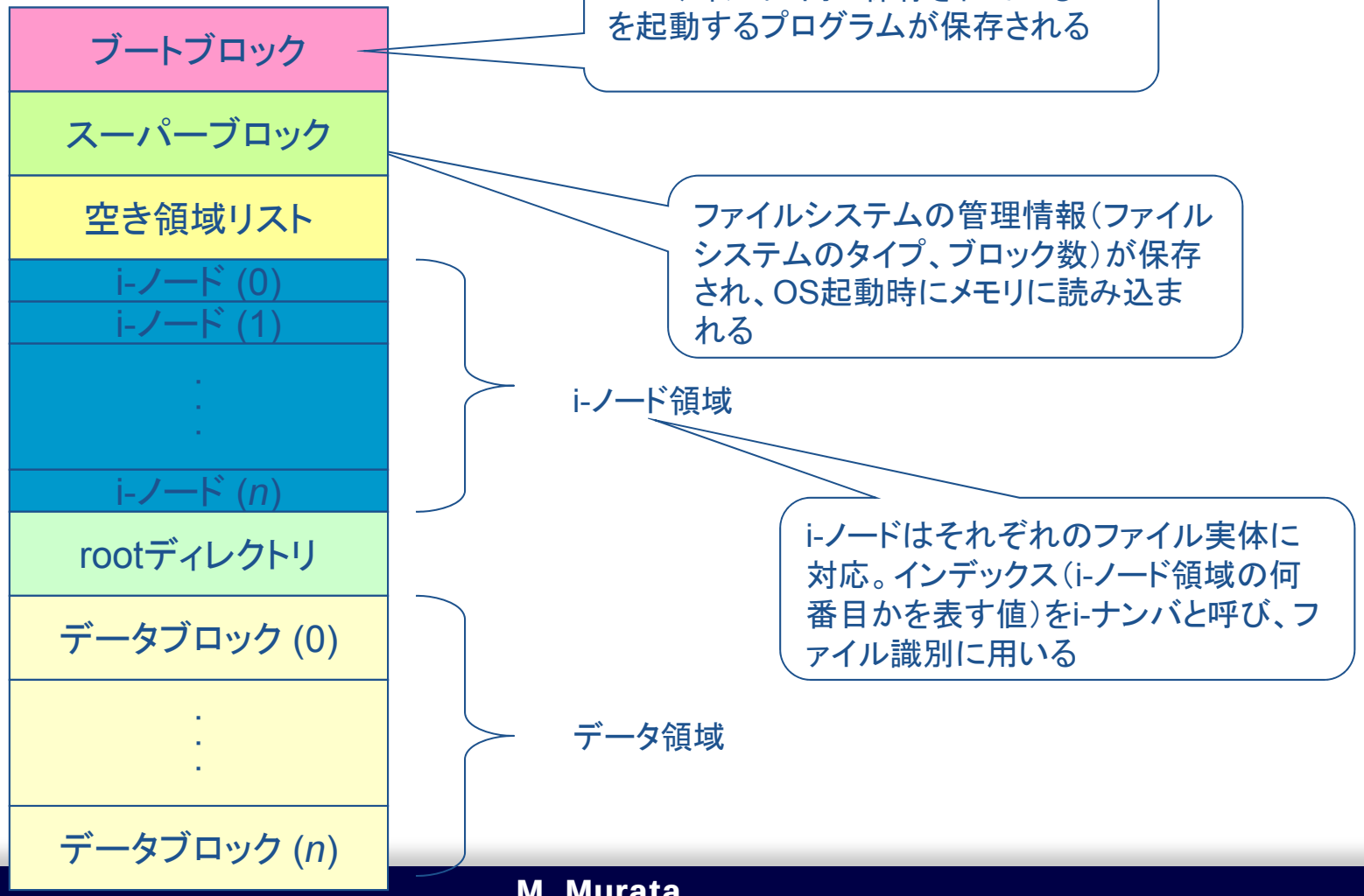
オペレーティングシステム
3章 メモリ管理
3.3 ファイル管理ーファイルシステムー
（UNIX実装の例）

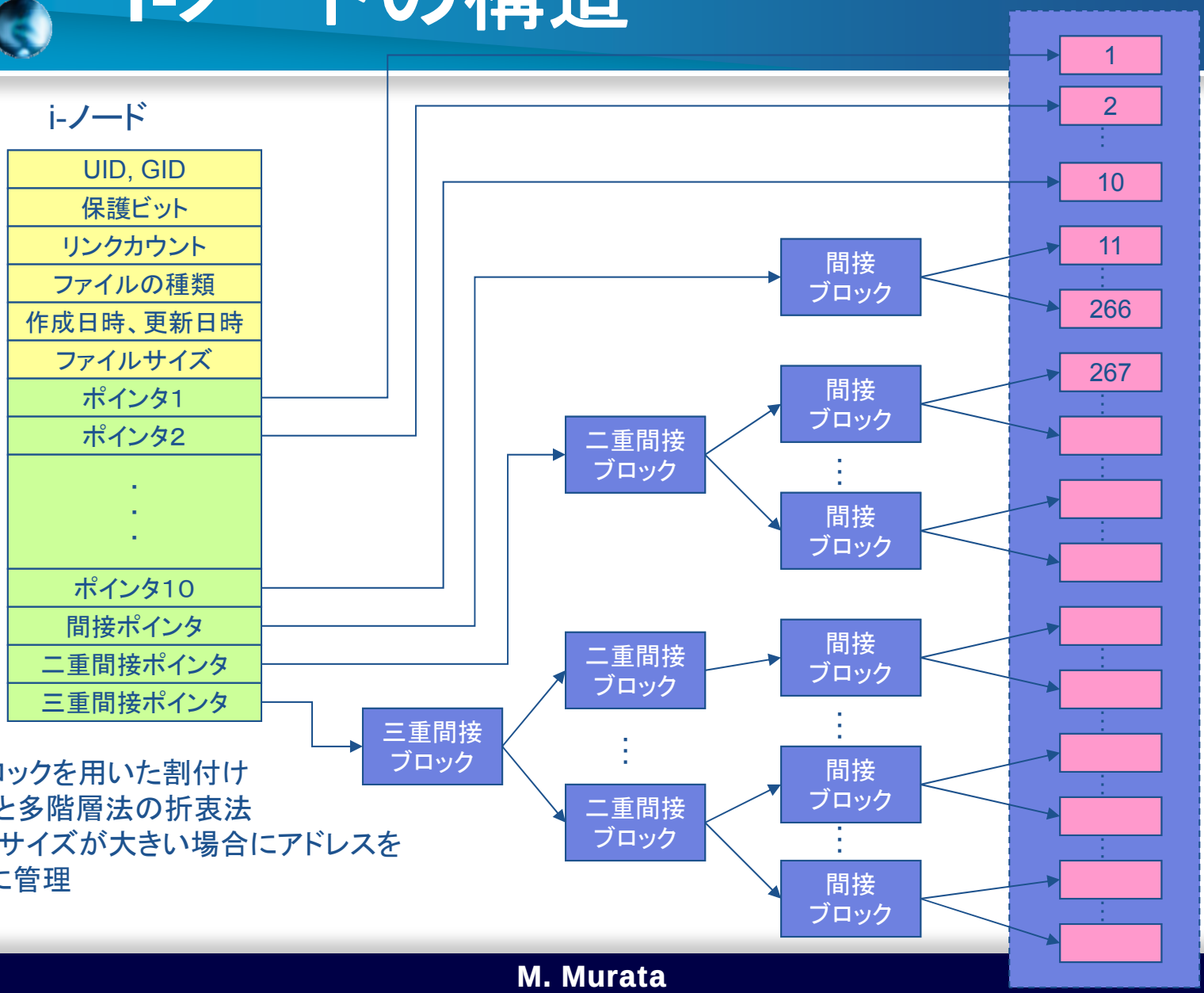
大阪大学大学院情報科学研究科
村田正幸
murata@ist.osaka-u.ac.jp
http://www.anarg.jp/

UNIXの実装

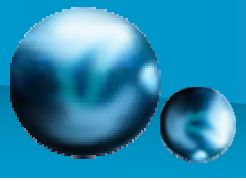
- ファイルシステムの構造



i-ノードの構造



索引ブロックを用いた割付け
リスト法と多階層法の折衷法
ファイルサイズが大きい場合にアドレスを
階層的に管理

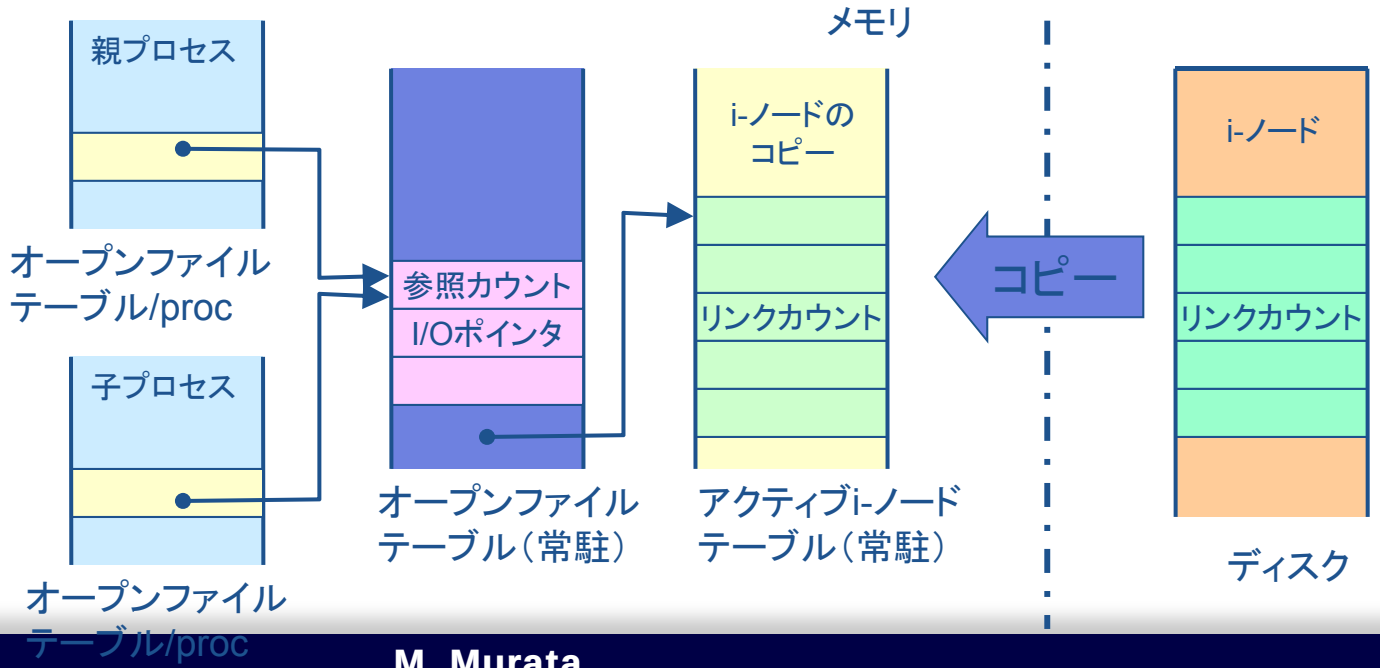


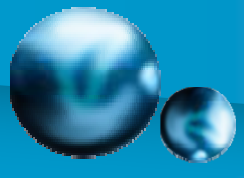
演習問題3.12

- UNIXファイルシステムにおいて以下を仮定する
  - データブロックサイズ：1KB
  - ディスクアドレスは32bit
- 0KBのファイル格納に必要なデータブロックはいくらか？
- 100KBのファイル格納に必要なデータブロックはいくらか？

ディレクトリの構造

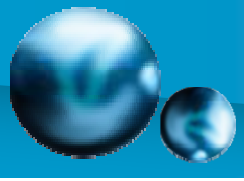
- ファイル名とi-ナンバの対応付けのみ
- 16バイト固定長のディレクトリエントリ
  - ファイル名 14B、i-ノード番号 2B
    - ファイルシステムの制限になる
- 4.4BSD FFS (Fast File System)
  - ディレクトリにファイル名の長さフィールドを持たせ、可変長 (256B)
  - i-ノード番号 4B





ファイルへのアクセス方法

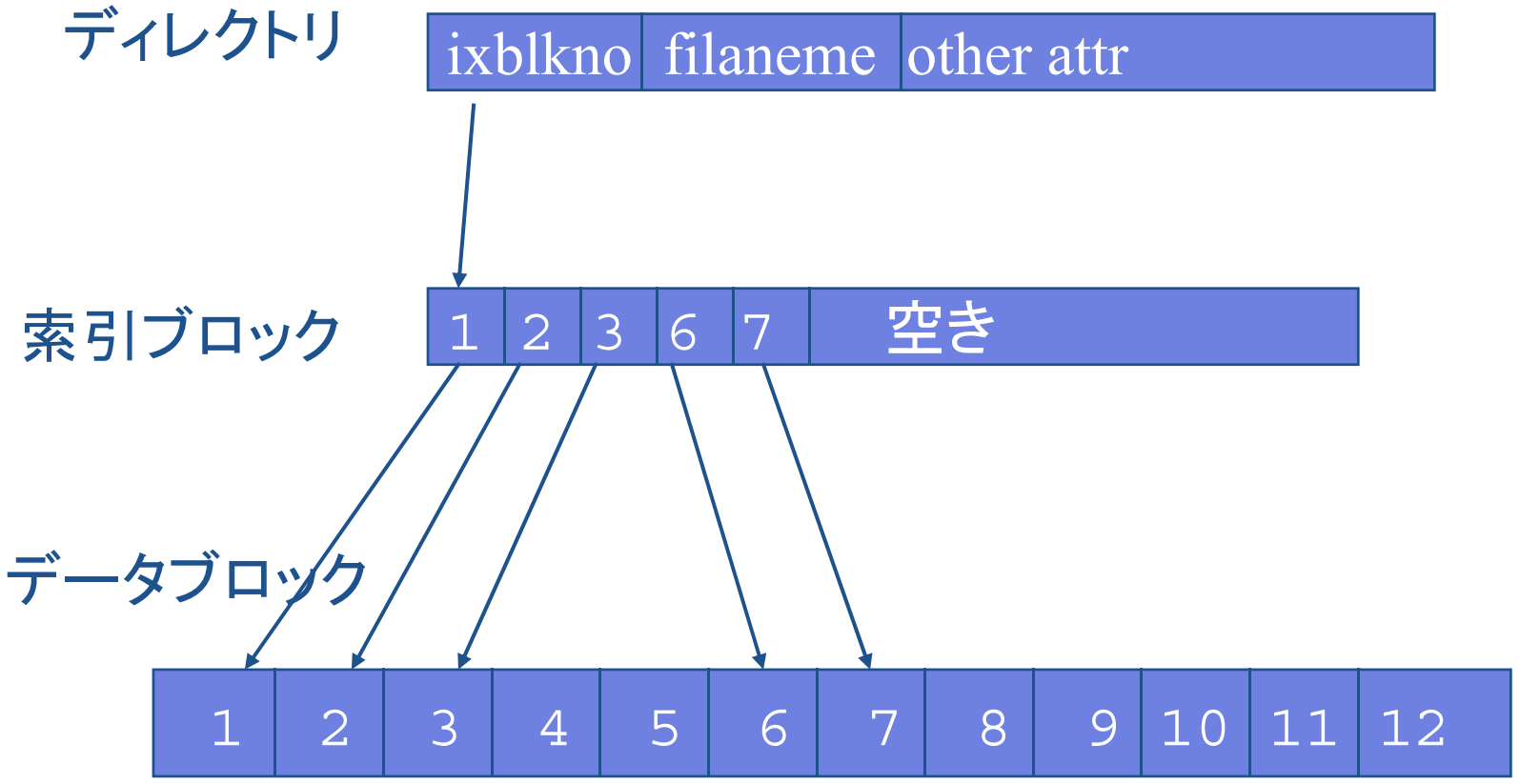
- ファイルのオープン時に、対応するi-ノードがi-ノードテーブル（ディスク内）からアクティブi-ノードテーブル（メモリ常駐）にコピーされる
- i-ノードが更新された場合、ファイルを閉じたときにディスクに書き戻される
  - 30秒ごとのsyncコールの場合も
- オープンファイルテーブル（メモリ常駐）
  - エントリは参照カウント、I/Oカウント、アクティブi-ノードテーブルのエントリへのポインタ



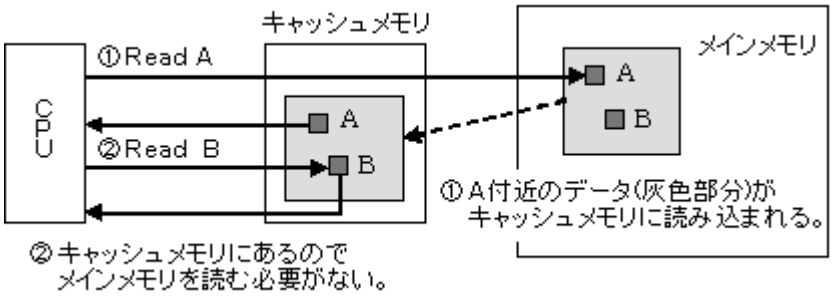
ファイルの生成と削除

- ファイルの生成
  - i-ノードエントリの確保
  - ディレクトリエントリの作成
    - シンボリックリンクの場合は、i-ノードエントリの確保は不要
- ファイルの削除
  - ディレクトリエントリ内のi-ナンバを見つけ、リンクカウントを1減らし、ディレクトリエントリを解放
  - リンクカウントが0で、どのプロセスからも参照されていなければ(オープンファイルテーブルの参照カウンタ)、ファイル実体のディスクブロックとi-ノード自体も解放
    - ファイルがオープン中の場合は、解放をファイルが閉じられるまで遅らせる
    - リンクカウントはディレクトリの一貫性チェックにも用いられる (fsck、scandisk)

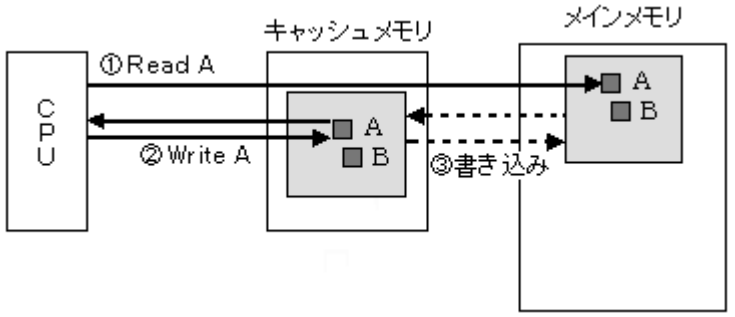
索引ブロック


ディレクトリ
ixblkno
filaneme
other attr
索引ブロック
1
2
3
6
7
空き
データブロック
1
2
3
4
5
6
7
8
9
10
11
12

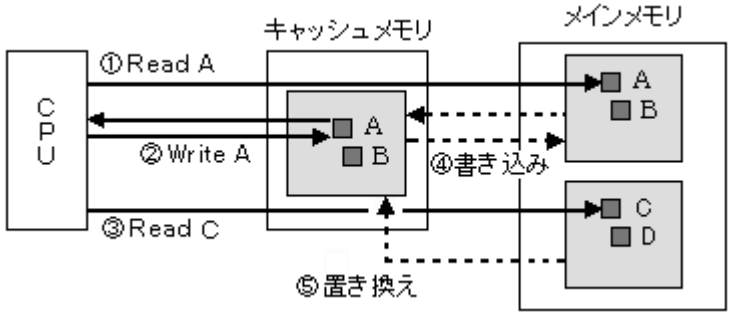
補足：キャッシュメモリの書き込み方式



- 【ライトスルー方式】
  CPU からの書き戻し時に、キャッシュメモリとメインメモリの両方に書き込む方式。書き込み時には、メインメモリとキャッシュに同時に書き込むため、高速化は図れない。読み込み時のみ高速。

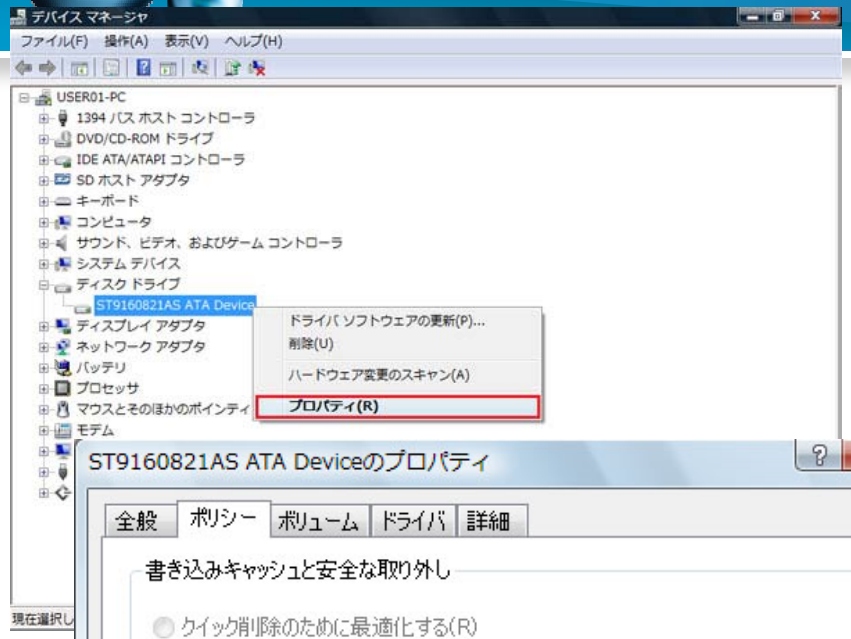


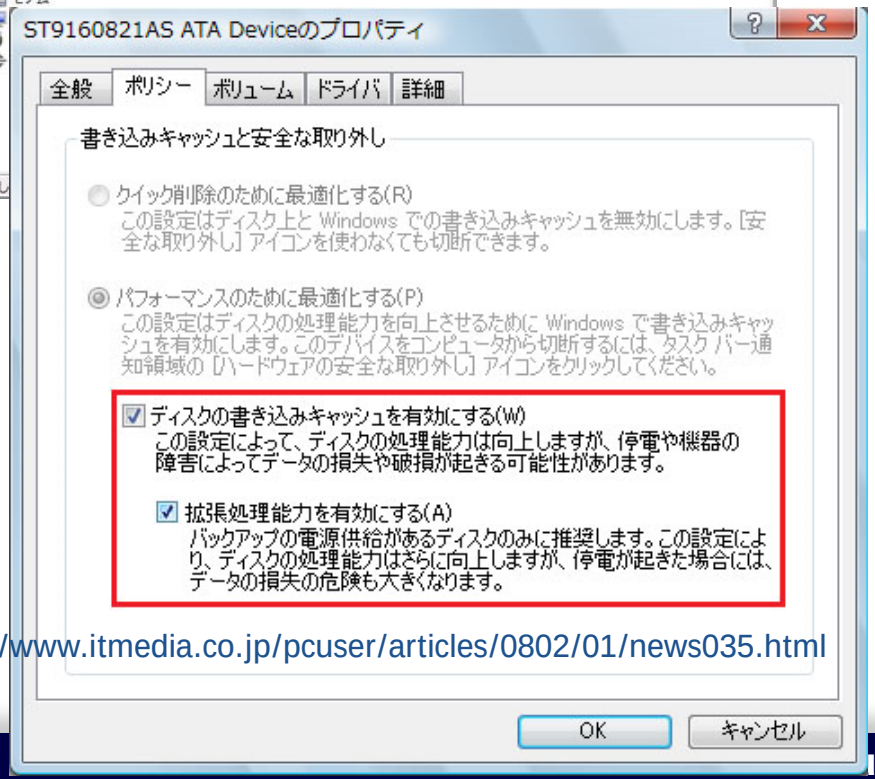
- 【ライトバック方式】
  CPU からの書き戻し時に、キャッシュメモリだけに書き込む方式。キャッシュメモリの内容を追い出す時に、主記憶に書き込まれる。
  読み書き両方で高速化が図れるためCPUを有効に使える。読み込み・書き込み両方で高速。



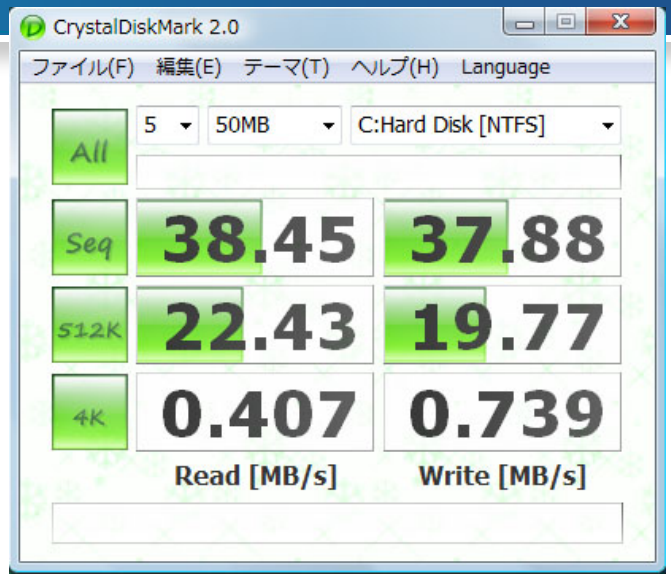
http://www.it-shikaku.jp/top30.php?hidari=03-02-03.php&migi=km03-02.php

ハードディスクの場合

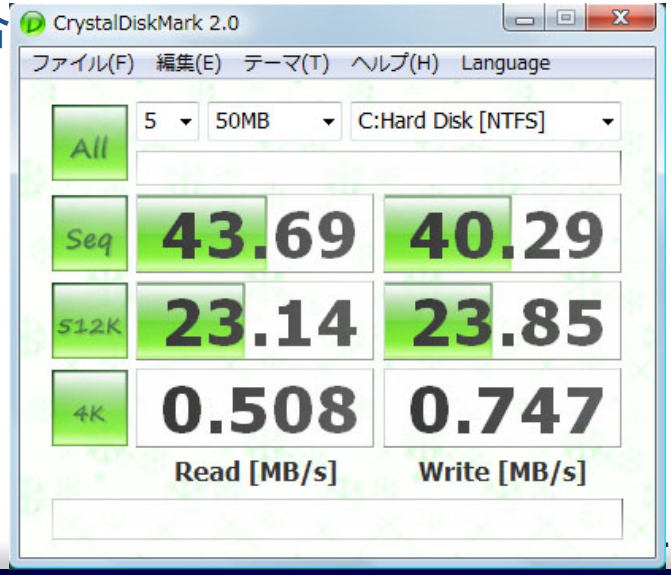




無効な場合



有効な場合



http://www.itmedia.co.jp/pcuser/articles/0802/01/news035.html