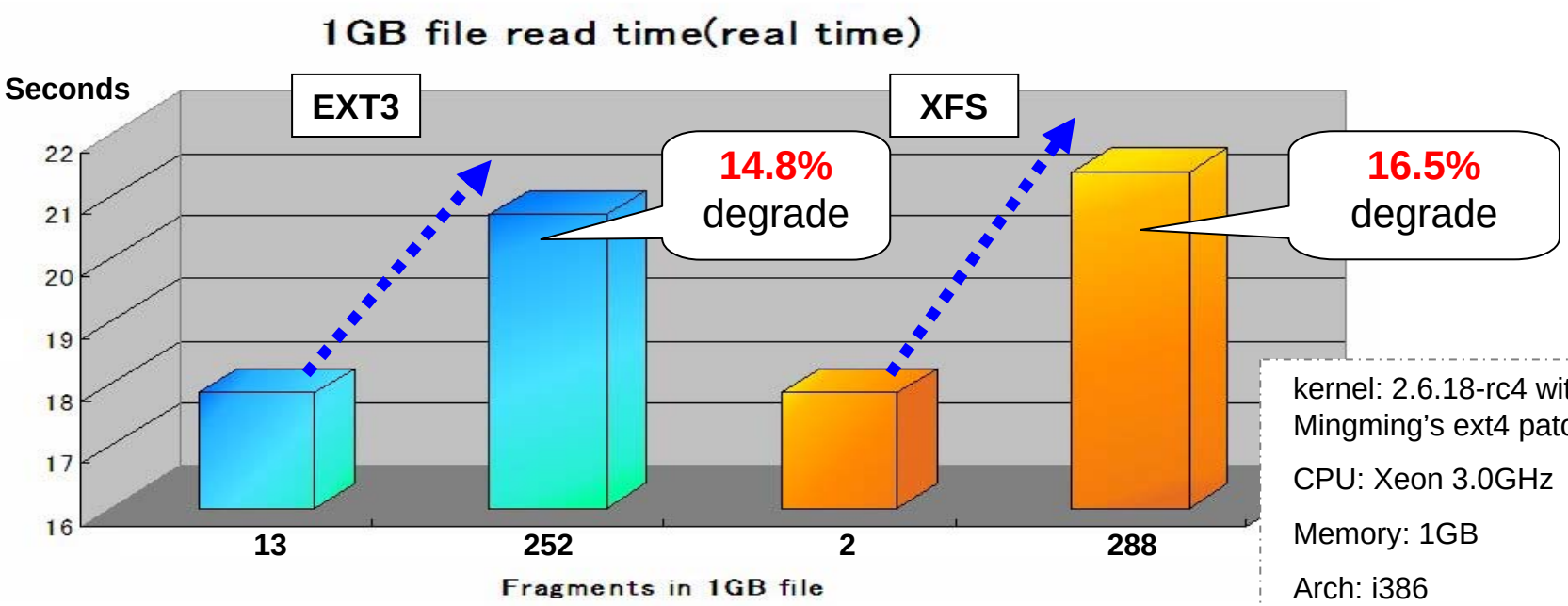
3. ext3/ext4のブロックアロケーション改造の動向

Linuxのファイルシステムにおけるフラグメント発生状況

ext4はブロック予約機能により可能な限り同一ファイルに連続ブロックを割り当てることができるが、多重のファイル書き込みを行うとフラグメントが発生し、ファイル読み込み性能が悪化する。

以下の2パターンでファイル書き込みを行った場合約15%の性能劣化を検出。

- 32個の1Gのファイルをシーケンシャルに書き込みした場合。
- 32個の1Gのファイルを32多重で書き込みした場合。



kernel: 2.6.18-rc4 with Mingming's ext4 patch set

CPU: Xeon 3.0GHz

Memory: 1GB

Arch: i386

フラグメントの種類

- **単一ファイルのフラグメント**

  単一ファイル内のブロックが不連続となっている状態。単一ファイルに対するファイルアクセス性能が低下する。

- **関連する複数ファイル間のフラグメント**

  アプリケーションが連続でアクセスするファイル群がディスク上で離れた位置に配置されている状態。サイズが小さい多数のファイルにアクセスするアプリケーションの性能が低下する。

- **フリースペースのフラグメント**

  ファイルシステムの空きブロックが連続せずに細かく分割されている状態。ファイルに対して連続ブロックを割り当てることができないため、ファイルのフラグメントが発生し易くなる。

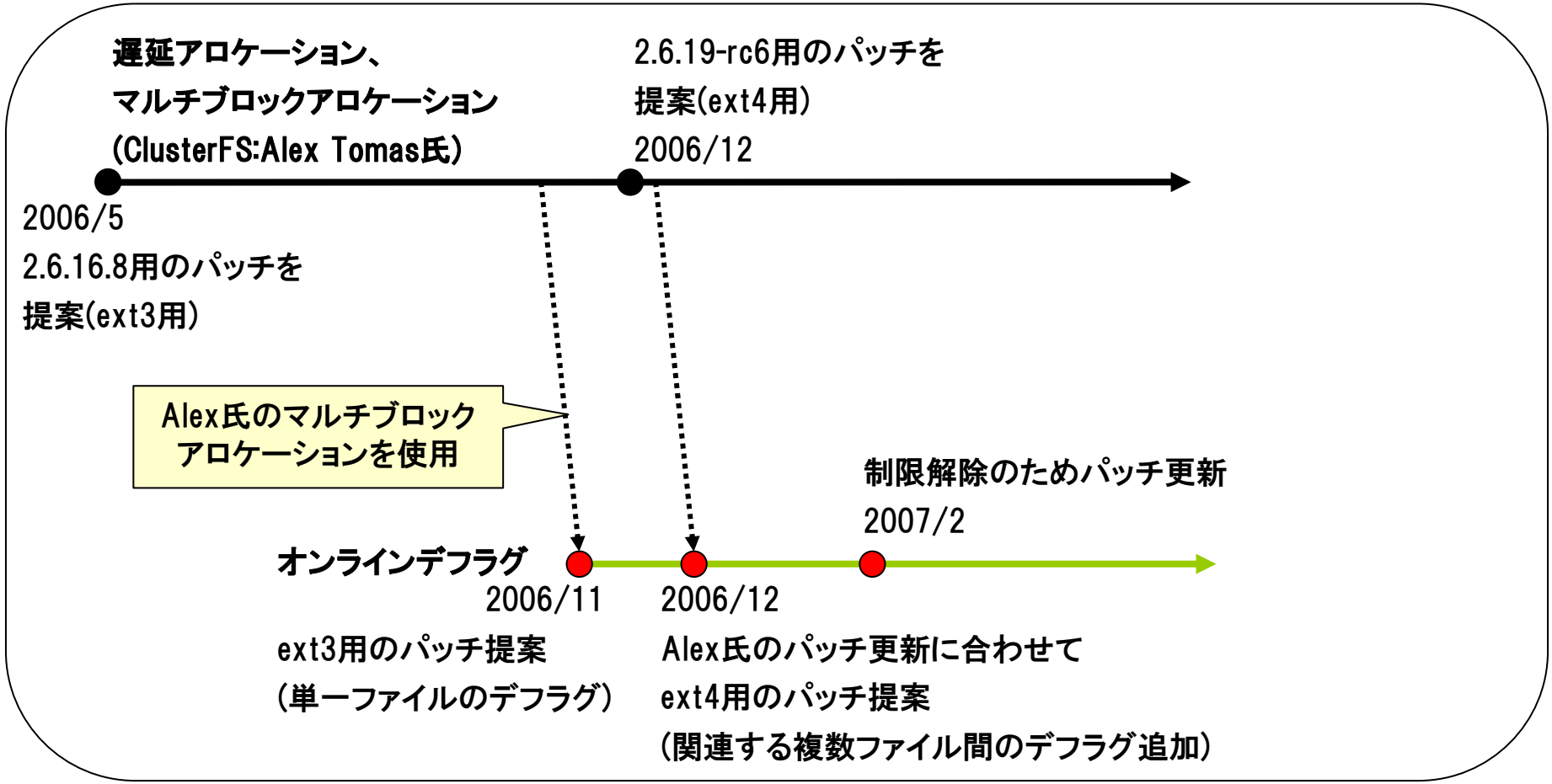
各ファイルシステムのオンラインデフラグ実装状況

ext3/4以外のファイルシステムでは既にオンラインデフラグを実装済のものもある。
巨大ファイルのI/O性能を向上させるためには、出来るだけ連続したブロックをアロケートする必要がある。そのため、巨大ファイルを保持可能なext4では、オンラインデフラグの実装が重要になる。

Linux					Windows	HP-UX
ext3	ext4	reiser4	JFS	XFS	NTFS	VxFS
-	-	-	✓	✓	✓	✓

✓ available , – not available

ext3/ext4のオンラインデフラグに関連する改造

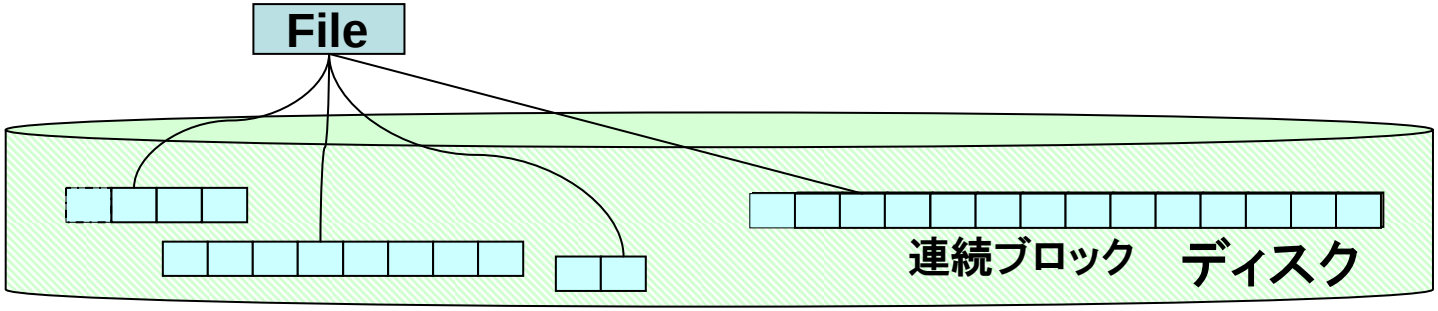


4. 私が実施したext4のオンラインデフラグ開発

オンラインデフラグとはファイルシステムをマウントした状態でファイルのフラグメントを解消する機能。

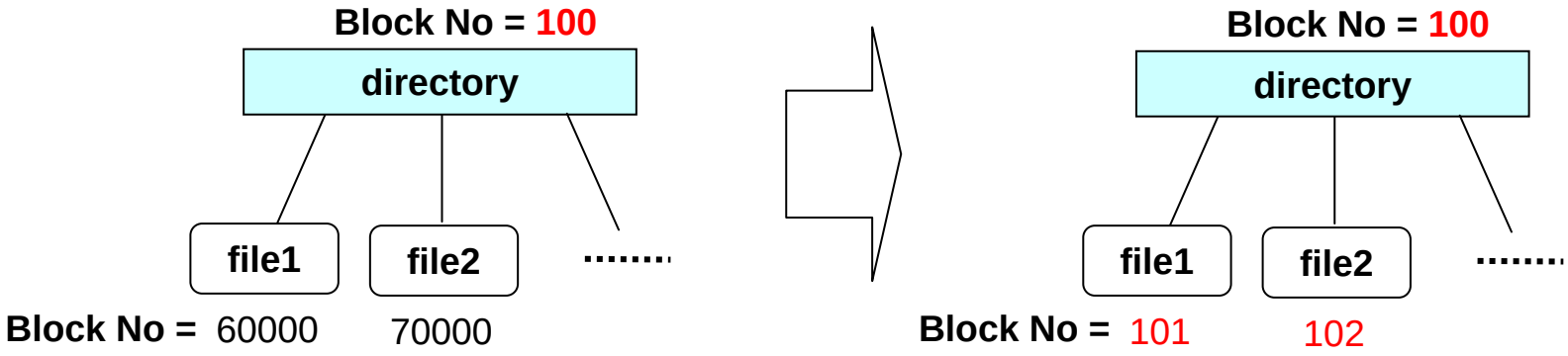
- 単一ファイルのデフラグメント

フラグメント化しているファイルのブロックを連続ブロックと置き換える。



- 関連する複数ファイルを近隣のブロックに移動

指定ディレクトリの近隣のブロックに配下の全てのファイルを移動。



デフラグメントの効果

- 単一ファイルのデフラグメント

  フラグメント化された1GBの50個のファイルに対してデフラグメントを実行。25%の読み取り性能の向上を確認。

	フラグメント数	I/O performance(秒)
デフラグ前	12175	618.3
デフラグ後	800	460.6

- 関連する複数ファイルを近隣のブロックに移動

  Linux 2.6.19-rc6のソースファイル(約20000ファイル)に対してデフラグメントを実行。全ファイルの読み込みに要する時間が29%改善。

	I/O performance(秒)
デフラグ前	42.2
デフラグ後	30.0

(# find top_dir –type f –exec cat {} ¥;).

今後の開発計画

- フリースペースのデフラグメント機能実装

  デフラグ対象外のファイルを移動しフリースペースを確保してから対象ファイルのデフラグを実行する機能。

- デフラグメント機能の動作安定化

  今後十分に評価を実施し、動作の安定化を図る。

5. 2007 Linux Storage & Filesystem Workshop

- Linuxのストレージ関連機能およびファイルシステムの開発者が検討している機能の実装について発表を行い、集まった開発者間で議論を行うことにより実装検討の促進を図ることを目的とするワークショップ。

- Filesystem trackの参加者は30人程度でLinux 2.6のメンテナであるAndrew Morton氏やext2/ext3/ext4の管理コマンド(e2fsprogs)のメンテナである Theodore Ts'o氏等多数の著名なメンテナが参加しており、Linux本流に採用されるための重要なコメントを得ることが期待できる。

参加することにより得た成果

- 8つのfilesystem trackの発表のうち4つがブロック配置の最適化やブロック検索ロジックの工夫による性能向上をテーマとしており、今後ストレージ拡大に伴いファイルシステムも拡大していく中で性能の向上が重要なテーマとなることを確認することができた。

- オンラインデフラグに実装すべき新機能や現状の実装に関する問題点等有効なコメントを頂くことができた。また、それらのコメントに関してその場で議論を行い、実装案についても提案頂き今後の開発促進に役立つ。

- 発表の後にext4開発者によるBOFが開催され、普段は電話会議やメールでしかコミュニケーションを取っていない技術者とF2Fで議論を行い、お互いの開発の方向性を確認することができた。

6. OSS開発を通して

- **パッチの提案はスピーディに**

  機能の概念や処理方式についてコメントをもらうことが大事。
  長時間かけて評価したパッチを提案後、根本的な考慮モレを指摘され
  今までの作業が無駄になってしまうことも…

- **コミュニティの議論の動向を確認し柔軟に実装方針を変更**

  実装が困難で悩んでいることも、既に他の技術者が提案済の機能
  を使用することで、容易に実装できることもある。

- **各シンポジウム等で自分の開発を紹介することで開発促進**

  各専門分野の技術者からの有効なコメントを得ることにより開発を促進
  することができる。外国語が堪能じゃなくても必死に伝える姿勢で臨めば、
  話を聞いてもらい有効なコメントをもらえる。

Empowered by Innovation