



Linux Conference 2005

Linux 用ログ構造化ファイルシステム

天海 良治 一 二 三 尚 ○小西 隆介
佐藤 孝治 木原 誠司 盛合 敏

日本電信電話株式会社 NTTサイバースペース研究所
OSSコンピューティングプロジェクト



発表の流れ

- ▶ 背景
- ▶ 開発目標
- ▶ ログ構造化ファイルシステムについて
- ▶ 設計方針
- ▶ Linux での実装
- ▶ 評価
- ▶ 実装経験
- ▶ まとめ

背景

- ▶ 障害に弱い ext2 (5 年程前)
 - ファイルシステムの信頼性の懸念
- ▶ ext3, XFS, ReiserFS, JFS (現在)
 - ジャーナリング技術採用の FS が充実



信頼性, 堅牢性, 復旧性の改善に大きく貢献

🚀 ジャーナリングファイルシステムで十分？

ジャーナリング FS: 変更するメタデータをジャーナルファイルに前書き, 後消しする方式

- ▶ 信頼性の確保には, 書き込みの順序保証が重要
 - ブロックデバイス層のエレベータシークによる副作用
 - 厳密な順序保証 → ジャーナルファイルとの間でシークが発生
- ▶ ユーザデータに対する障害復旧時の一貫性保証は不完全
 - データジャーナリング (ext3 のオプション) は?
 - 書き出すデータ量が2倍になる
 - ジャーナルファイルとユーザデータの領域間でシークが発生

高い信頼性を実現し, かつシークによる性能低下を抑えられる解はないか？

もう一つの選択肢：LFS

信頼性向上のアプローチとしてログ構造化ファイルシステム (Log-structured File System)[Rosenblum91] に着目

▶ ディスクブロックの書き込みを上書きではなく追記で実現

- 信頼性と性能が両立可能
- 高速なスナップショット機能が実現可能

▶ ローカルFSでの実装例は少ない

- Linux 2.2 時代に Linlog-FS があったが、実験段階で開発停止

nilfs (New Implementation of the Log-structured File System) を開発



開発目標

▶ 高信頼であること

- 壊れにくく、ファイルシステムでデータベース並みの信頼性をもつ

▶ 高可用であること

- 停止時間が短く、障害発生後の復旧時間が短い

▶ 運用性に優れること

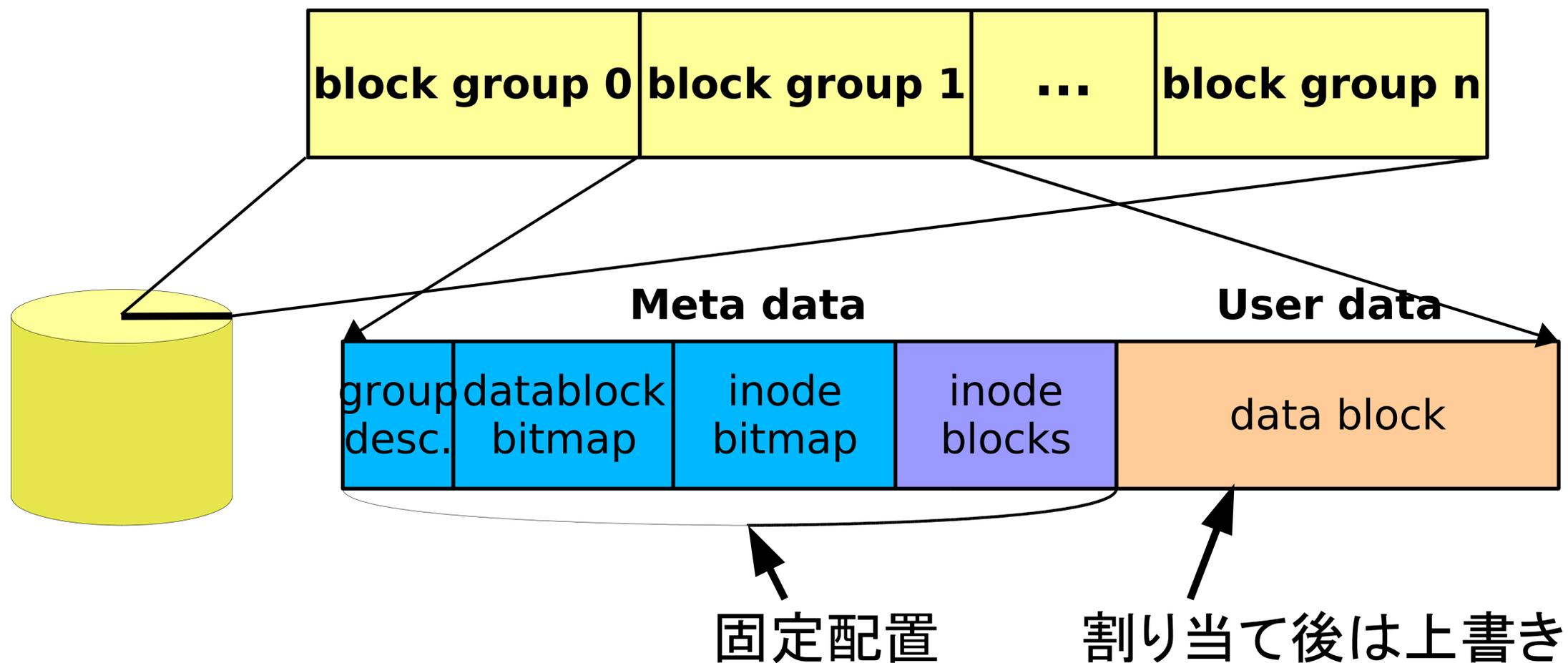
- オンラインバックアップ、誤消去したファイルの復旧を可能にするため、短時間にスナップショットがとれる

▶ 高性能であること

- ext3 に同等もしくはそれ以上の性能

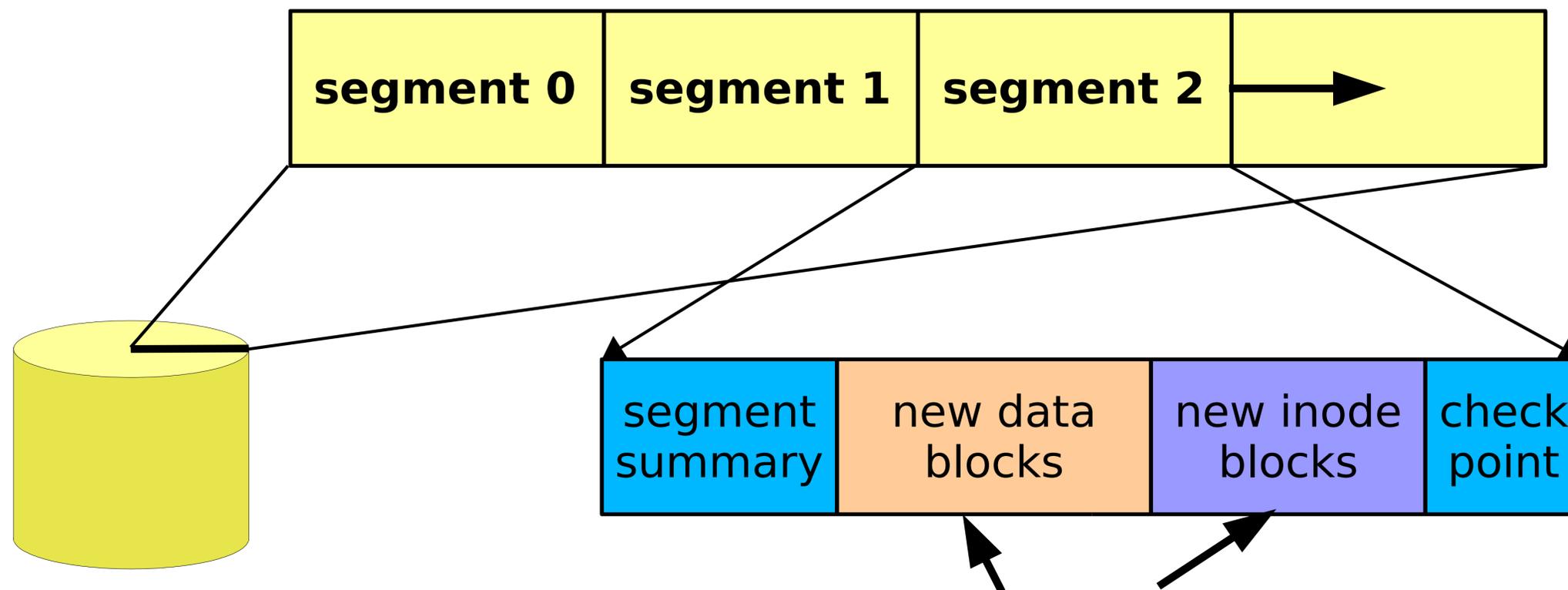
従来ファイルシステムのブロック配置

- ▶ inode を含むメタデータは固定割り付け
- ▶ データブロックは、最初の write 時に割り当て、以後上書き
 - ⇒ 上書きは破壊： 過渡的に整合性のない中間状態が発生
 - ⇒ ジャーナリングFS： ジャーナリングファイルにも更新ブロックを書き、復旧不可能な中間状態をなくす



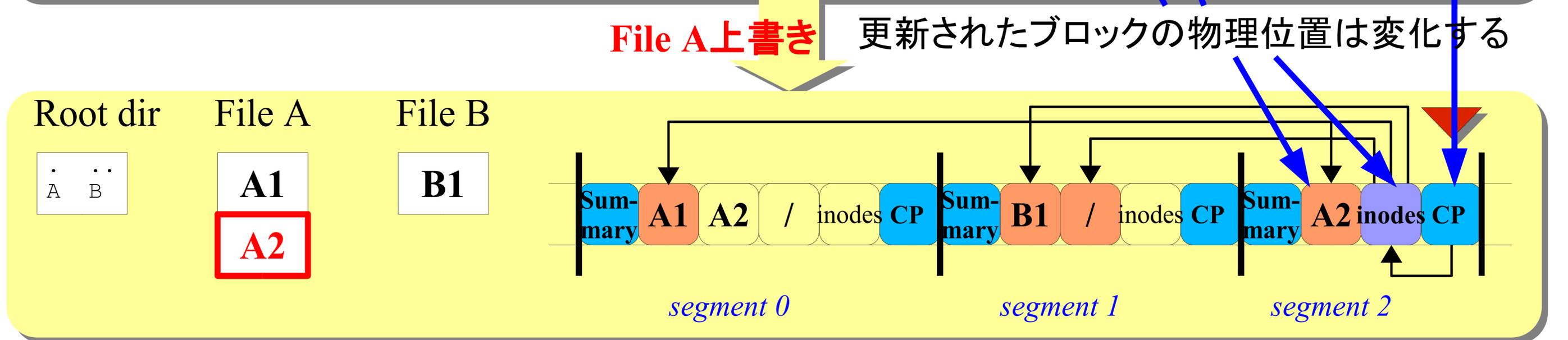
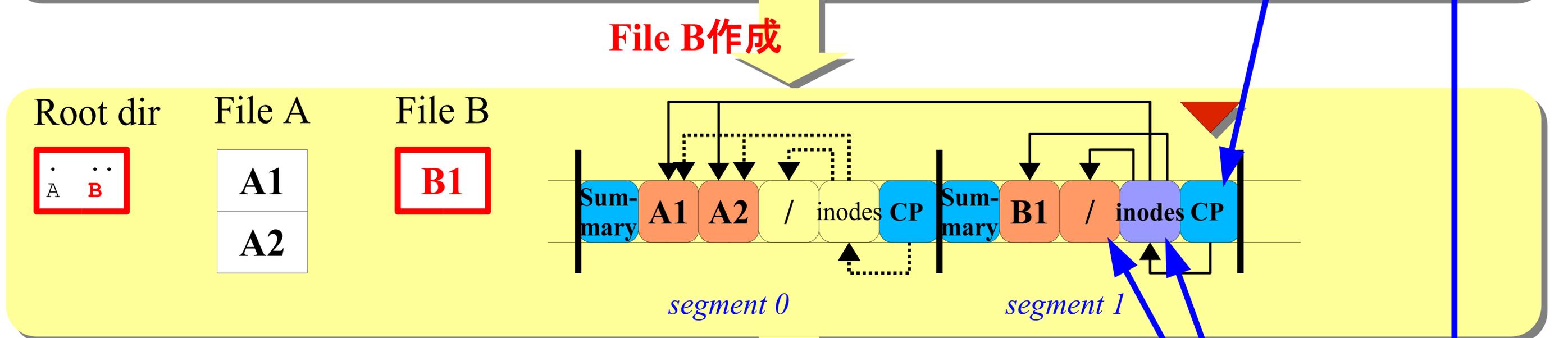
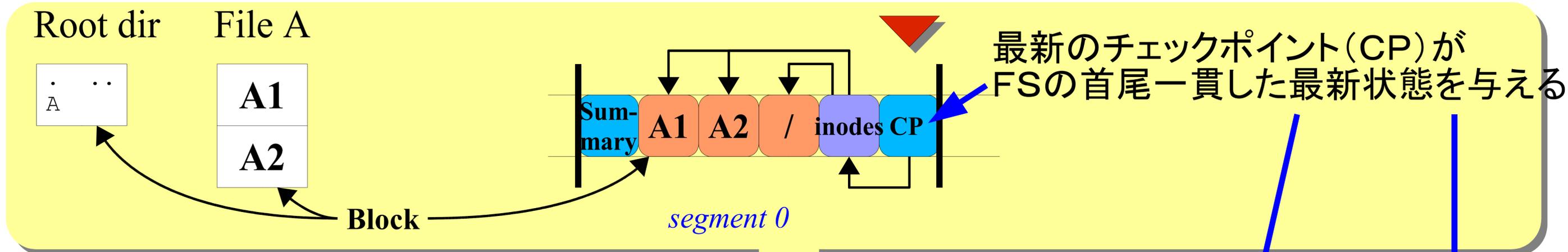
LFS のブロック配置

- ▶ 復旧不可能な中間状態の発生を追記により回避
 - inode, データブロックとも, 常に新しい領域に書き出す
 - 書き出しはセグメントという単位で atomic に実施



データ, inode の変更分を新たなセグメントに書き出す

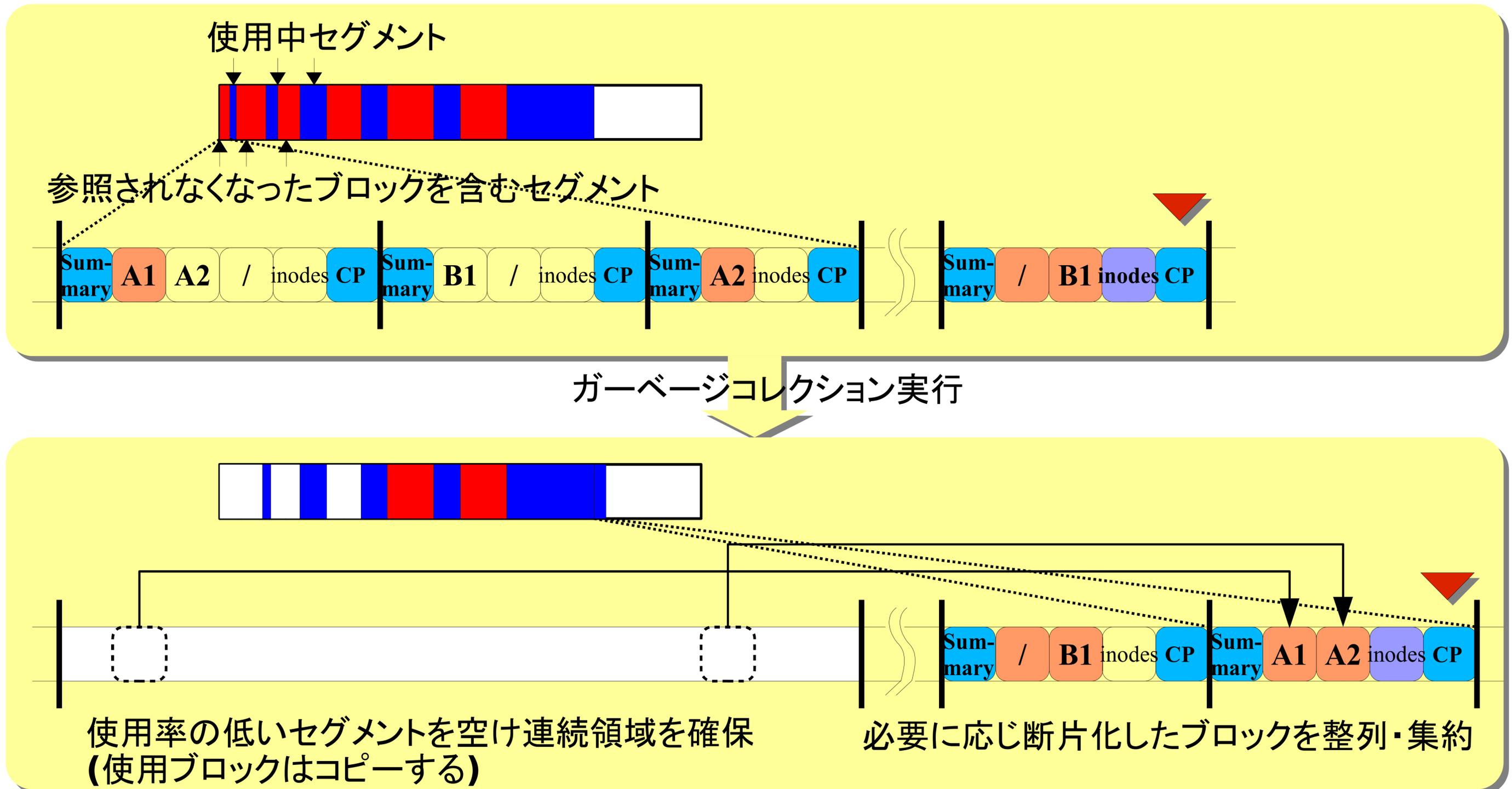
LFS における書き込みの例



Garbage Collection (GC)

書き込みを続けるには、不要ブロックを整理し、連続空き領域を作る処理が必要。

⇒ LFS の性能低下要因

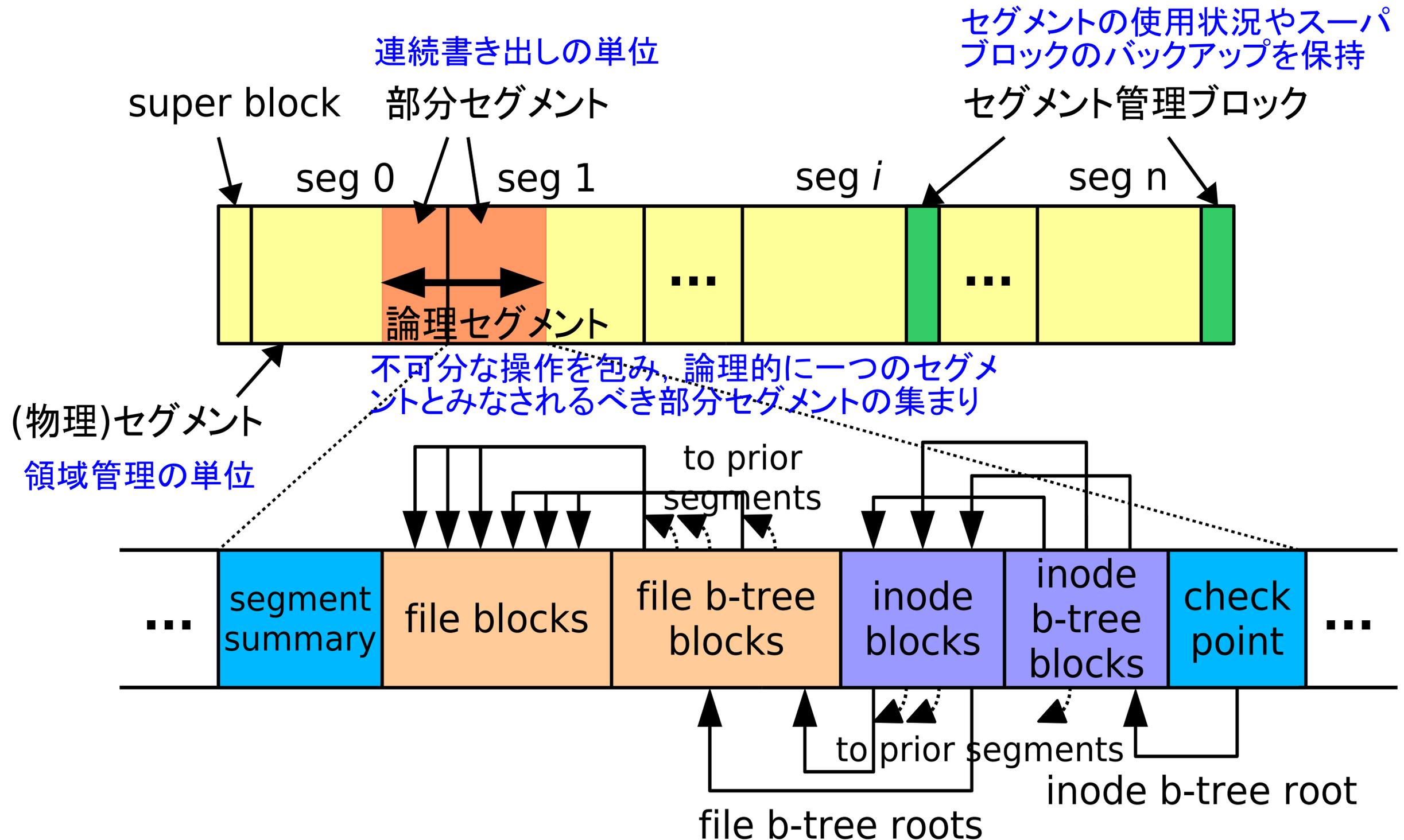




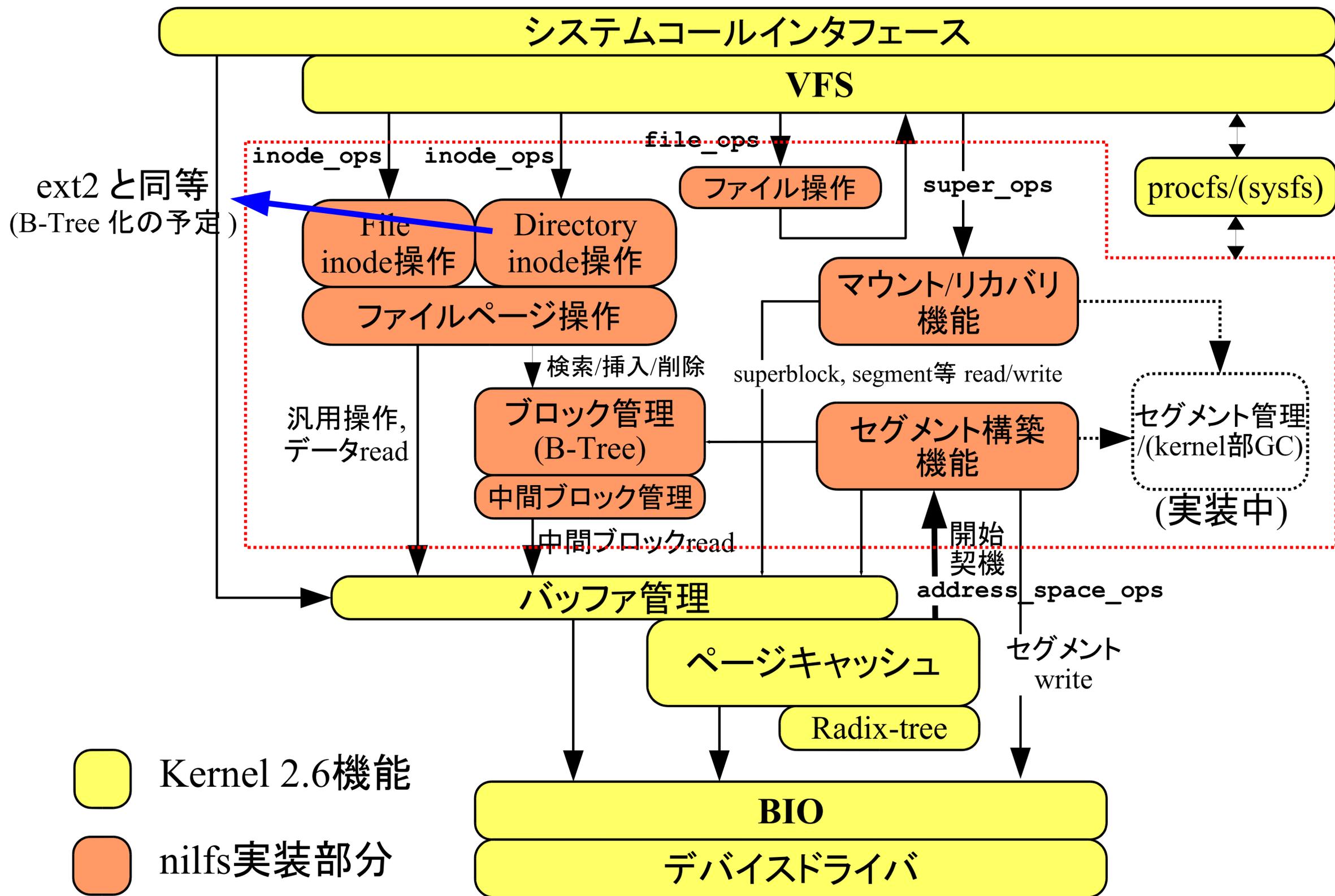
nilfs(v1) の設計方針

- ▶ ファイルブロックと inode を B-Tree (平衡木) で管理
 - 大きなファイル (64 ビットサイズ) や多数 inode の管理・検索の効率化
 - ディスクブロック番号の動的な変化に対応
- ▶ 信頼性の確保
 - チェックサム付与による正しい書き込み完了の確認
 - ディスクブロックの上書きは極力排除
 - BIO 活用による連続書き込みの実装と順序保証
- ▶ その他
 - Linux セマンティクスの踏襲
 - カーネルコードへの変更を避ける (モジュールで導入可能)
 - VFS, ページキャッシュ, バッファ管理等との親和性

nilfs(v1) のディスクレイアウト

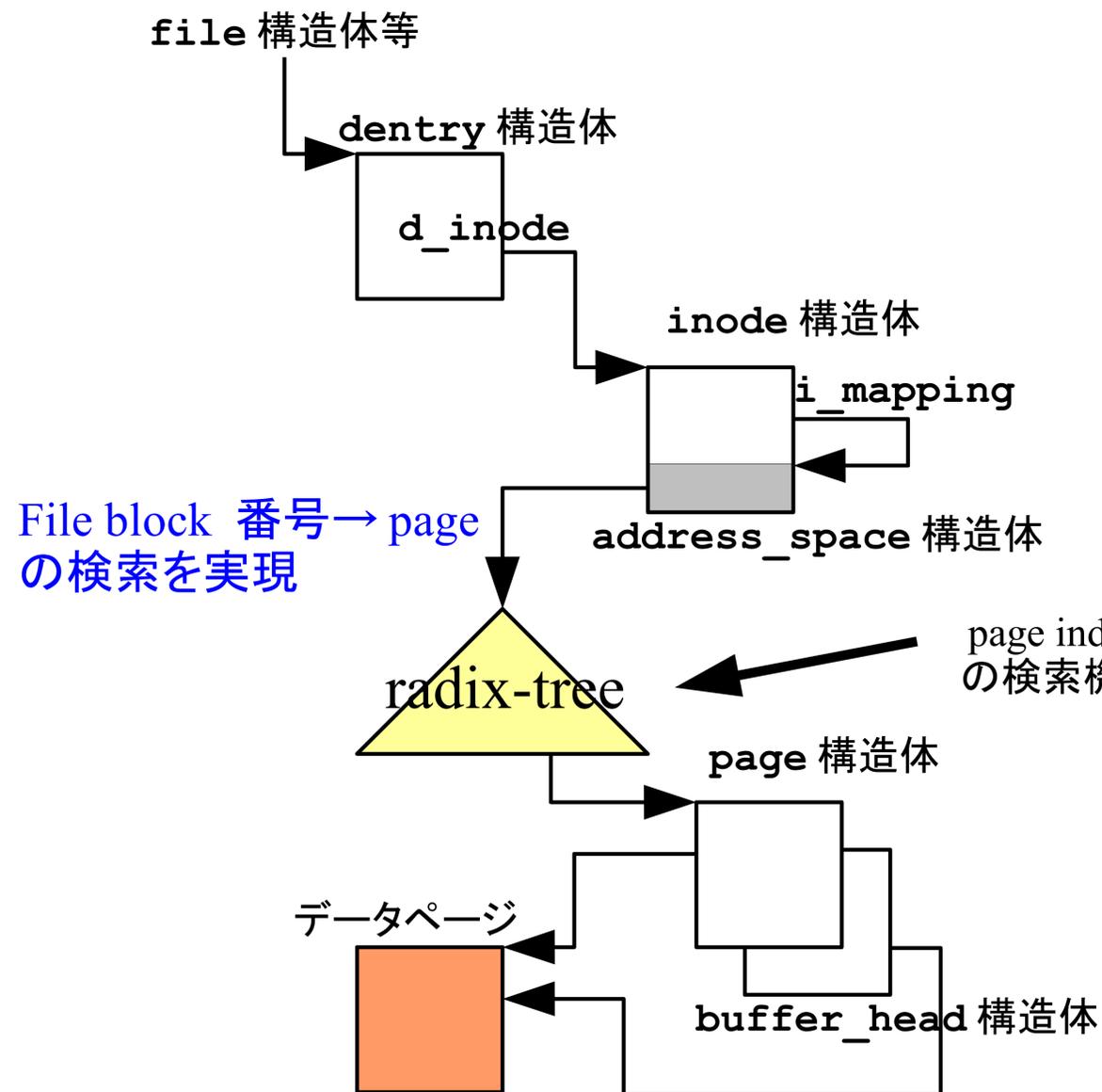


nilfs(v1) の機能構成

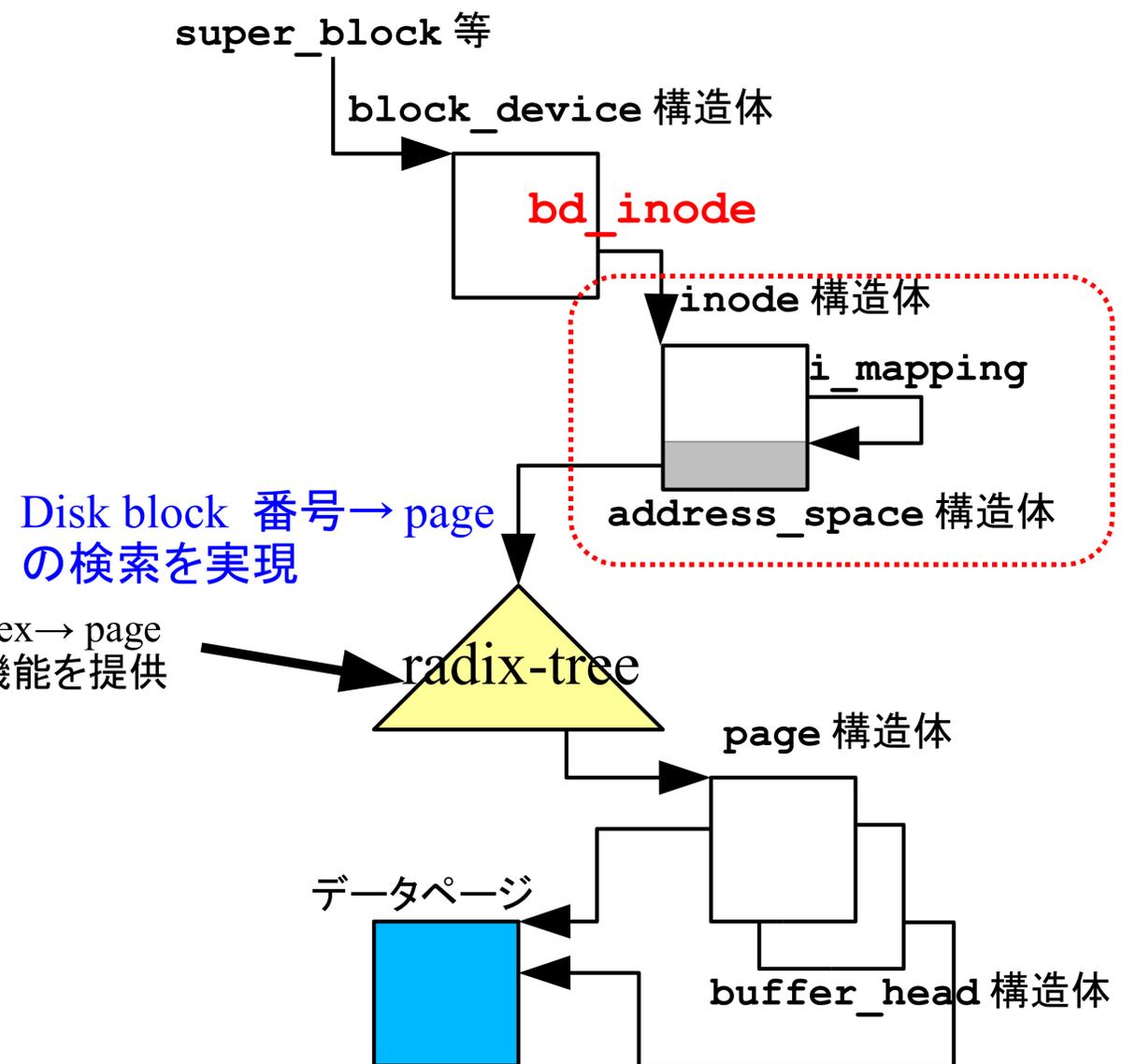


Linux のページキャッシュにおけるバッファの管理

ユーザデータの管理



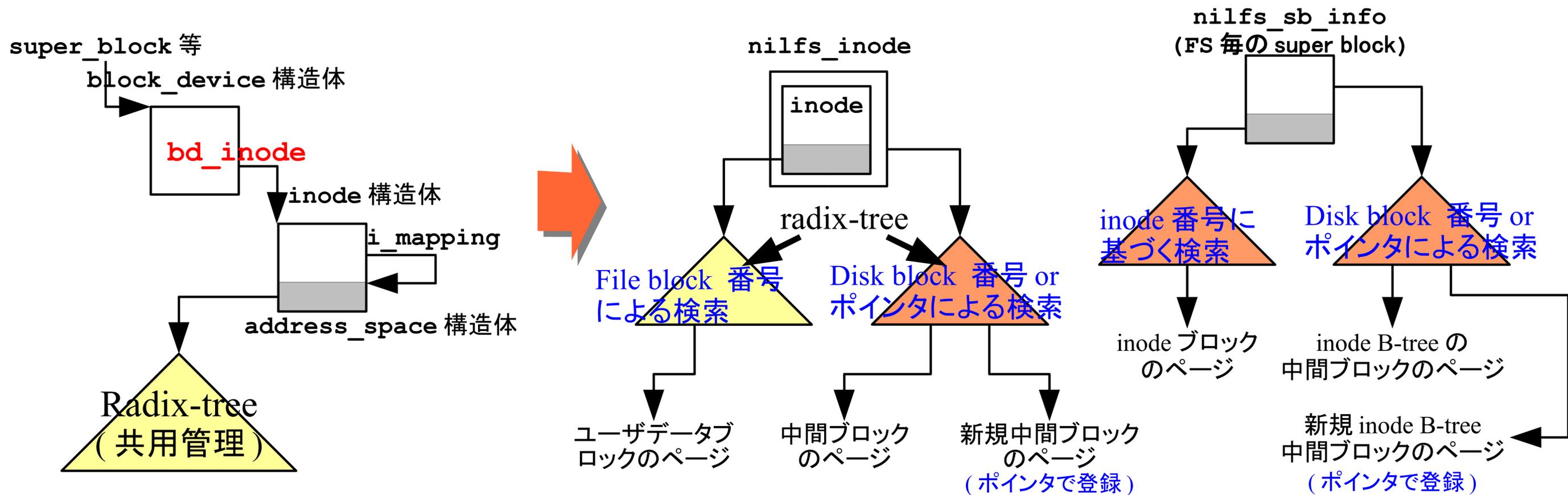
メタデータの管理



- ▶ 間接ブロックを含めメタデータは、ディスクブロック番号をキーに `bd_inode` で一括管理

nilfs におけるバッファの管理

- ▶ B-tree 中間ブロックのバッファ管理の効率化
- ▶ 動的なブロック位置変化への対応
 - ブロックの位置は書き出す直前まで決まらない
 - ファイル, inode のバッファ検索キーは論理的な番号であり, 変更不要
- ▶ Block device オブジェクトに頼らない書き出し制御の実現



ページキャッシュ機能を用いつつ, bd_inode を使わず個別に管理



スナップショット機能

- ▶ 各チェックポイントが、それぞれ一貫したスナップショットを与える
 - チェックポイント作成機能 + GC からの保護機能として実装
 - チェックポイントの管理は通常ファイルで行う予定
 - ディスク容量を消費する
 - 保全するデータの絞り込み, 追い出し等の工夫は今後

- ▶ デモンストレーション

性能測定について

▶ 測定項目 (iozone)

- シーケンシャルライト・ランダムライトのスループット
- シーケンシャルリード・ランダムリードのスループット

▶ 測定条件

- Kernel Version: 2.6.10
- Ext3 Journaling Mode: Ordered mode (Default)
- nilfs は GC 未実装. 性能チューニング未実施.

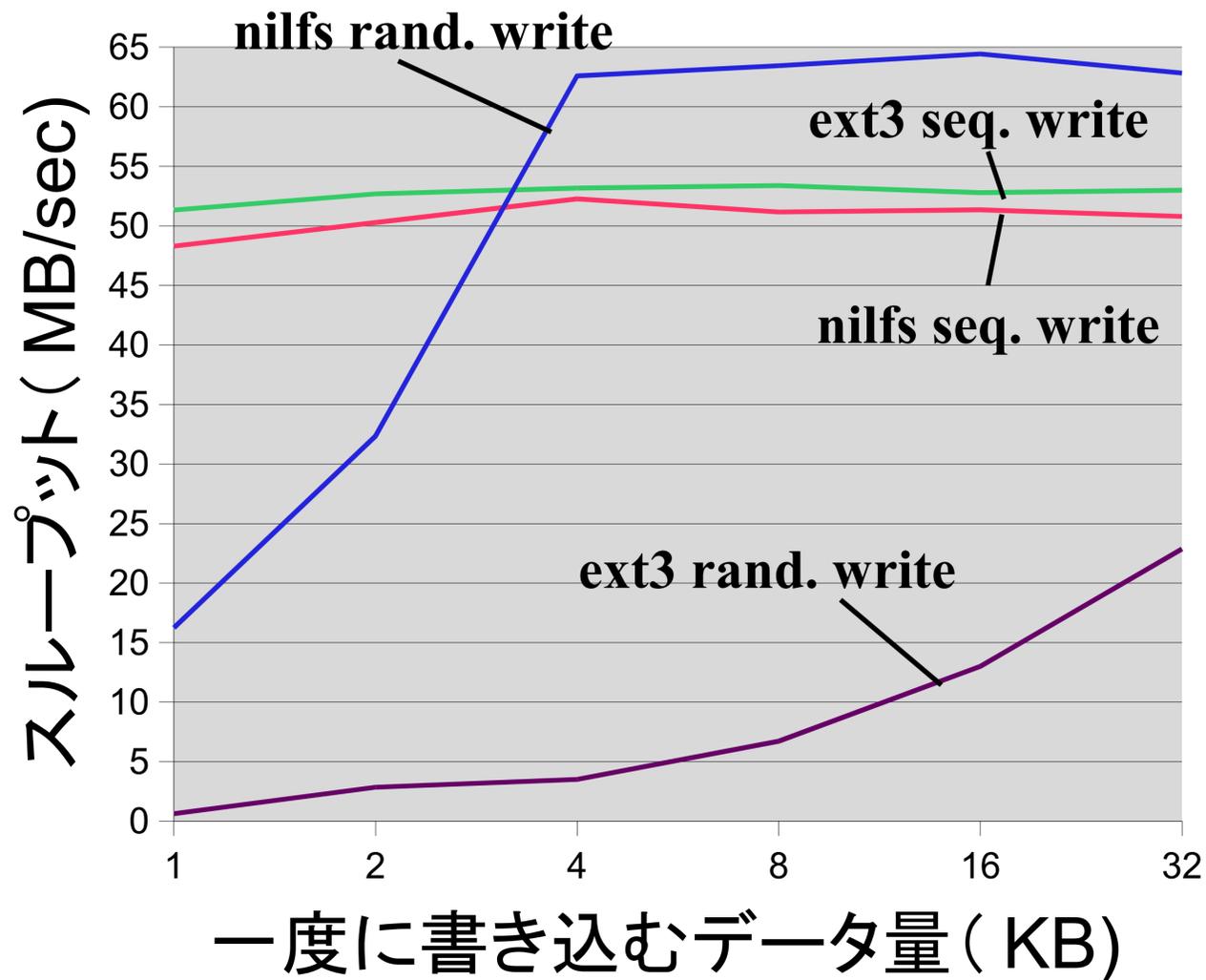
測定マシンのスペック

CPU	Pentium 4 3.0GHz
メモリ	1GBytes
ディスク	IDE Disk (ATA 7,200 rpm)

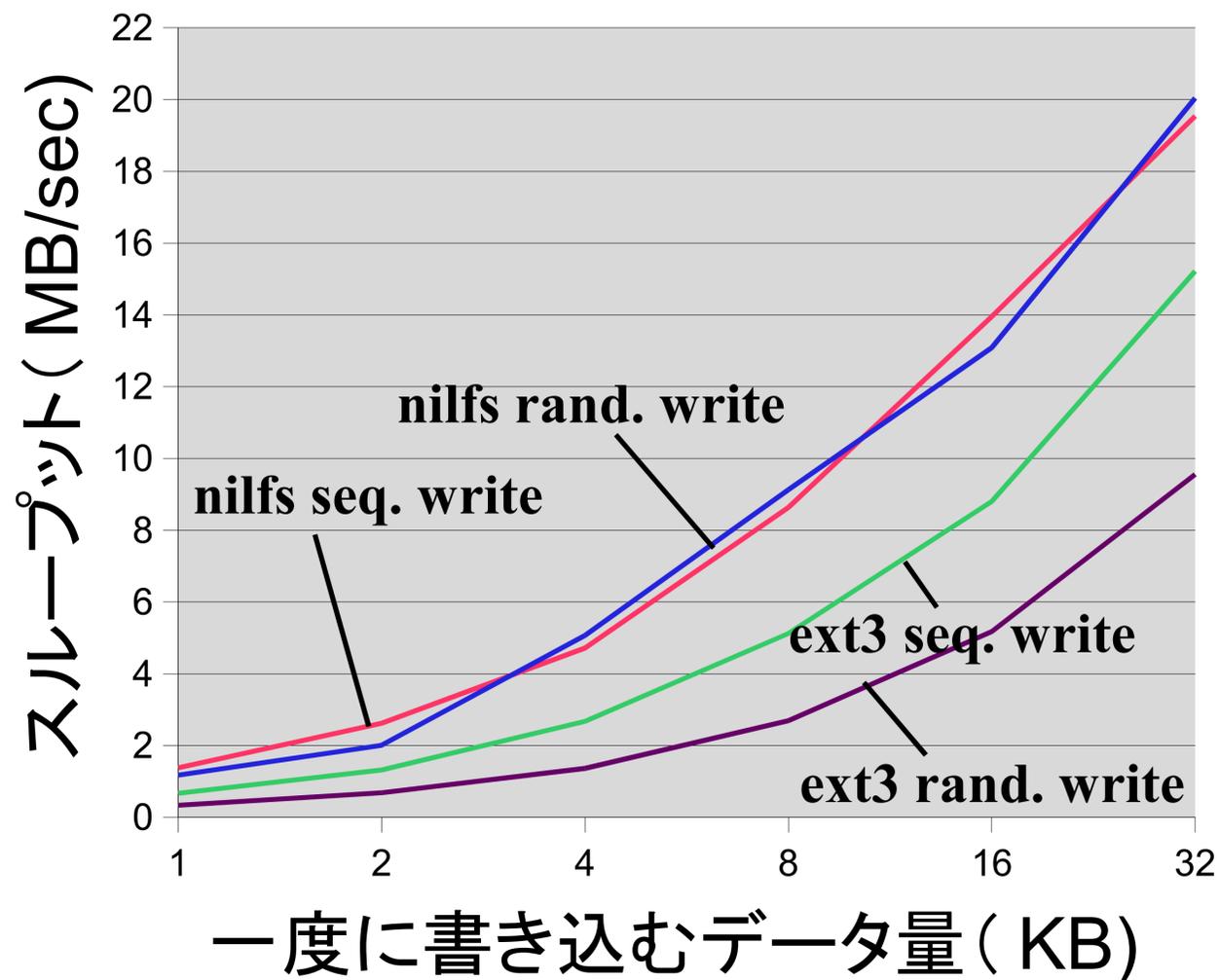


書き込みのスループット

write + fsync



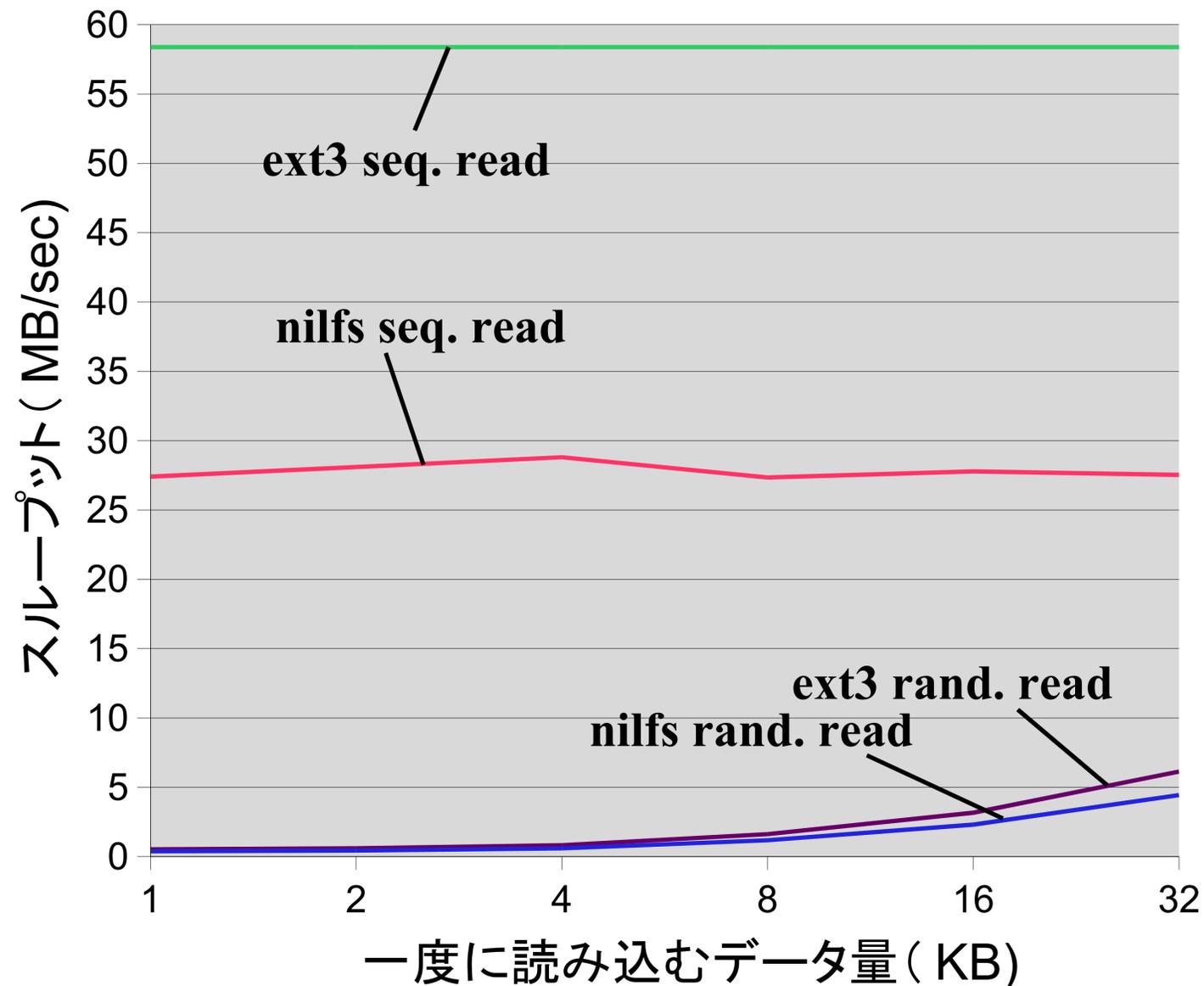
write with O_SYNC



- 連続書き込み (ext3)
- ランダム書き込み (ext3)
- 連続書き込み (nilfs)
- ランダム書き込み (nilfs)

- ランダムライトは上書き
- シーケンシャルライトは新規作成
- ファイルサイズは 512MB

読み込みのスループット



- ファイルサイズは 512MB
- 先読み設定はデフォルト (read_ahead_kb=128kb, hdparm read_ahead=256kb)
- キャッシュの影響を排除するため、ファイル作成後 umount/mount を実施

B-Tree の先読みや、論理セグメントの一定サイズごとの分割による局所化、その他のチューニングを検討中

開発環境

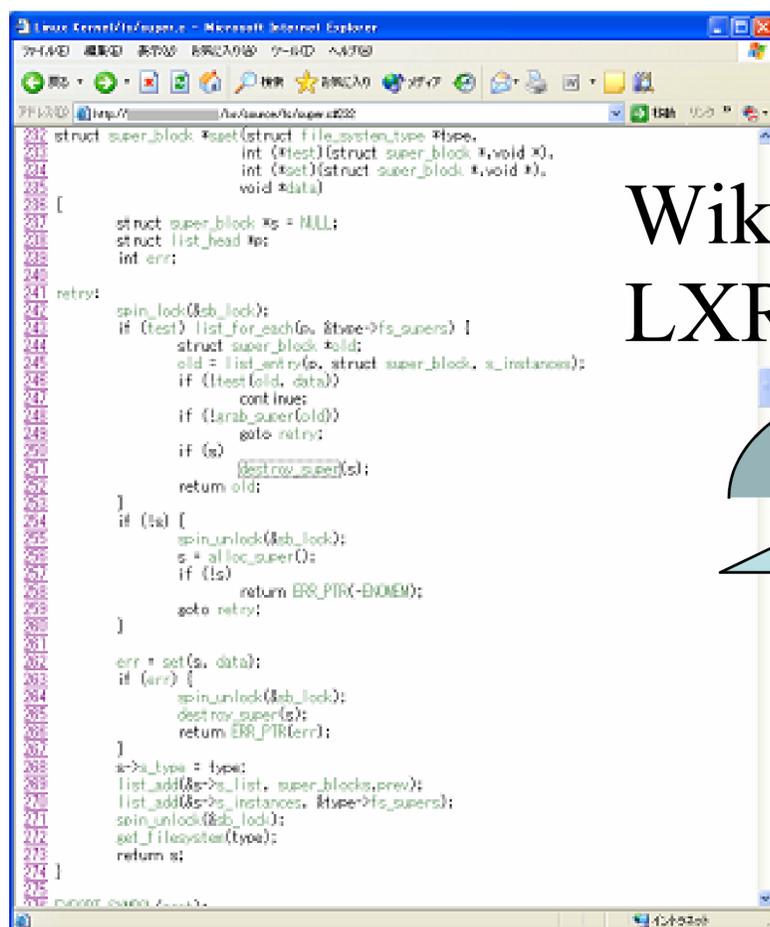
▶ CVS

▶ LXR (LXR Cross Referencer)

- カーネル内部調査や, 関数仕様の確認に必須
- Linux バージョン更新の追従に必須

▶ Pukiwiki

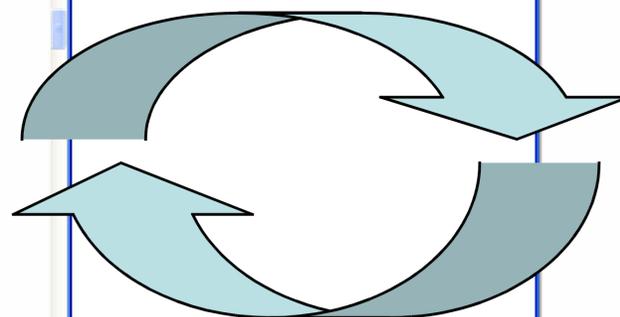
- 検討仕様の整理, ノウハウ蓄積, bugtrack に活用



```
struct super_block *set(struct file_system_type *type,
                        int (*test)(struct super_block *,void *),
                        int (*set)(struct super_block *,void *),
                        void *data)
{
    struct super_block *s = NULL;
    struct list_head *list;
    int err;

    retry:
    spin_lock(&sb_lock);
    if (!test) list_for_each(list, &type->fs_supers) {
        struct super_block *old;
        old = list_entry(list, struct super_block, s_instances);
        if (!test(old, data))
            continue;
        if (!is_sb_super(old))
            goto retry;
        if (!is)
            destroy_super(old);
        return old;
    }
    if (!is) {
        spin_unlock(&sb_lock);
        s = alloc_super();
        if (!s)
            return ERR_PTR(-ENOMEM);
        goto retry;
    }
    err = set(s, data);
    if (err) {
        spin_unlock(&sb_lock);
        destroy_super(s);
        return ERR_PTR(err);
    }
    s->type = type;
    list_add(&s->list, super_blocks.prev);
    list_add(&s->instances, &type->fs_supers);
    spin_unlock(&sb_lock);
    get_filesystem(type);
    return s;
}
```

Wiki との連携により
LXR を補完



get_new_inode

- 機能
 - o ノードを取得し, 新たに割り当てたときは初期化する.
- 定義
 - o static struct inode * get_new_inode(struct super_block *sb, struct hlist_head *head, int (*test)(struct inode *, void *), int (*set)(struct inode *, void *), void *data)
- 引数
 - o sb: スーパーブロックオブジェクトへのポインタ
 - o head: ノードハッシュテーブルのエントリ(リスト)の先頭
 - o test: ノード間の比較のためのコールバック関数へのポインタ
 - iget5_locked()がbdi_get()から呼び出されているときは bdev_test()
 - o set: 新しいノードを初期化するためのコールバック関数へのポインタ
 - iget5_locked()がbdi_get()から呼び出されているときは bdev_set()
 - o data: test()とset()に渡されるデータへのポインタ
 - iget5_locked()がbdi_get()から呼び出されているときはデバイス番号へのポインタ
- 戻り値
 - o 成功したときはノードオブジェクトへのポインタ, 失敗したときはNULL
- 参照元
 - o iget5_locked()
- 参照先
 - o alloc_inode()
 - o spin_lock()
 - o find_inode()
 - o set()
 - o list_add()
 - o hlist_add_head()
 - o spin_unlock()
 - o is_sb()



デバッグ環境

▶ kgdb

- Linux カーネルのバージョンに追従せず

▶ qemu (emulator)

- spinlock のデッドロック解析には役立った
- SMP に起因する問題, タイミング問題などに適用できない

▶ printk()

- 大量のメッセージ出力により, 欠落が発生
- /proc/kmsg の cat , リダイレクションが良好



▶ 典型的なバグ

- 参照カウンタの増減ミスによるメモリリーク, メモリ逼迫
- スピンロックやセマフォのデッドロック
 - put/get, lock/unlock 系操作をマクロでデバッグ版に置換し, 網羅チェック
 - スピンロックは正しい使い分けが重要 (spin_lock, spin_lock_irq, spin_lock_irqsave,...)

▶ はまった落とし穴

- inode からブロックサイズを得るには i_blksize ではなく, i_blkbits を使う (i_blksize は Disk I/O の最適サイズ)
- alloc_page() で得たページは __free_page() で返す. free_page() は __get_free_page() で得たページの解放用 (一ヶ月リークがとれませんでした. やめてよね...)



まとめ

- ▶ Linux ログ構造化ファイルシステム nilfs を開発
 - 信頼性・可用性と性能の両立に期待
 - 追記性を活かした高速なスナップショット機能を搭載
 - ジャーナールFSでないもう一つの選択肢を提供
- ▶ nilfs v1 を GPL で近日公開予定



今後の課題

- ▶ GC 実装
- ▶ バグ取り / チューニング
 - ディレクトリの B-Tree 化
 - B-Tree 対応先読み
 - シークを 1 回でも減らすならなんでも ...
- ▶ ユーザインタフェース
 - fsck
 - スナップショット関連 (時間軸ツール : tls, tdiff, tgrep...)
- ▶ スナップショットの動的拡大 / 縮小
- ▶ GRUB 対応
 - スナップショットからのブート
- ▶ スナップショットを含んだバックアップ
- ▶ df, ディスクフル問題
- ▶ セキュリティ
- ▶ ...



『Linux 用ログ構造化ファイルシステム』

ご清聴ありがとうございました。