ext3 諸元拡大に関する研究開発動向、及び改造方式の検討

ＮＥＣソフトウェア東北株式会社　　佐藤　尚
日本電気株式会社　ＯＳＳ推進センター　　立川　江介

概要

現在、Linux の標準で利用されているファイルシステムとして ext3 がある。ext3 の最大ファイルシステムサイズは約 8TB、最大ファイルサイズは約 2TB であり、現在のディスクストレージ容量の増大傾向から考えると、近い将来に諸元不足となる可能性がある。ext3 は Red Hat Enterprise Linux 4 等において標準のファイルシステムとして利用されており、利用実績が多く一定の安定性が確保されている。そのため、利用者は今後も継続利用を希望すると考えられ、ext3 の最大ファイルシステムサイズ、最大ファイルサイズ拡大に対するニーズが高まると考えられる。

現在 Linux 開発用メーリングリスト[1][2]では ext3 の諸元拡大に関して数件の実装案が議論されている。これらの案は最大ファイルサイズのみを拡大する方式や、ディスクフォーマット及び制御ロジックに大きな改造を行うため品質安定化までには数年を要すると考えられる方式のみである。従って現状の ext3 の品質を維持しつつ短期間に最大ファイルシステムサイズ、最大ファイルサイズの両方を拡大する改造方式は存在しない。そこで、我々は新しい ext3 諸元拡大方式を前述のメーリングリスト上で議論を行い検討した。

本論文では、これまでに議論されてきた ext3 の諸元拡大方式を紹介し、さらに我々が現在前述のメーリングリスト上で議論している改造方式について述べる。

1.　はじめに

IDC Japan の報告書[3]によると、国内ディスクストレージシステムの 2005 年～2009 年の年間平均成長率（CAGR：Compound Annual Growth Rate）は、出荷容量で 59.6%と予測しており、年々増大傾向にあることが分かる。

また、現在ext3 以外のファイルシステムの最大ファイルシステムサイズ、最大ファイルサイズは表 1、表 2のようになっている。（表１、表２中の値は有効数字４桁以下の精度において多少の増減がある。以降の最大値に関する表記も同様である）

FS 種別	OS 種別	最大FS サイズ	最大 ファイル サイズ
ext3	Linux	8TB	2TB
ReiserFS	Linux	16TB	2TB
XFS	Linux	16TB	16TB
JFS	Linux	16TB	16TB
VxFS	HP-UX	1TB	1TB
HFS+	Mac OS	16TB	16TB
NTFS	Windows	16TB	16TB

表 1　各ファイルシステムの諸元(32bit 環境)

FS 種別	OS 種別	最大FS サイズ	最大 ファイル サイズ
ext3	Linux	8TB	2TB
ReiserFS	Linux	1EB	16TB
XFS	Linux	8EB	8EB
JFS	Linux	32PB	4PB
VxFS	HP-UX	32TB	2TB
HFS+	Mac OS	16EB	16EB
NTFS	Windows	256TB	16TB

表 2　各ファイルシステムの諸元(64bit 環境)

ext3 の諸元は XFS や JFS のような大規模ファイルシステムと比べると小さいが、多くのディストリビューションに標準的に組み込まれており利用実績が多く、一定の信頼性が保証されている利点がある。そのため、現在の信頼性を維持しつつ ext3 の最大ファイルシステムサイズ、最大ファイルサイズを拡大する必要があると考えられる。

そこで我々は、ext3 諸元拡大方式の検討と改造を実施した。

2. ext3 のサイズを制限している要因

- **ファイルシステムサイズの制限**

  ext3 のディスク上のフォーマットは図 1に示すように、ファイルシステムを構築するパーティション上にブロックグループと呼ばれるブロックの集合単位に分割される形式となっている[4][5]。

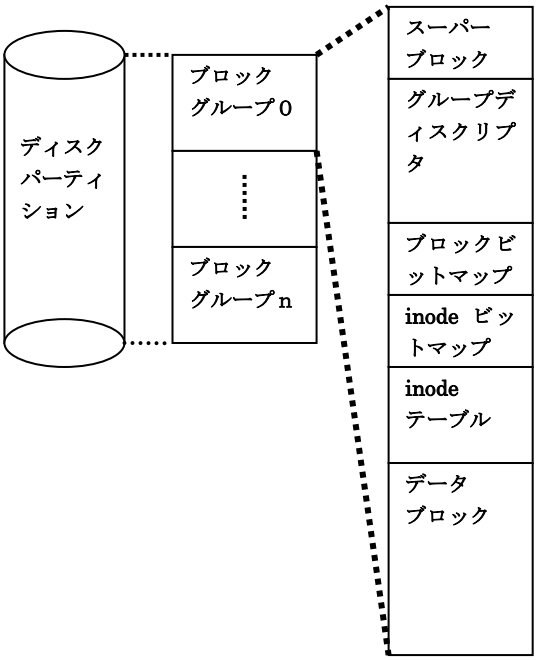

図 1　ext3 のディスクフォーマット

現状ブロック関連の構造体エントリのサイズは 4 バイトである。例えば、スーパーブロックに存在するファイルシステムの全ブロック数を保持するエントリは以下のように 4 バイトで定義されている。

```
struct ext3_super_block {
        :          :
      __le32    s_blocks_count;
        :          :
}
```

また、これに加えてカーネル内のブロック数を扱う変数が符合付きの 4 バイトで定義されている箇所が多数存在する。このため、ext3 の最大ファイルシステムブロック数は 2^{31}-1 個に制限される。ext3 の最大ブロックサイズは 4KBであることから最大ファイルシステムサイズは約 8TB（8TB－4KB）となる。

- **ファイルサイズの制限**

  ext3 ディスク inode（ext3_inode）

内の 512 バイト単位でブロック数を保持するエントリ(i_blocks)は 4 バイトである。また、VFS inode の対応するエントリ(i_blocks)も 32bit 環境では 4 バイトである。そのため、最大ファイルサイズは約 2 TB となる。(実際には間接ブロック等の管理ブロックも含まれるため 2TB より少し小さくなる。)

また、ext3 のディスク inode (ext3_inode) は、ファイルのブロックを 4 バイトの 15 個分の配列 (i_block) を使用して、以下の図 2に示すように間接ブロックを利用した方式で管理している。

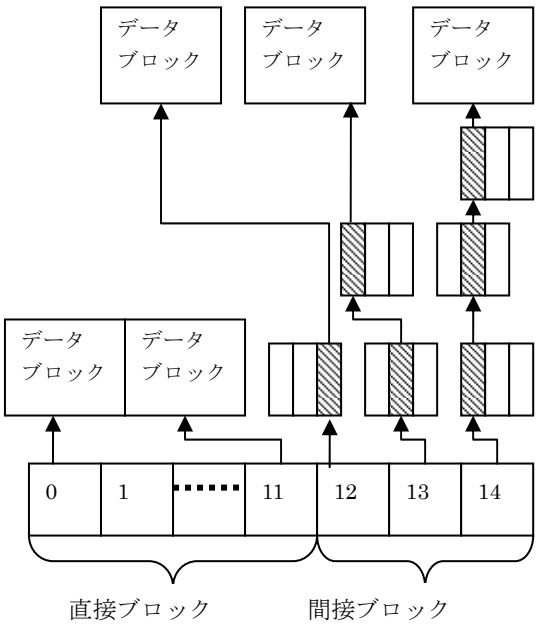

図 2　ext3 のファイルブロック管理

上記配列の 0～11 の要素は直接データブロックのブロック番号を保持する。12 以降は間接ブロックの番号を指し、間接ブロックの中にデータブロックの番号を保持する。12,13,14 の要素になるにつれて間接の段数が 1 段、2 段、3 段と増える。ファイルブロックがこのような構造で管理されるため、ext3 の最大ファイルサイズは i_blocks の制限が無かったとしても、以下のように約 4TB となる。

(　12(直接ブロック)
+　1024(第 1 間接ブロック)
+ 1024×1024(第 2 間接ブロック)
+ 1024×1024×1024(第 3 間接ブロック)
) ×4096(ブロックサイズ)
= 4402345721856 バイト(約 4.1TB)

3.　既存の ext3 諸元拡大改造方式

これまでに Linux 開発用のメーリングリスト[1][2]では ext3 の最大ファイルシステムサイズ、最大ファイルサイズ拡大に関して数件の実装方式が議論されている。これらの方式は、最大ファイルサイズのみを拡大する改造方式や、ディスクフォーマット及び制御ロジックに大きな改造を行うため品質安定化までには数年を要すると考えられるものであった。ここでは、それらのうち以下の主要な 2 件について実装方式を説明する。

3.1　ext3_inode 内の i_blocks 拡大

Goldwyn Rodrigues 氏が提案している改造方式である[6]。

ext3 ディスク inode (ext3_inode) 内の i_blocks と VFS inode 内の i_blocks を 8 バイト化することにより最大ファイルサイズのみを以下のように拡大する。

	改造前	改造後
ファイルサイズ	2TB	4TB

表 3　i_blocks 拡大による最大ファイルサイズ拡大結果

- **ext3 ディスク inode 内の i_blocks の 8 バイト化**

  ext3_inode 内の以下のエントリを、i_blocks の上位 4 バイトとして使用する。

  struct {
  __u32 h_i_translator;
  }

  上記エントリは HURD(GNU により開発されたＯＳ)において使用されるため、2TB 以上のファイルを作成した場合は、ext3 を HURD から使用できなくなるデメリットがある。

- **VFS inode 内の i_blocks 8 バイト化**

  VFS inode 内の i_blocks の型を unsigned long から sector_t に変更する。sector_t は通常のカーネルでは 8 バイトとなる。(エンベデッドシステム等の小規模システム用のコンフィグレーションでは 4 バイトとなる。)

3.2 ext3 64bit 化

Laurent Vivier 氏らにより提案された改造方式である[7]。

ext3 のファイルシステムサイズやファイルサイズ等を管理するデータ領域を現状の 4 バイトから 8 バイトに拡大する改造である。この方式では以下の表 4に示すようにXFSやJFS相当の最大ファイルシステムサイズ、最大ファイルサイズを実現できる。

	改造前	改造後
FS サイズ	8TB	512PB
ファイルサイズ	2TB	512PB

表 4 ext3 64bit 化による拡大結果

- **ブロック関連の構造体エントリ、及び変数の 8 バイト化**

  ext3 のディスク上及びメモリ上の構造体のブロック関連エントリを 8 バイト化する。例えば、ext3 のディスクスーパーブロック（ext3_super_block）では以下のように空きエントリを利用し、既存のブロック関連エントリに対応する上位 4 バイト用のエントリを追加している。

```
ext3_super_block {
    以下は既存のブロック関連エントリ
    __le32  s_blocks_count;
    __le32  s_r_blocks_count;
    __le32 s_free_blocks_count;
    以下は上記エントリの上位 4 バイト用として追加で定義したエントリである。
    __le32  s_blocks_count_hi;
    __le32  s_r_blocks_count_hi;
    __le32 s_free_blocks_count_hi;
}
```

  また、構造体以外でもブロックを扱う関数のインターフェース、及び変数の型を sector_t に変更している。

- **ブロックグループ内のブロック数拡大**

  ブロックグループ内のブロック数は、現状グループ内にブロックビットマップ用のブロックが 1 つしか存在しないため、ブロックサイズ×8 個に制限されている。つまり、4KB のブロックでは 32768 個がグループ内に保持できる最大のブロック数となる。

  そのため、以下の図 3に示すようにブロックビットマップを複数ブロックにしてブロックグループを拡大してい

る。

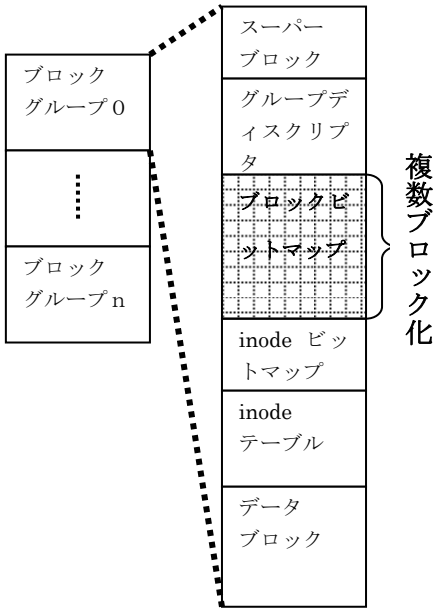

図 3　ブロックグループの拡大

- **エクステント方式によるファイルデータブロック管理**

ファイルブロックは従来の間接ブロック方式のように１ブロックずつ管理するのではなく以下の図 4に示すエクステントと呼ばれる構造により管理する。

エクステント(ext3_extent)

論理ブロック番号	物理ブロック番号	ブロック数

- 論理ブロック番号
  ファイル相対のブロック番号
- 物理ブロック番号
  ファイルシステム相対のブロック番号
- ブロック数
  エクステントで管理されるブロック数

図 4　エクステントの構造

エクステントは複数個の連続ブロックを１つのエクステントにより管理できるため、多数ブロックを管理する際のディスク上の管理領域の効率化や、ブロック検索時の性能向上を実現できる。

エクステントは以下の図 5に示すように管理される。この図はエクステントの深さが 2 段の場合の例である。エクステント数が多い場合はエクステントの深さがさらに深くなる。

ext3_extent_idx は次のノードブロックの物理ブロック番号、論理ブロック番号を保持する構造体である。

n 次ノードブロック内の ext3_extent_idx エントリ群、またはリーフノード内の ext3_extent 群は論理ブロック番号により昇順にソートされている。カーネルはファイルブロックを検索する際に二分木探索アルゴリズムによる高速な検索により ext3_extent_idx または、ext3_extent を検索する。

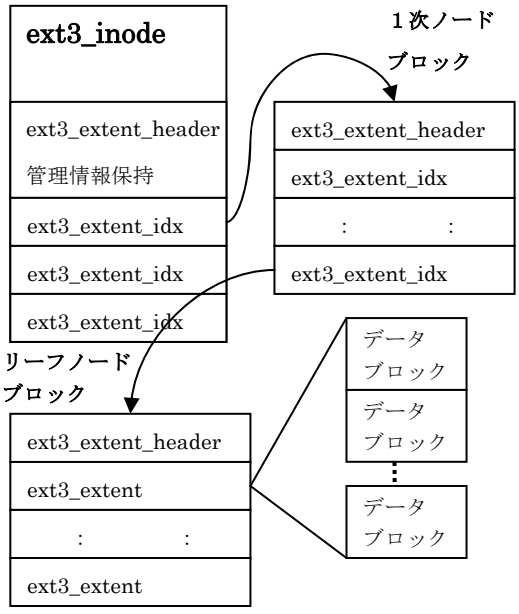

図 5　エクステントによるブロック管理

4.　新しい改造方式の検討

上記の 2 つの改造方式のうち **3.1** で述べ

ている方式はファイルシステムサイズの拡大ができない。また、3.2 で述べている方式では、ディスクフォーマットの大規模な変更やロジックに関する修正も多数行っているため、評価により品質が安定しディストリビューションに含まれるようになるまで数年を要すると考えられる。

そこで、我々は現状の ext3 の品質を維持しつつ ext3 のファイルシステムサイズ、ファイルサイズの両方を短期間で拡大するため以下のような方式を検討した。(ターゲットは最新の Linux 2.6 系カーネルである)

4.1 ファイルシステムサイズ拡大

- **ブロック用変数の型変更**

  ext3 のブロックを扱う構造体エントリの型は 4 バイトなので、本来は 2^{32}-1 個までのブロック数を保持することができるはずである。しかし、ブロックを扱う関数のインターフェースや変数の型に符号付の 4 バイトを使用している箇所が多数あるため、最大ファイルシステムブロック数は 2^{31}-1 個に制限されている。

  そこで、ext3 のカーネルコードにおいてブロックを扱っている符号付 4 バイトの変数の型を符号なし 4 バイトに変更した。これにより、ext3 の最大ファイルシステムブロック数は 2^{32}-1 個となる。

- **ブロック数出力時のフォーマット文字列修正**

  現在ブロック数出力時のフォーマット文字列に%d,%ldを多数箇所で使用しているため、2^{31}個以上のブロックを扱う際に不正なブロック番号がカーネルメッセージに出力される。そのため、それらの箇所において%u,%luを使用するように変更した。

- **percpu_counter の long long 化**

  percpu_counter はメモリスーパーブロック(ext3_sb_info)において空きブロック数をカウントするために使用される構造体である。

  percpu_counter は以下のように CPU 毎のカウンタ(counters)を持ち各 CPU のカウンタ値が一定の値（正負値両方）に達した時にメインカウンタ(count)にその値を反映する仕様である。これはカウンタ更新時の各ＣＰＵ間の排他制御によるオーバーヘッドを避けるためである。

```
struct percpu_counter {
        spinlock_t lock;
        long count;
        long *counters;
};
```

  現状このカウンタの型が long のため型を変更する必要がある。ただし、count は各 CPU の反映のタイミングにより負値となる場合もあるため、unsigned long にはせずに、long long とすることにした。

  また、percpu_counter は ext3 以外の共通処理でも使用しているため、新規に以下のような percpu_llcounter を作成し ext3 で使用するようにした。

```
struct percpu_llcounter {
        spinlock_t lock;
        long long count;
        long long *counters;
};
```

4.2 ブロックサイズ拡大

現状マウント時のチェックにおいて 4KB より大きいブロックサイズの ext3 をマウントできないようになっている。しかし、実際にはアーキテクチャのページサイズまで

のブロックサイズが使用可能である。(ext2 では Linux 2.4 において既にページサイズまでのブロックサイズを持つファイルシステムのマウントを許可している。)

そこで、ext3 においても同様に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した。

上記の改造により、32bit環境（最大ブロックサイズが 4KB）では最大ファイルシステムサイズがXFSやJFS相当の 16TBに拡大される。また、それ以上に拡大可能な64bit環境も含めた最大ファイルシステムサイズを表 5に示す。

ブロックサイズ	改造前	改造後
4KB	8TB	16TB
8KB	―	32TB
16KB	―	64TB
64KB	―	256TB

表 5　ファイルシステムサイズ拡大結果

4.3 ファイルサイズ拡大

- **VFS inode 内の i_blocks の 8バイト化**

  現在VFS inode内のi_blocks(512 バイト単位のファイルブロック数保持)のサイズが 32bit環境において 4 バイトであるため、図 6に示すように 2TB 以上のファイルに対するstat64 システムコールのst_blocksの結果が不正となる。

  本問題は2TB以上のファイルを作成できるファイルシステム(XFS,JFS 等)における既存の問題である。

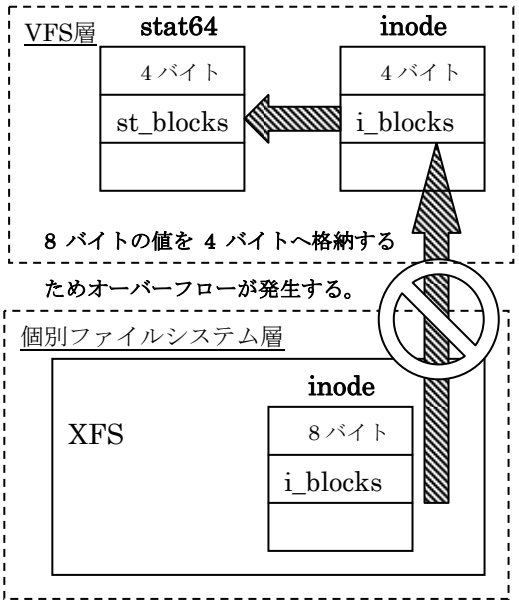

図 6　st_blocks が不正となる問題

そこで、i_blocks の型としてブロックデバイスのブロック数として使用されている sector_t を使用することにした。sector_t はコンフィグパラメータ CONFIG_LBD により 8 バイトと 4 バイトに切り替えが可能であり、デフォルトでは 8 バイトである。

- **ディスク inode(ext3_inode) 内の i_blocks の単位変更**

  ディスク inode(ext3_inode) 内の i_blocks は 512 バイト単位のブロック数を __le32(4 バイト)で管理しているため、最大ファイルサイズは 2TB に制限されている。

  そこで、ディスクフォーマットを変更せずに最大ファイルサイズを拡大するため、ext3_inode 内の i_blocks の単位を 512 バイト単位からファイルシステムブロックサイズ単位に変更する。

  また、i_blocks の単位がファイルシ

ステムブロックサイズであることを示すフラグを、mke2fs コマンドにおいてスーパーブロックに設定する。

カーネルはスーパーブロックに当該フラグが設定されていた場合は、以下の図 7のようにディスク inode(ext3_inode)とメモリinode(VFS inode)間において、512B⇔ファイルシステムブロックサイズの変換を行ってからi_blocksの授受を行う。古いカーネルでは当該フラグが設定されているファイルシステムは使用不可能とする（マウントでエラーとする）。

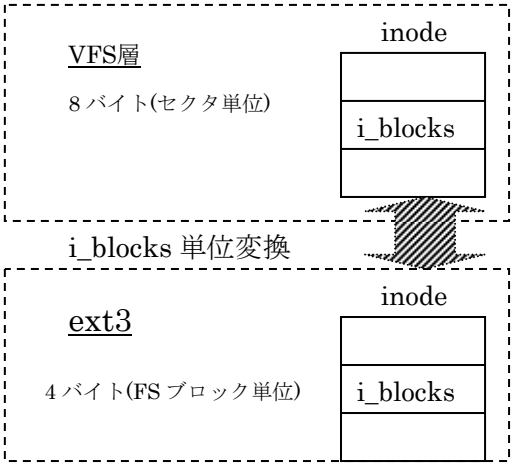

図 7　i_blocks の単位変換

上記の改造により最大ファイルサイズは表 6のように拡大される。

ブロックサイズ	改造前	改造後
4KB	2TB	4TB
64KB	ー	256TB

表 6　ファイルサイズ拡大結果

5.　メーリングリスト[1][2]上での議論

4で述べた新しい改造方式は主に以下の内容のパッチセットに分割して上記メーリングリストに提案済である。

A)　VFS inode 内の i_blocks 8バイト化

2005 年 12 月 7 日にパッチを提案している[8]。

i_blocks の型として sector_t を使用している点について、NFS クライアントのメンテナである Trond Myklebust 氏から「sector_t はファイルシステムのブロック数用の型であり、ファイルブロック用に使用すべきではない」との指摘があった[9]。

そのため、i_blocks 用に新しい型 blkcnt_t を追加しコンフィグレーションパラメータ CONFIG_LSF によって 8 バイトと 4 バイトのサイズを切り替えるようにした。

本改造は Andrew Morton 氏 により、mm-tree に 2006 年 1 月 13 日に 2.6.15-mm4 で組み込まれ、その後特に問題は報告されず、2006 年 3 月 27 日に 2.6.17-rc1 に正式に取り込まれた[10]。

B)　ブロックサイズ拡大

2006 年 1 月 18 日にパッチを提案している[11]。

Andreas Dilger 氏より「ディレクトリの先読みブロックの数を増加すべき」との指摘を受けたが、ディレクトリ先読みロジックの改善検討を Andrew Morton 氏が先行して実施しているとのことで、我々は特にディレクトリ先読みに関して作業を行わないことにした[12]。

その他、特に問題点や実装に関する反対意見はない。

C)　ファイルシステムサイズ拡大

2006 年 3 月 15 日にパッチを提案している[13]。

ブロック用変数の型に”unsigned int”等の型を直接使用している点について、Andreas Dilger 氏から「将来的に 4G 以上のブロック数をサポートする際に改造を簡単にするため、ブロックの種別毎に以下のような型を追加したほうがよい」との改善提案があった[14]。

```
// ファイルシステム相対ブロック番号
typedef unsigned long ext3_fsblk_t;
// ファイル相対ブロック番号
typedef unsigned long ext3_fileblk_t
// ブロックグループ相対ブロック番号
typedef long ext3_grpblk_t;
// ファイルシステムブロック数
typedef unsigned long ext3_fscnt_t;
```

我々としては上記方針に関しては賛成であるが、ext3_fscnt_t はファイルシステムブロック数、ファイルブロック数の両方で共通使用できるように提案する予定である。

D) ファイルサイズ拡大

2006 年 3 月 18 日にパッチを提案している[15]。

現在、Andreas Dilger 氏から「2TB 以上のファイルのみセクタ⇔FS ブロックサイズの変換を行うようにしたほうがよい」との改善の提案があった[16]。今後実装を検討する予定である。

6. まとめと今後の課題

本論文で述べている我々が検討した新しい実装方式は現状の ext3 のロジックに殆ど影響が無く、安定性を維持しつつ最大ファイルシステムサイズ、最大ファイルサイズを拡大できるメリットがある。また、今回の改造方式の基盤となる VFS inode 内の i_blocks の 8 バイト化を行う改造は、2.6.17-rc1 から Linux に組み込まれている。

今後残りの改造内容が Linux に組み込まれるように、前述のメーリングリスト上で ext3 の技術者との議論を継続する。

また、本開発により拡張された ext3 において今後以下の課題の解決が必要と考えられる。

- **ブロック検索の性能向上**

  ブロック数が増大することによりブロック検索性能の向上が必要となるので、遅延アロケーション、デフラグメント等のブロック配置最適化機能が必要になると考えられる。

- **ディスク使用効率改善**

  ブロックサイズを拡大することによるディスク使用効率の悪化を防止するため、テイルパッキング機能（1 ブロックに複数ファイルの末尾データを格納する機能）を検討する必要がある。

- **さらなる諸元の拡大**

  今後さらにディスクストレージ容量が増大することが予測されるため、さらなる諸元の拡大が必要になると考えられる。

  その場合は、本格的な ext3 の 64bit 化を検討する必要がある。

7. 参考文献

[1] Linux カーネルに関する開発用メーリングリストのアーカイブ

http://marc.theaimsgroup.com/?l=linux-kernel

[2] ext3 開発用メーリングリストのアーカイブ

http://marc.theaimsgroup.com/?l=ext2-devel

[3]「国内ディスクシステム市場の中期予測

を発表」
IDC Japan のサイトより
http://www.idcjapan.co.jp/Press/Current/20050705Apr.html

[4] Linux Kernel Source
http://www.kernel.org/

[5] Daniel P. Bovet, Marco Cesati
詳解 Linux カーネル 第２版 (2003)

[6] Goldwyn Rodrigues
[PATCH] Pushing ext3 file size limits beyond 2TB
http://marc.theaimsgroup.com/?l=linux-kernel&m=108936038122036&w=1

[7] Laurent Vivier
Ext2/Ext3 improvement project
http://www.bullopensource.org/ext4

[8] Takashi Sato
RE: stat64 for over 2TB file returned invalid st_blocks
http://marc.theaimsgroup.com/?l=linux-kernel&m=113395320603205&w=2

[9] Trond Myklebust
Re: stat64 for over 2TB file returned invalid st_blocks
http://marc.theaimsgroup.com/?l=linux-kernel&m=113391723427409&w=1

[10] Linux に取り込まれたパッチの commit log
- stat64 の不具合修正
http://www.kernel.org/git/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=abcb6c9fd13fc2ad7757b818924dc8109a0e3775
- ブロック数用の型 blkcnt_t 追加
http://www.kernel.org/git/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=a0f62ac6362c168754cccb36f196b3dfbddc3bc3
- statfs64 の不具合修正
http://www.kernel.org/git/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=e2d53f9525790dfacbcf09f359536311d3913d98

[11] Takashi Sato
[PATCH] ext3: Extends blocksize up to pagesize
http://marc.theaimsgroup.com/?l=linux-kernel&m=113758968325136&w=2

[12] Andrew Morton
Re: [Ext2-devel] [PATCH] ext3: Extends blocksize up to pagesize
http://marc.theaimsgroup.com/?l=linux-kernel&m=113782762910045&w=1

[13] Takashi Sato
[Ext2-devel] [PATCH 1/2] ext2/3: Support 2^32-1 blocks(Kernel)
http://marc.theaimsgroup.com/?l=ext2-devel&m=114242642113864&w=2

[14] Andreas Dilger
Re: [Ext2-devel] [RFC][8/21]ext3 modify variables to exceed 2G
http://marc.theaimsgroup.com/?l=linux-kernel&m=114494857419905&w=1

[15] Takashi Sato
[Ext2-devel] [PATCH 1/4] ext2/3: Extends the max file size(ext2 in kernel)
http://marc.theaimsgroup.com/?l=ext2-devel&m=114268692909004&w=2

[16] Andreas Dilger
Re: [Ext2-devel] [PATCH 1/4] ext2/3: Extends the max file size(ext2 in kernel)
http://marc.theaimsgroup.com/?l=linux-kernel&m=114288361509138&w=1