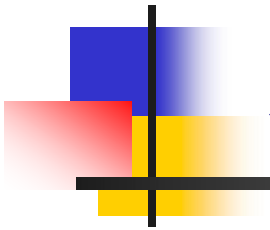
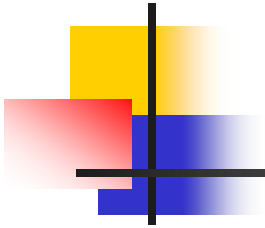


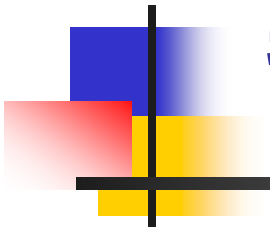
ext3ファイルシステムへの スナップショット機能の設計と実装

NTTコムウェア株式会社
オープンソースソフトウェア推進部
前野真輝
(E-Mail: maeno.masaki@nttcom.co.jp)



目次

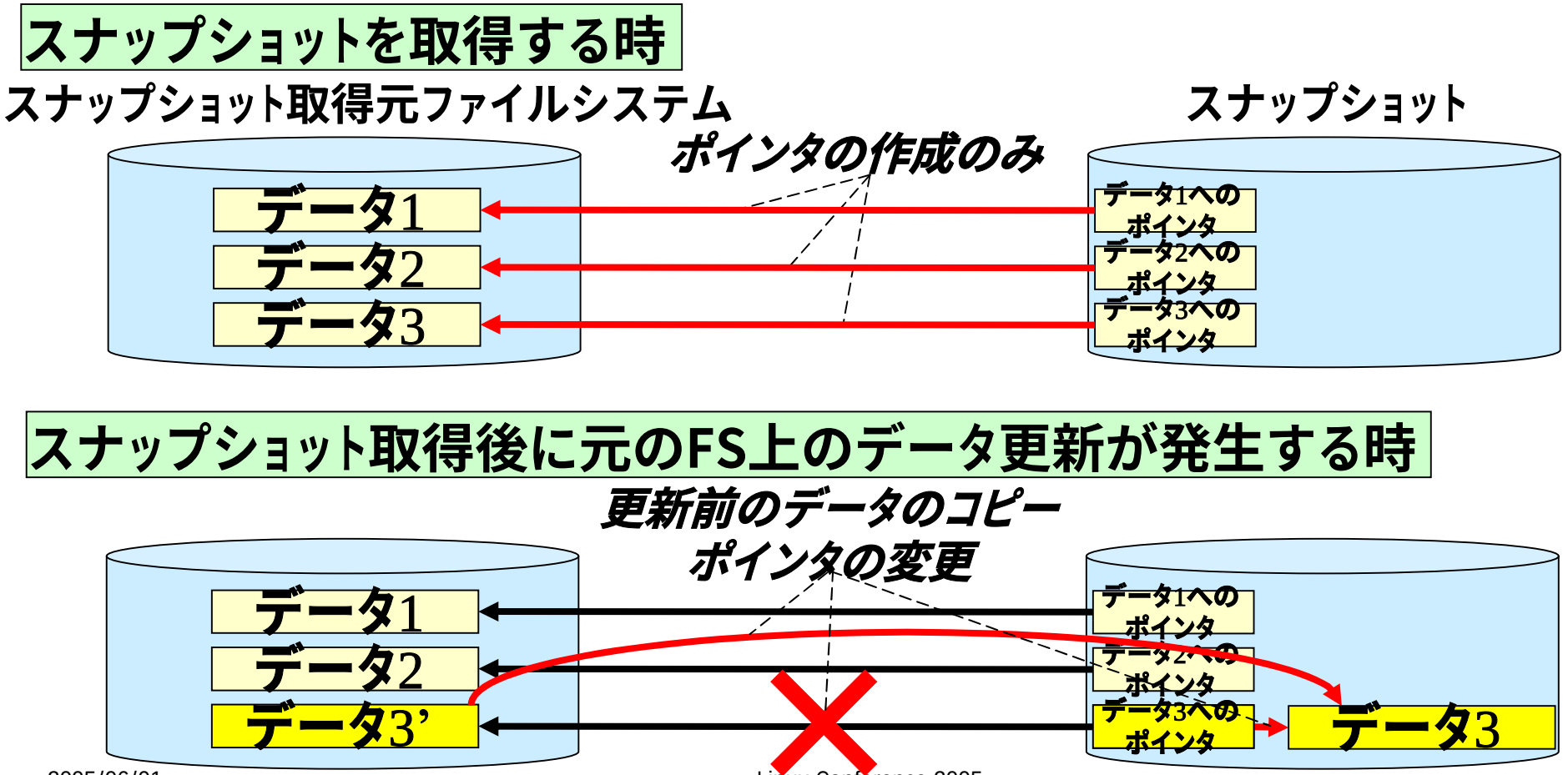
- 背景
  - スナップショットの概要及び実装例
  - ext3スナップショットの実現動機
- ext3スナップショットの実現方針
  - ffsスナップショットの概要
  - ext3ファイルシステムのジャーナル機能の概要
- ext3スナップショットの設計
- 現時点の実装とその評価
- 今後の課題



背景

スナップショットの概要

- スナップショットはある瞬間のイメージを保持
  - 利点: 短い時間でイメージを取得可能

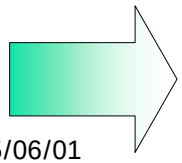


スナップショットの既存の実装例

名称	OS	レイヤ	特徴
LVM (=logical volume manager) スナップショット	Linux	ブロックデバイスレイヤ	■Device-mapperを利用
ffs (=BSD fast file system) スナップショット	FreeBSD NetBSD	FSレイヤ (更新型)	■ffsを拡張 ■soft updatesを利用
WAFL (=write anywhere file layout) スナップショット	ストレージシステム	FSレイヤ (追記型(LFS))	■NetApp社が特許を持つWAFLを拡張 ■スナップショットが同じinode番号を持つ
SnapView TimeFinder	ストレージシステム	FSレイヤ	■EMC社のSymmetrix等の製品に採用

ext3スナップショットの実現動機

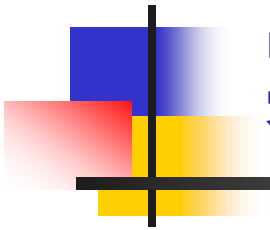
- 現在Linux上で有力なのはLVMスナップショット
  - ブロックデバイスレイヤでの実装
  - スナップショット専用領域を別に確保して差分情報を格納
  - chunk単位(=大きなサイズ)でスナップショット取得後の更新前データのコピー(=Copy-On-Write)して保護
- 利点
  - スナップショット取得時に存在したデータの更新が速い
  - スナップショットとして利用できる量を自由に決定可能
- 欠点
  - FSレベルでみると、更新されていないデータもコピーが行われるため、余分なコピーが発生し、余分にディスク容量を消費
  - 利用できる量を超えるとスナップショットは消滅



•FSレイヤで実装する事により欠点を解決できるのではないか
•現在Linuxで標準FSであるext3FSへの実装

ext3スナップショットの実現目標

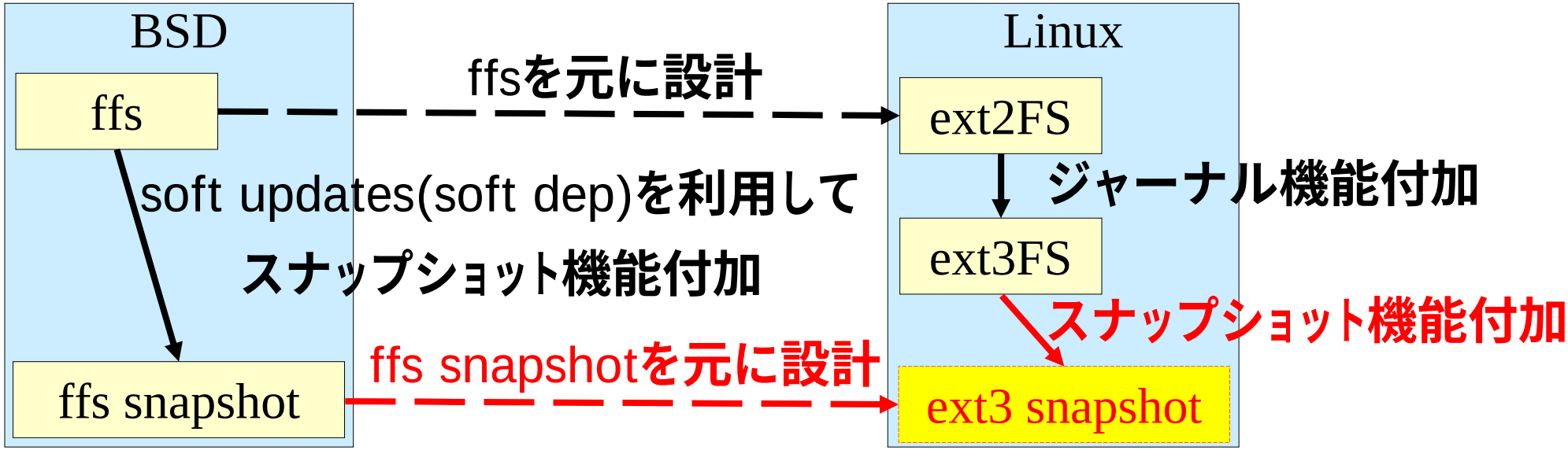
- スナップショットを通常通りext3FSとしてマウントして利用可能
  - ジャーナリングFSでスナップショット機能が実装可能である事を実証する
- スナップショット数にFSレイアウト上の制限を付けない
  - スーパブロックにスナップショットinode番号を埋め込む事は受け入れない
- 短時間での(スナップショットの)クラッシュリカバリ
  - ext3FSの特徴を失わない



ext3スナップショット
実現方針

ext3スナップショットの実現方針

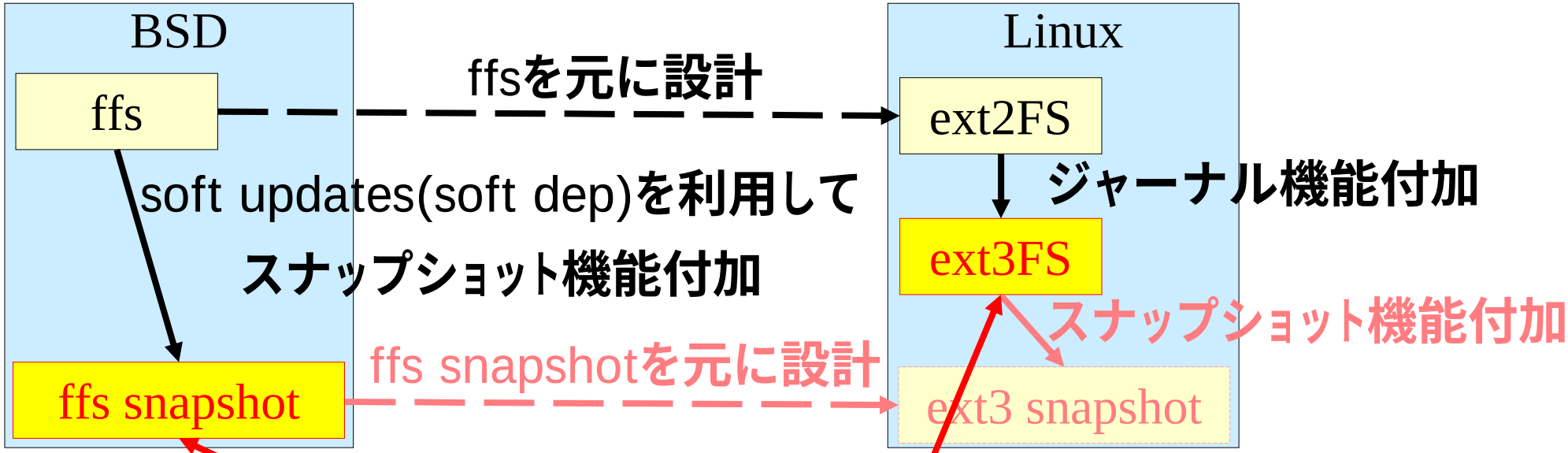
- ffsとext3の関係



- ext3スナップショットもffsスナップショットを基本にする
  - ただしext3FSのジャーナリングに関する制限を克服する実装が必要

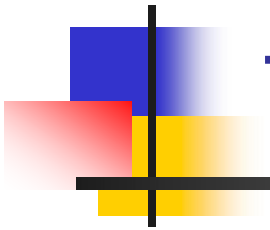
ext3スナップショットの実現方針

- ffsとext3の関係



- ext3スナップショットもffsスナップショットを基本にする
  - ただしext3FSのジャーナリングに関する制限を克服する実装が必要

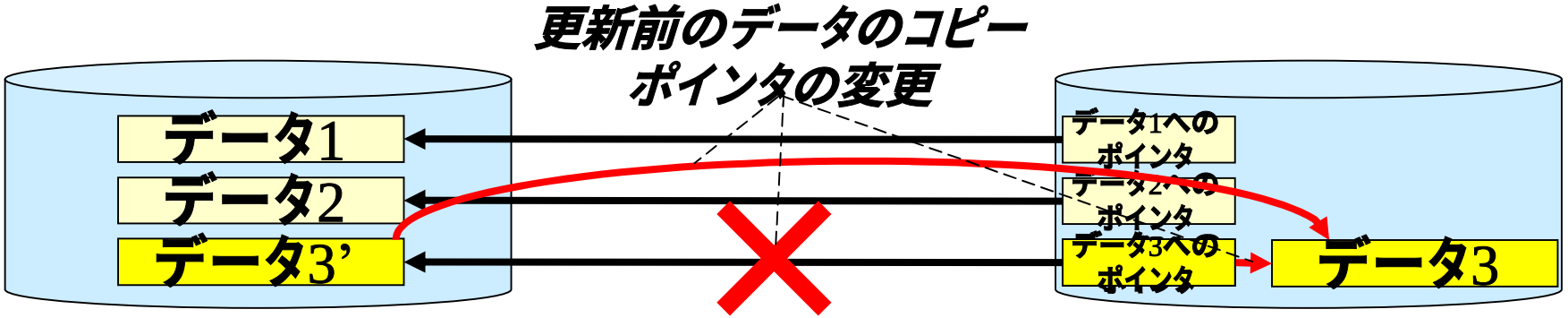
まずffsスナップショットとext3FSのジャーナル機能について説明する



ffsスナップショット

ffsスナップショットの概要

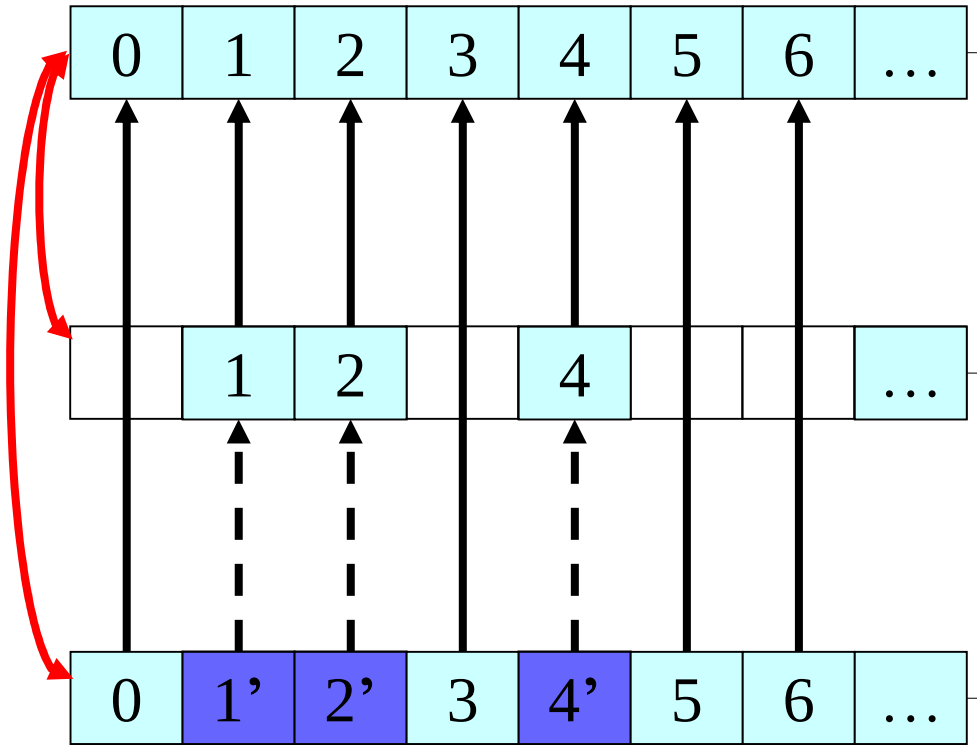
- あらゆる更新操作についてデータを書き換えた後にポインタを変更するsoft updatesの機構を利用
  - inode更新, 直接/間接ブロック確保, truncate down/up, ディレクトリ/エントリの追加/削除等のあらゆる更新操作のデータ依存関係を考慮
- ブロック書き出し時にCopy-On-Writeする事により、スナップショット取得時のデータを保護
  - 前述した動作と同じ



ffsスナップショットのデータ退避先とその利用

- Copy-On-Writeしたデータ退避先は取得元FS内の通常ファイル(=スナップショットファイル: 取得元FSと見かけ同容量のsparseファイル)へ行う

②: vnode仮想デバイスへスナップショットファイルと取得元FSを関連付ける



④: vnode仮想デバイスをマウントしスナップショットをFSとして利用

vnode仮想デバイス(=スナップショット用デバイス)

③: スナップショットファイルにブロックが存在すればスナップショットファイルから、存在しなければ取得元FSから読み出し

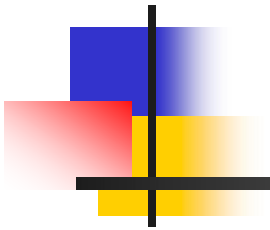
スナップショットファイル

①: スナップショット取得後から更新されるブロックを更新前にスナップショットファイルへコピー(=Copy-On-Write)

スナップショット取得元FS

= スナップショット取得後に更新されたデータブロック

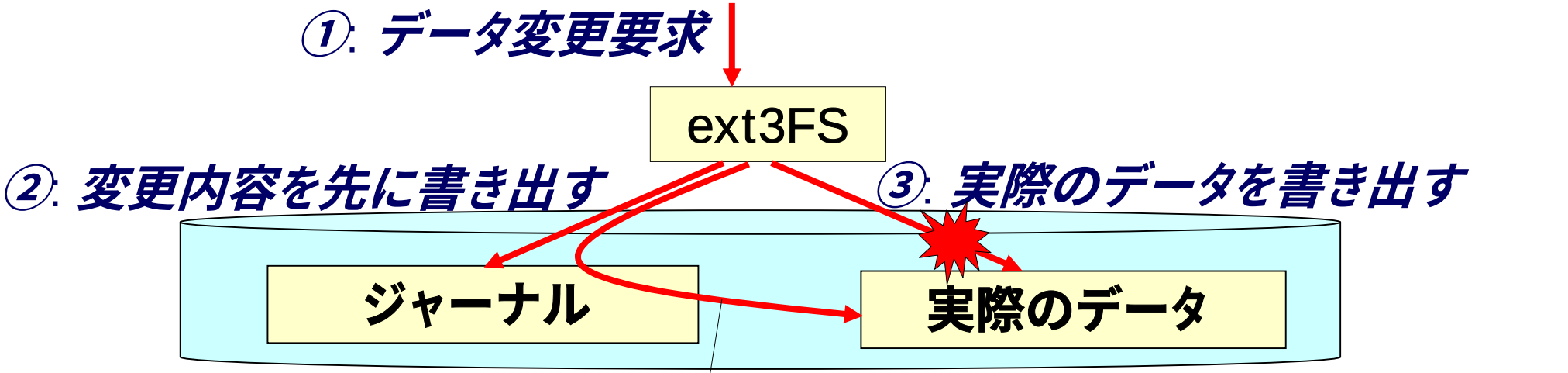
= スナップショット取得後に更新されていないデータブロック



ext3ファイルシステム (ジャーナル機能)

ext3FSのジャーナルの仕組みとその役割

- 複数のブロックにわたる不可分な変更内容をまずジャーナル領域へ書き出す事で、システムクラッシュしても、ジャーナルリプレイにより高速なリカバリを実現する
  - ジャーナルも通常ファイルと同様にinodeやデータブロックを持つ
  - ジャーナルは別デバイスに確保する事も可能



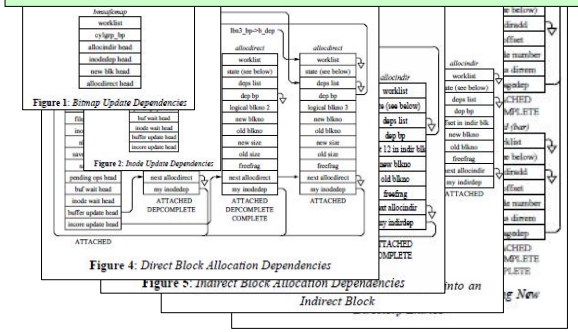
③：③中にクラッシュした場合ジャーナル内のデータを利用してリカバリする

ext3FSのジャーナルモード

- data=journal
  - データを含む全ての変更をジャーナルする
- data=ordered (ext3のデフォルトモード)
  - 通常ファイルのデータ以外(inodeや間接ブロックやビットマップブロックなどのメタデータ)に対しジャーナルする
  - ジャーナルを書き出す前に実際の通常ファイルのデータを書き出す
- data=writeback
  - 通常ファイルのデータ以外(メタデータ)に対しジャーナルする
  - ジャーナルと通常ファイルのデータの書き出し順序は問わない

ffs soft updatesとext3FSのディスク書き出し依存関係

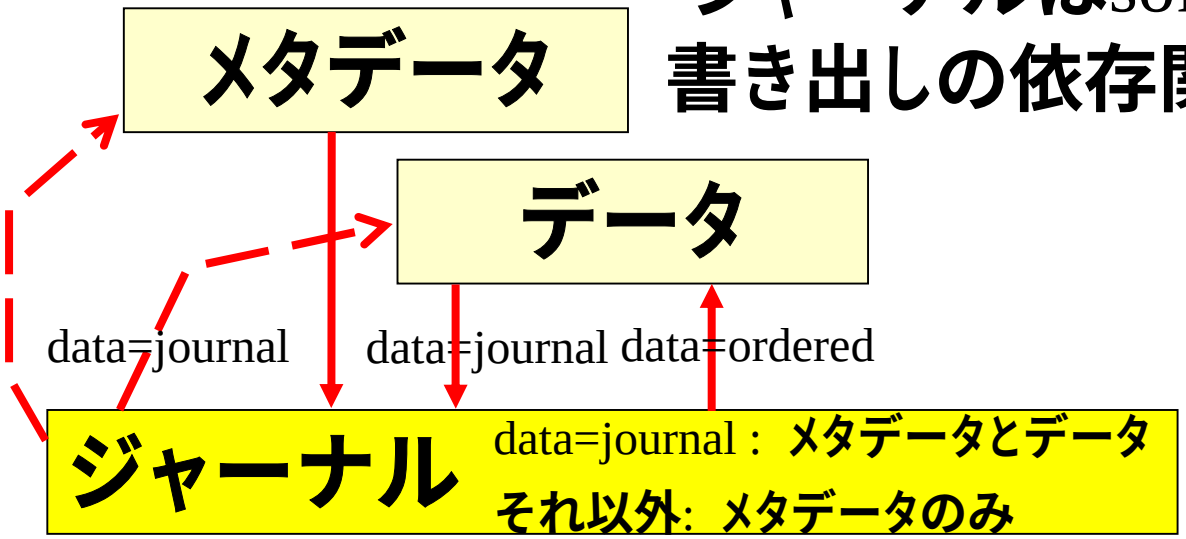
ffs soft updates



…

- soft updatesは全ての追加/更新/削除の各操作に対して細かい単位でディスク書き出しの依存関係を追跡する必要有

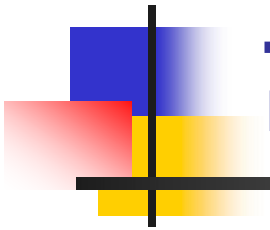
ext3ジャーナル



- ジャーナルはsoft updatesに比べてディスク書き出しの依存関係が大幅に単純化

A ⟶ B
ディスク書き込み依存関係
(Bを書き出し完了後, Aを書き出す)

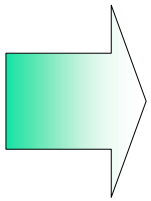
A - - → B
ジャーナル解放依存関係
(Bを書き出し完了後, Aを解放する)



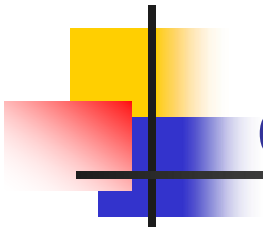
ext3スナップショット設計

ext3スナップショットの実現方針

- 基本的な考え方はffsスナップショットと同様
  - スナップショットの作成
    - スナップショットを取得する元のパーティション内へ見かけ上同容量のスナップショットファイルを作成
  - スナップショットの保護
    - ファイルシステム上のデータが更新される直前にスナップショット取得時のデータをスナップショットファイルへ退避 (=Copy-On-Write)
  - スナップショットの利用
    - スナップショットファイルを関連付けたスナップショット専用のブロックデバイスからスナップショットを利用



基本的な考え方はffsスナップショットと同様で既に動作を説明済みなので、処理の流れに沿うのではなく、各機能に焦点を当てて説明する

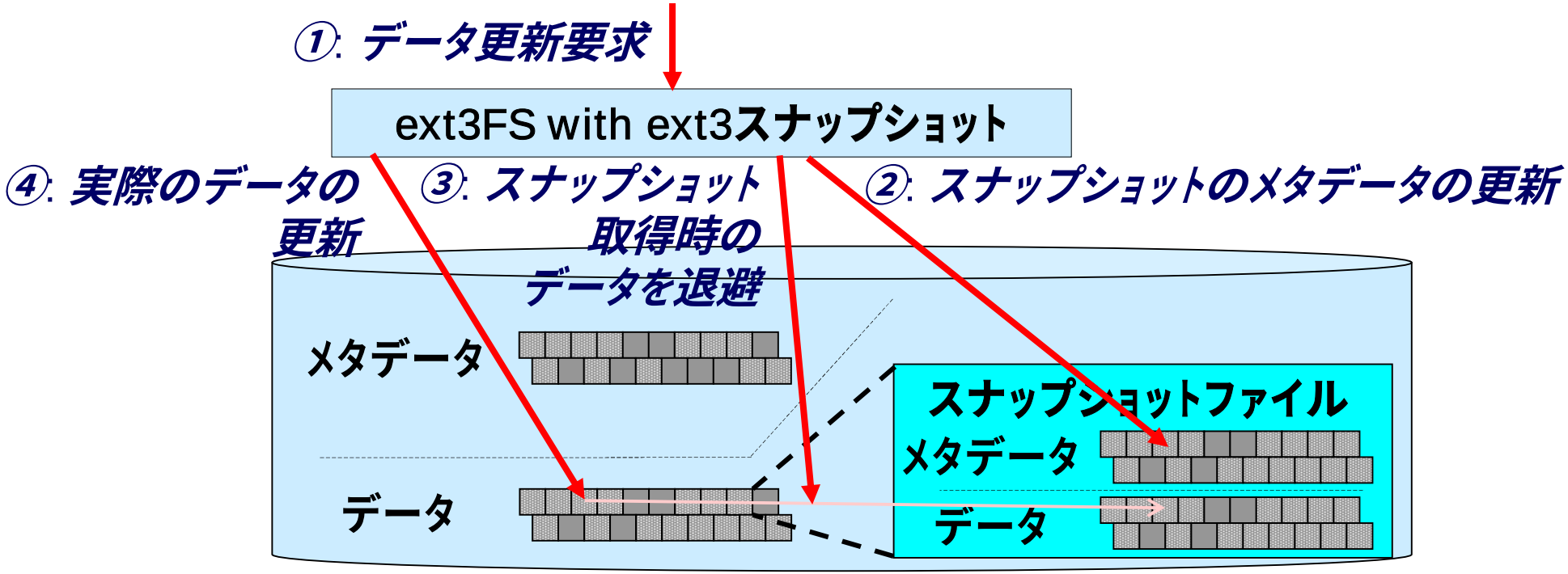


ext3スナップショットへ必要な機能

- ffsスナップショットと異なる方式で実装する機能
  - Copy-On-Write
  - beforeイメージジャーナル
  - スナップショットファイルのdelayed allocation
  - クラッシュリカバリ
- ffsスナップショットと同じ考え方で実装する機能
  - スナップショット初期化(スナップショットファイル作成)
  - ブロック解放抑止
  - スナップショット用デバイス

必要な機能: Copy-On-Write

- スナップショットのデータの保護
  - 実際のデータが更新される前にスナップショット取得時のデータをスナップショットファイルへコピーする (=Copy-On-Write)



必要な機能: Copy-On-Write

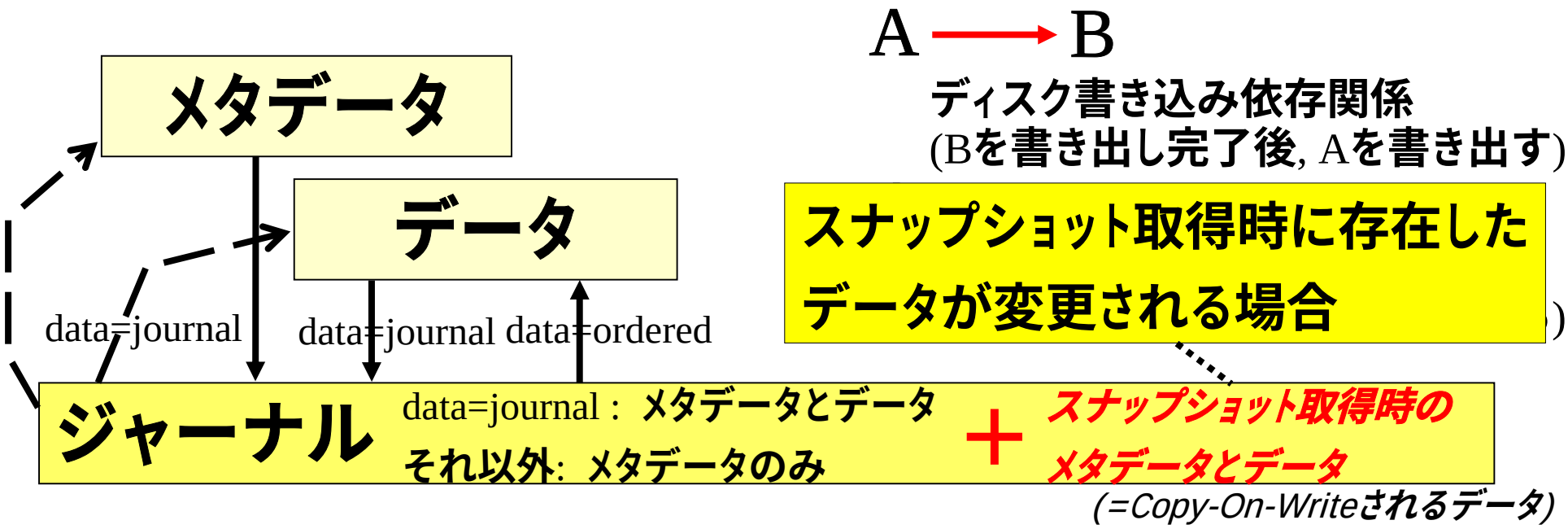
- 問題点
  - ffsスナップショットではディスクブロックの書き出し前にCopy-On-Writeしているが、ext3FSでは既にジャーナル領域に書き出されてしまった後である
    - その間にクラッシュするとスナップショットデータはジャーナルリプレイにより失われる
- 解決策
  - ディスクブロック書き出し時ではなく、データ更新時にCopy-On-Writeする

必要な機能: Copy-On-Write

- Copy-On-Writeのタイミング
  - メタデータ更新の直前
    - ジャーナルの変更前処理
    - inode bitmapによるinodeブロック内クリア処理
  - データ更新の直前
    - write()処理
    - mmap()処理
    - direct_IO()処理
    - ブロック境界でないtruncate down処理

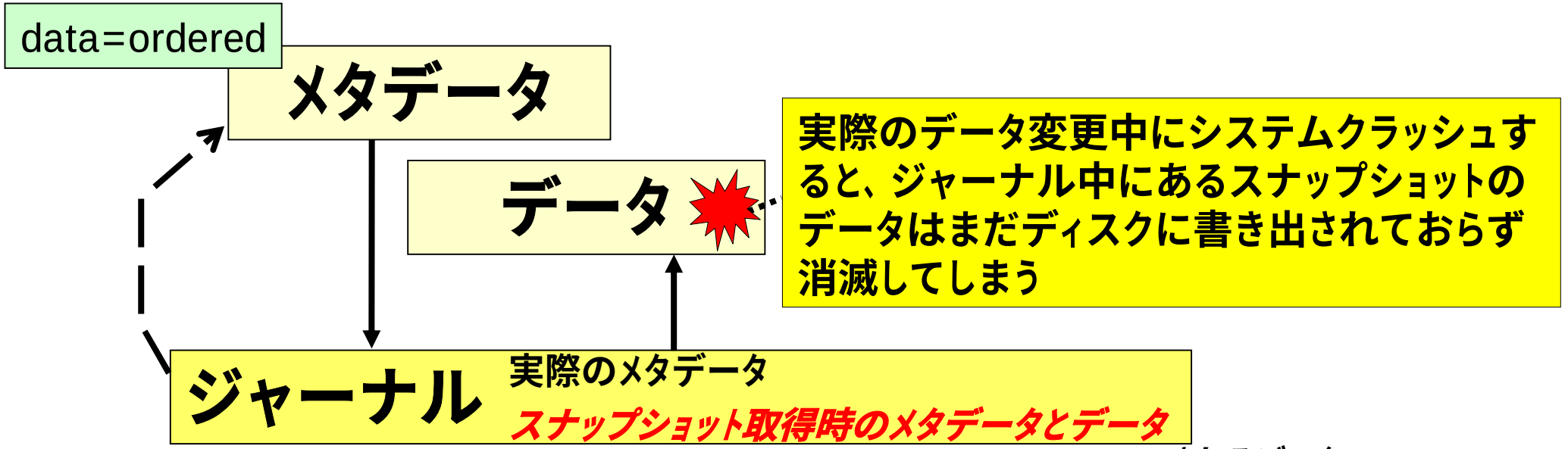
必要な機能: beforeイメージジャーナル

- クラッシュ時のスナップショットのデータの保護
  - スナップショット取得時のメタデータとデータをジャーナルへ書き出す(=ジャーナル量増加)
    - ただし、ジャーナルモードに依存せずにそれを行わなければならない



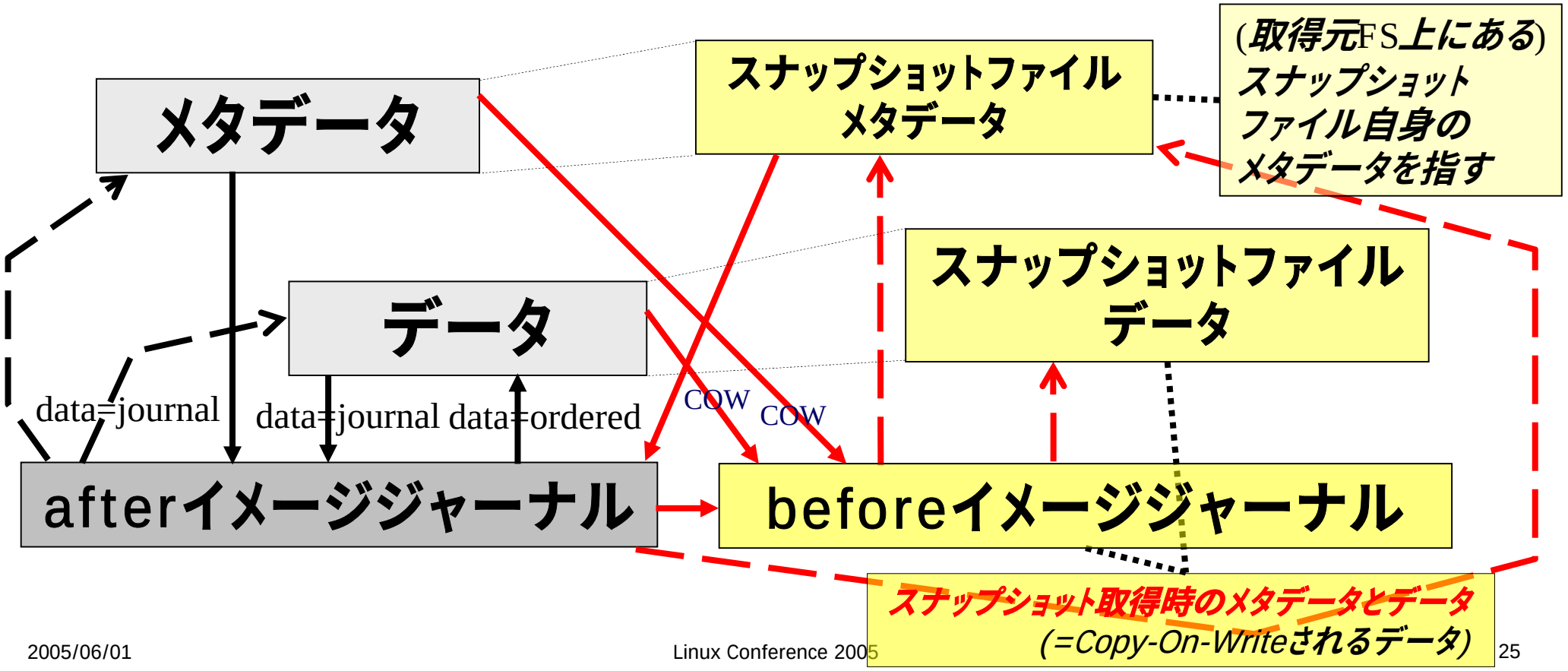
必要な機能: beforeイメージジャーナル

- 問題点
  - data=orderedではジャーナルよりもデータへの書き出しを先に行わねばならず、既存のジャーナルへスナップショットのデータを追加する方針には問題がある
    - data=journalでは問題なく、data=writebackでは新たな制約を設ければ良い



必要な機能: beforeイメージジャーナル

- 解決策
  - 従来のジャーナルとは別のスナップショット用のジャーナル(=beforeイメージジャーナル)を設ける
    - 従来のジャーナルはafterイメージジャーナルと呼ぶ



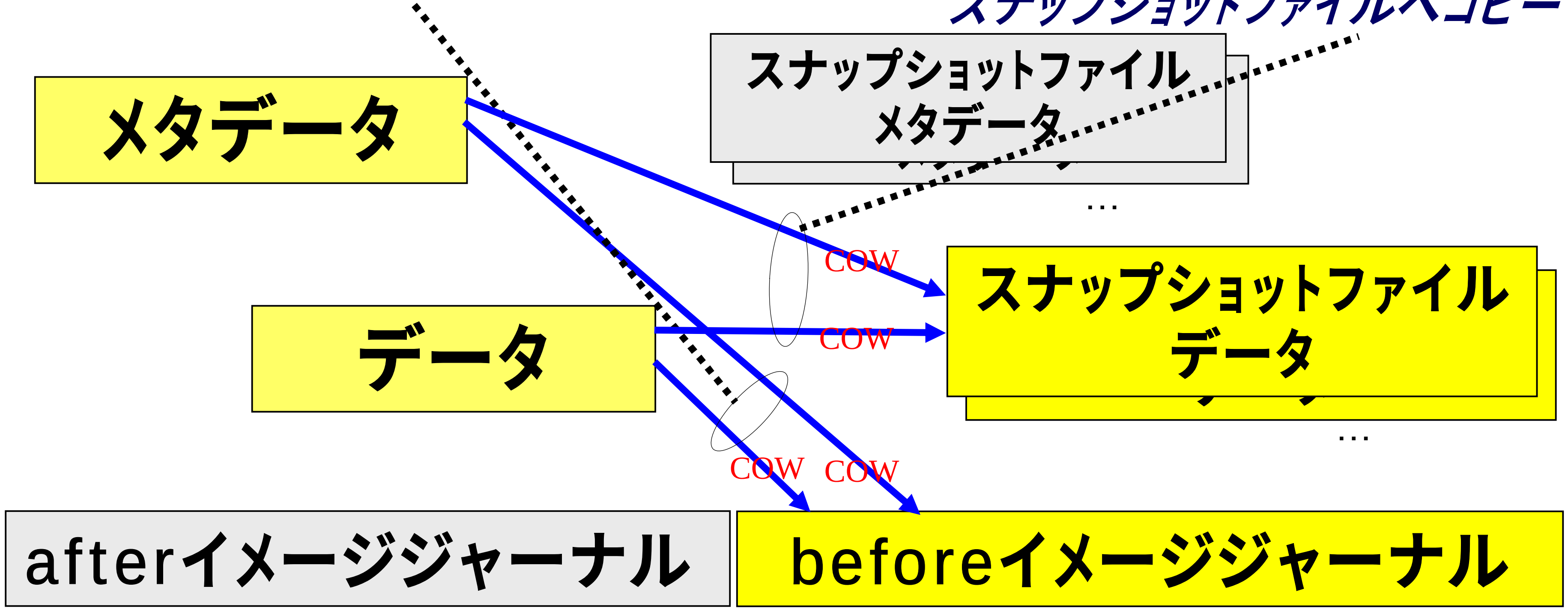
必要な機能:
Copy-On-Writeとbeforeイメージジャーナル

- **Copy-On-Write処理によるスナップショット取得時のデータの流れ**

①: afterイメージジャーナル(data=orderedの場合はデータ)を書き出す前にスナップショット取得時の情報をコピー

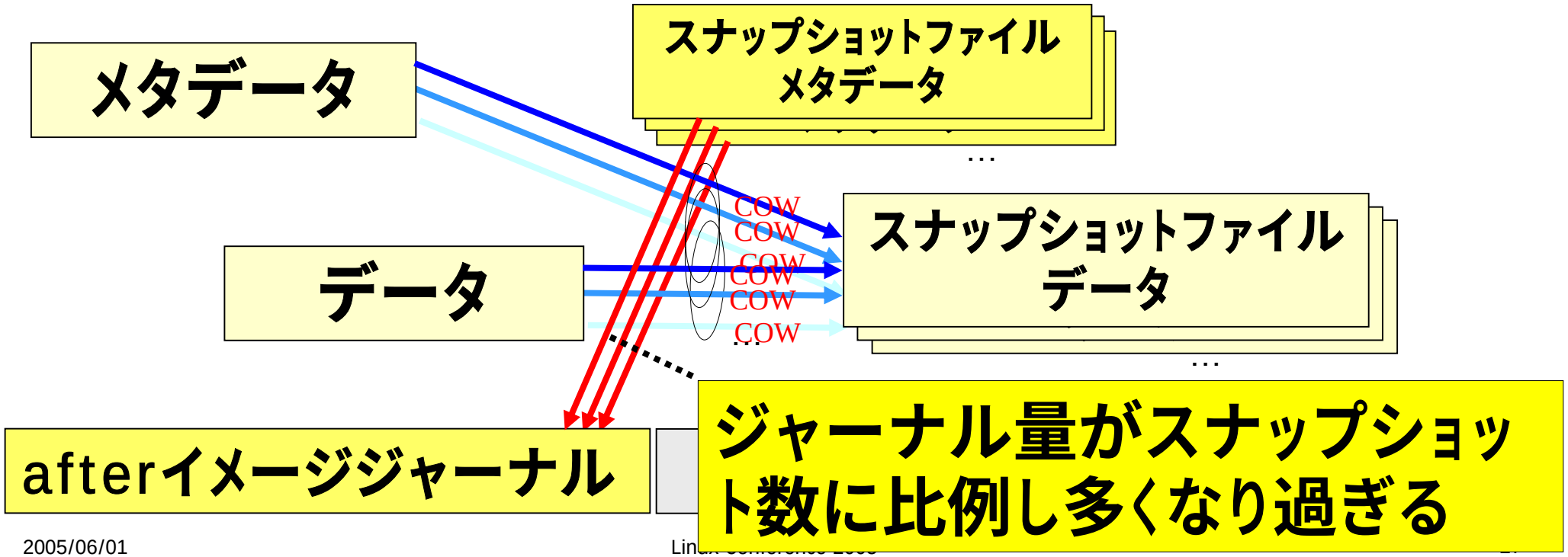
①の前後(スナップショット取得時の情報が更新されるのを検知した後):

スナップショット取得時の情報を各々のスナップショットファイルへコピー



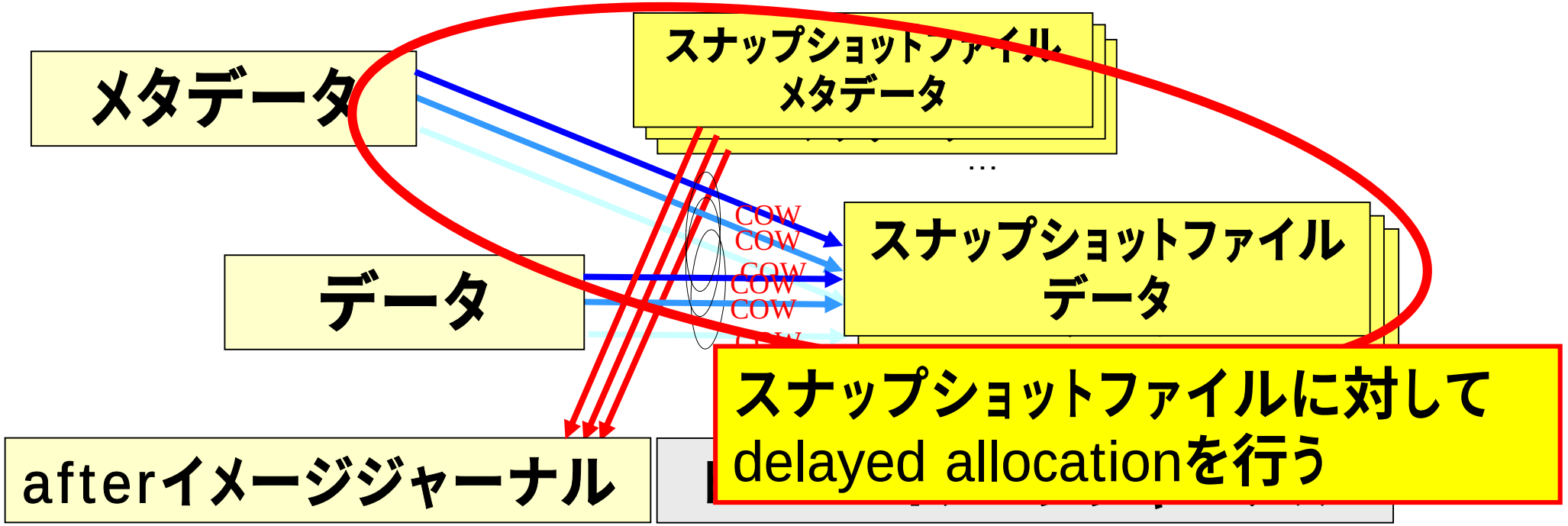
必要な機能: スナップショットファイルのdelayed allocation

- 複数スナップショット取得時の問題点
  - スナップショット数が増えれば増えるほど、システムコールを完了するための変更ブロック数が増加し、afterイメージジャーナル量が多くなり過ぎ、処理が破綻



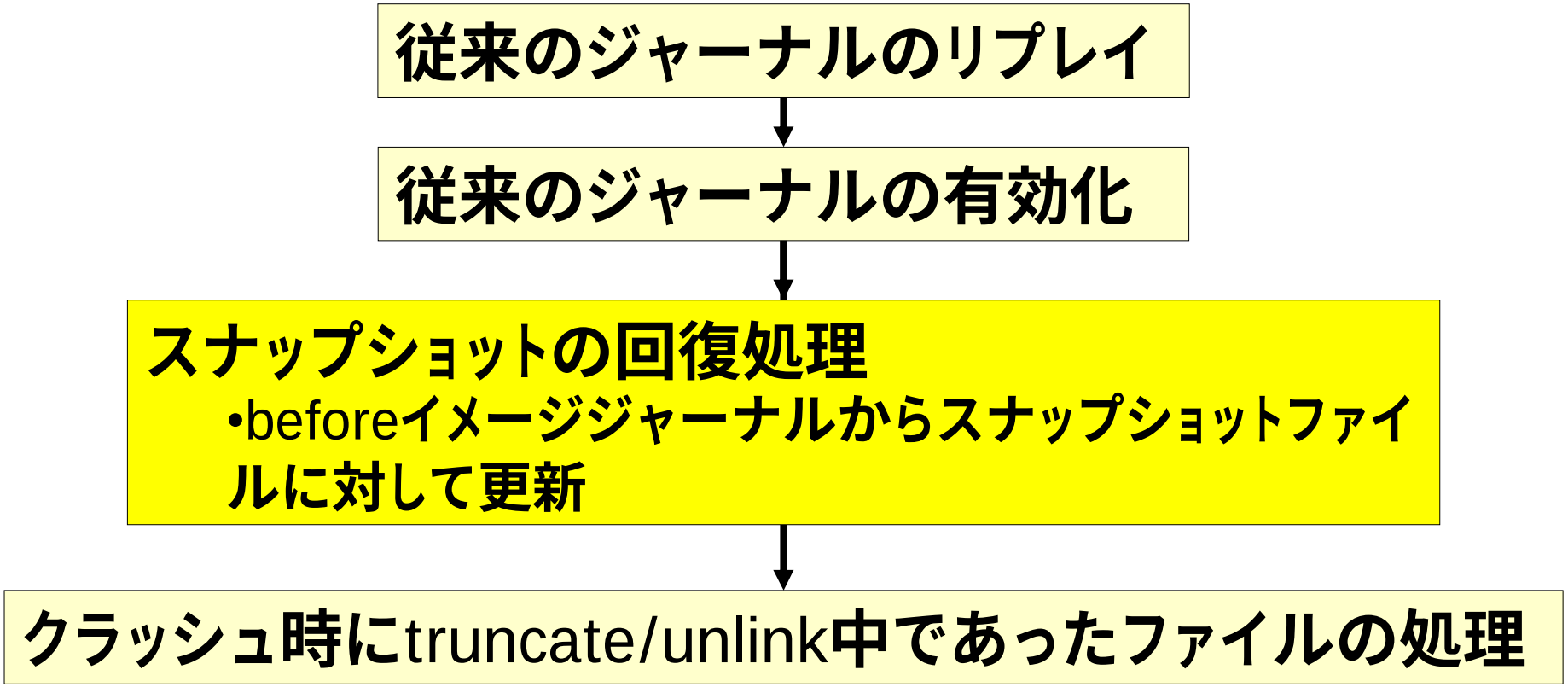
必要な機能: スナップショットファイルのdelayed allocation

- 解決策
  - ブロック割り当て処理をwrite () 時に行うのではなく、ページ書き出し時に行うようにする
    - Copy-On-Write時に(afterイメージ)ジャーナルを必要としなくなる



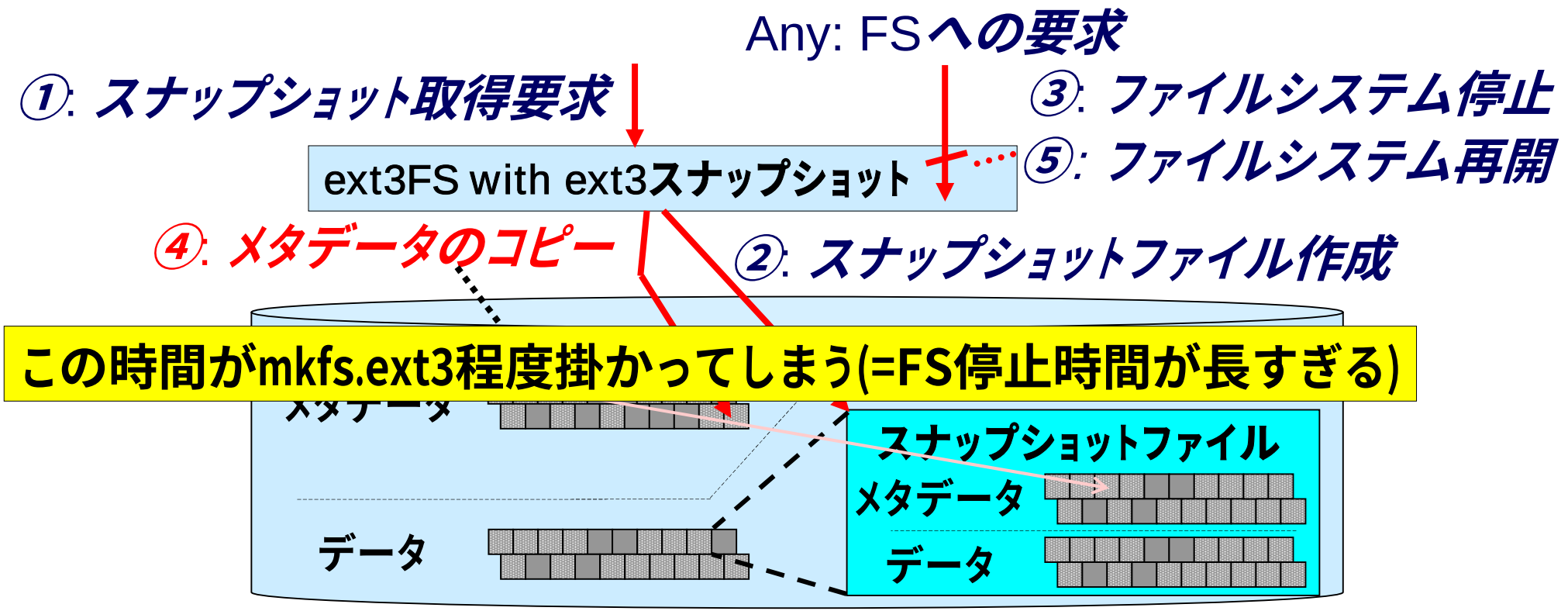
必要な機能: クラッシュリカバリ

- スナップショットの回復処理の追加
  - クラッシュリカバリの流れ



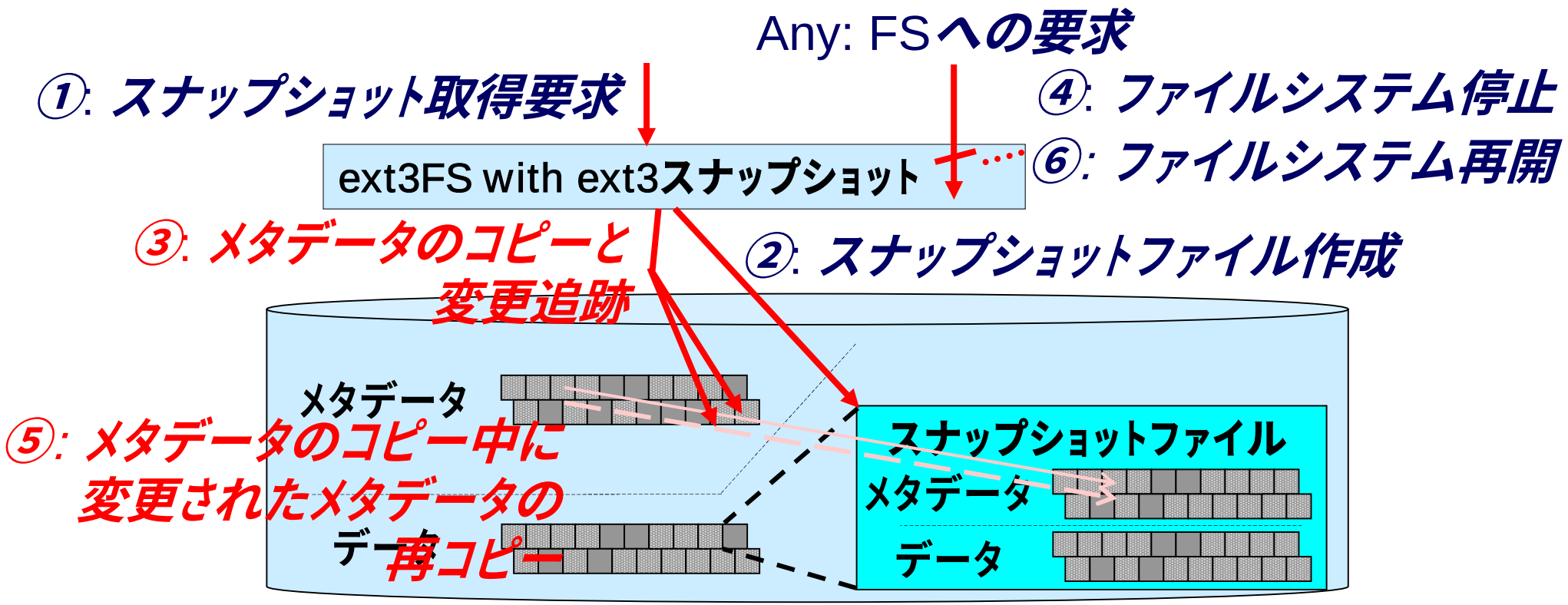
必要な機能: スナップショット初期化

- スナップショットファイル作成
  - 見かけ上FSと同容量のファイルを作成
  - FSを停止させてメタデータをコピー



必要な機能: スナップショット初期化

- 解決策
  - FSの停止時間を短くするために前もってメタデータをコピーし、コピー中に変化したメタデータだけをFSを停止させて再度コピー

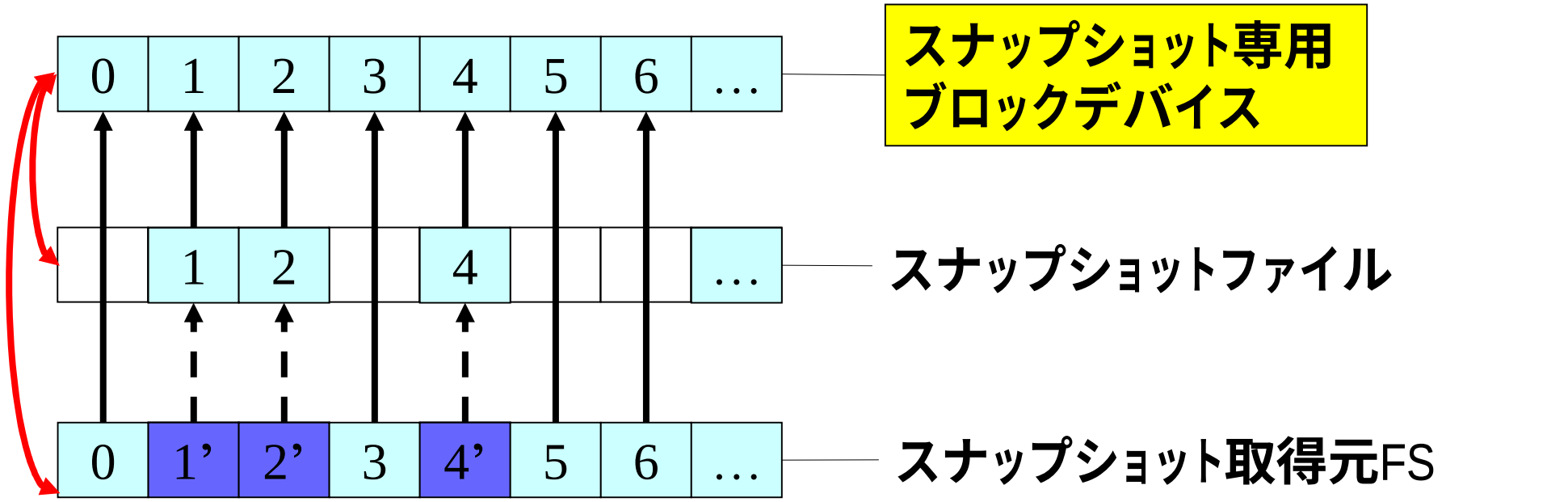


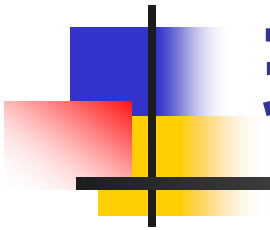
必要な機能: ブロック解放抑止

- ブロック解放時のスナップショットのデータの保護
  - スナップショット取得後は、スナップショットが該当ブロックを参照している可能性があり、もし参照している場合はブロック解放しない
- ブロック解放抑止のタイミング
  - ブロック解放時
    - free_blocks()処理
  - Copy-On-Write時と異なり、解放処理はジャーナルされない(必要がない)ために解放抑止処理はブロック解放時で問題無い

必要な機能: スナップショット用デバイス

- スナップショットの利用
  - スナップ取得元デバイスのデータとスナップショットファイルを選択的に読み込む専用ブロックデバイスから利用





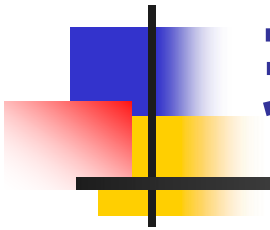
ext3スナップショット
現時点の実装状況

現時点での実装

- スナップショットの基本動作は実装完了
  - FS毎に1つのスナップショットが取得でき、スナップショット用デバイスを介してスナップショットをマウントし読み出せ、スナップショットを解放できる
  - 制約条件
    - アンマウントやシステムクラッシュによりスナップショットは消滅する
    - ディスクフル時はスナップショット取得元FSがReadOnlyへ移行する
      - ディスクフルに関与しない部分への読み出しはスナップショット・取得FS共に可能

現時点で実装した機能

- 実装済が青色，未実装が赤色
  - ffsスナップショットと異なる方式で実装する機能
    - Copy-On-Write
    - beforeイメージジャーナル
      - 既存のジャーナル量を増やしてdata=journalでスナップショットファイルをジャーナルしたアドホックな実装
    - スナップショットファイルのdelayed allocation
    - クラッシュリカバリ
  - ffsスナップショットと同じ考え方で実装する機能
    - スナップショット初期化(スナップショットファイル作成)
    - ブロック解放抑止
    - スナップショット用デバイス



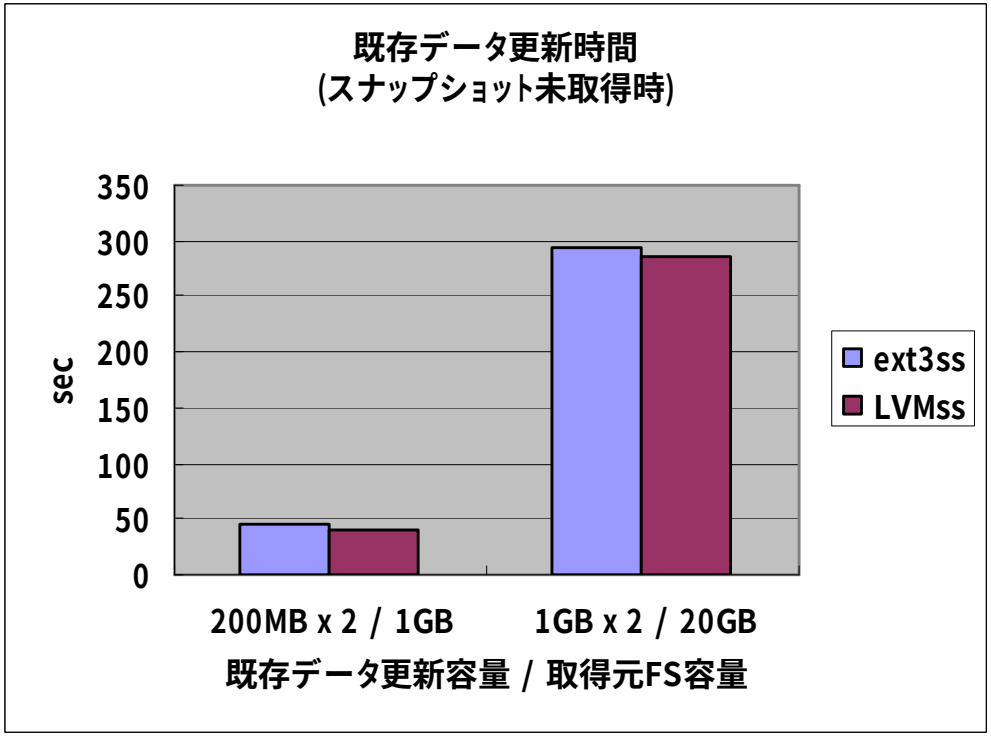
ext3スナップショット
現時点の実装の評価

評価観点と測定内容

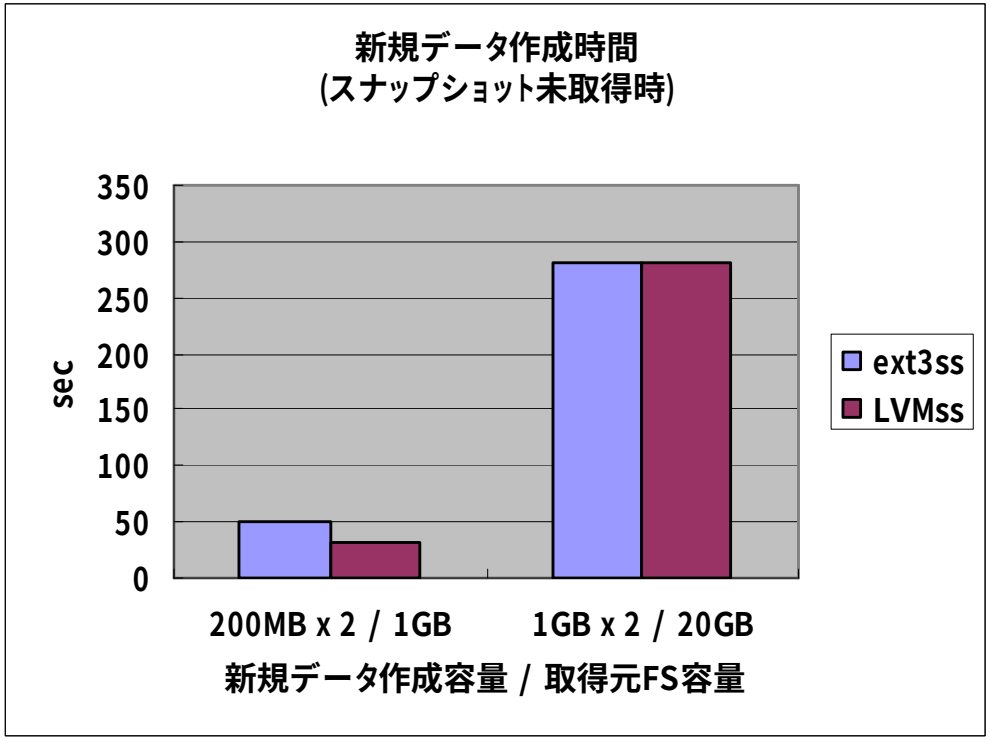
- 評価観点
  - LVMスナップショット(ext3 on LVM)との比較
    - スナップショット未取得時のデータ更新・新規作成の速さ
    - スナップショットを取得する速さ
    - スナップショット取得後のデータ更新の速さ
    - スナップショット取得後のデータ新規作成の速さ
      - ext3スナップショットではCopy-On-Write処理が発生しないが、LVMスナップショットではCopy-On-Write処理が発生する
- 測定内容(3回測定した各平均時間)
  - スナップショット未取得時のデータ更新・新規作成の時間
  - スナップショット取得の時間とその際のFS停止の時間
  - スナップショット取得後の既存ファイルのデータ更新の時間
  - スナップショット取得後の新規ファイルのデータ作成の時間

スナップショット未取得時の既存データ更新と新規データ作成の時間

既存データ更新時間



新規データ作成時間

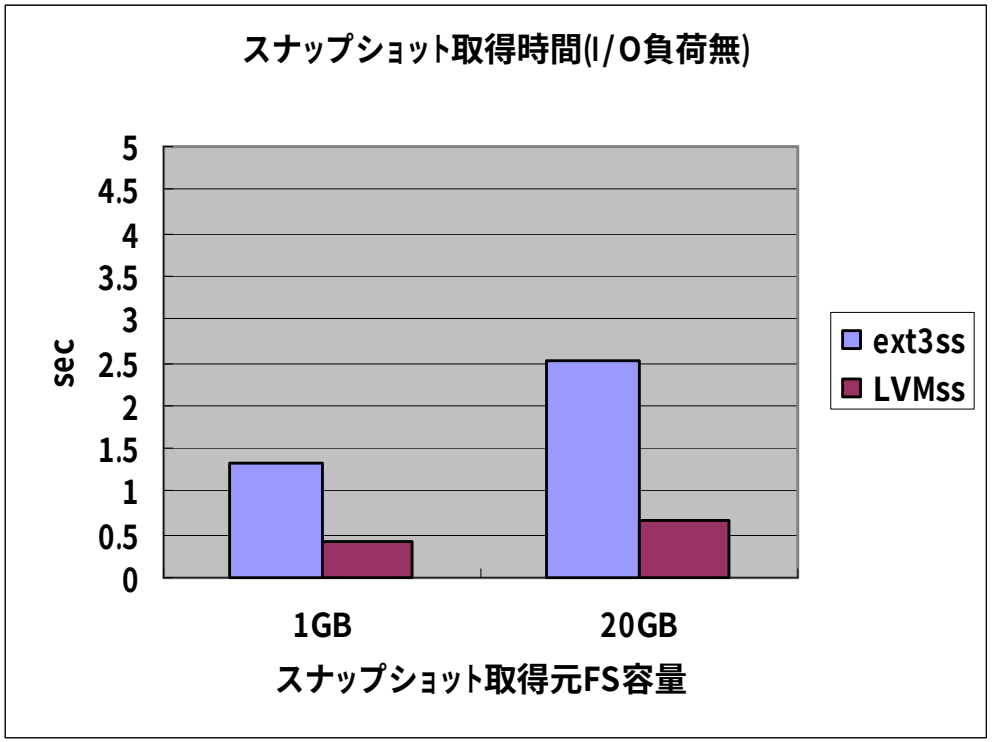


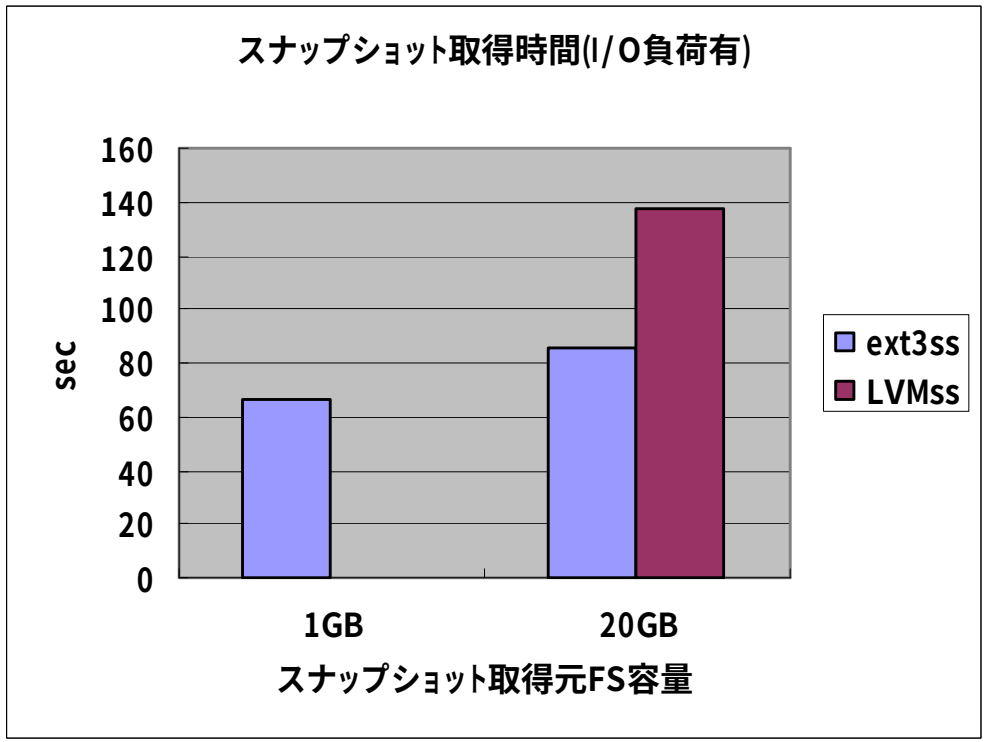
- データの更新・新規作成ともに若干遅い
  - Copy-On-Writeのために予約するジャーナル量が増え、ジャーナルのディスク書き出し動作が増えているため

スナップショット取得の時間

スナップショット取得時間

ddコマンドで新規データ作成しながら
スナップショットを取得





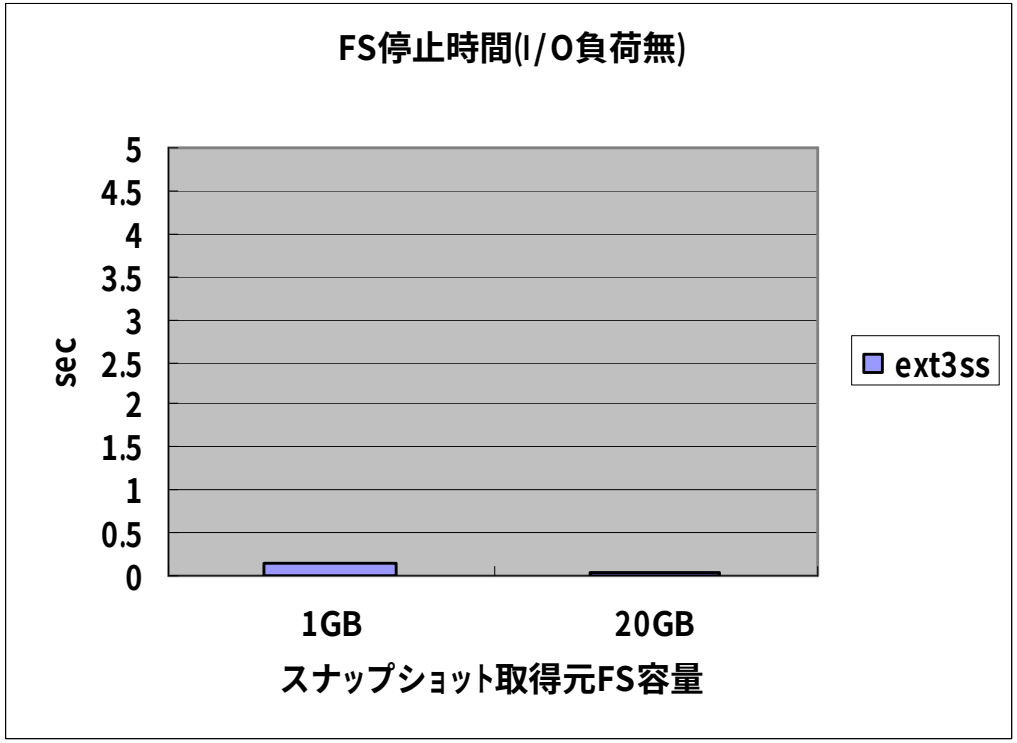
- ext3の方が3〜4倍程度遅い
  - メタデータのコピー処理で必要なブロックを前もって確保するのではなく、その都度ブロックを確保しているため

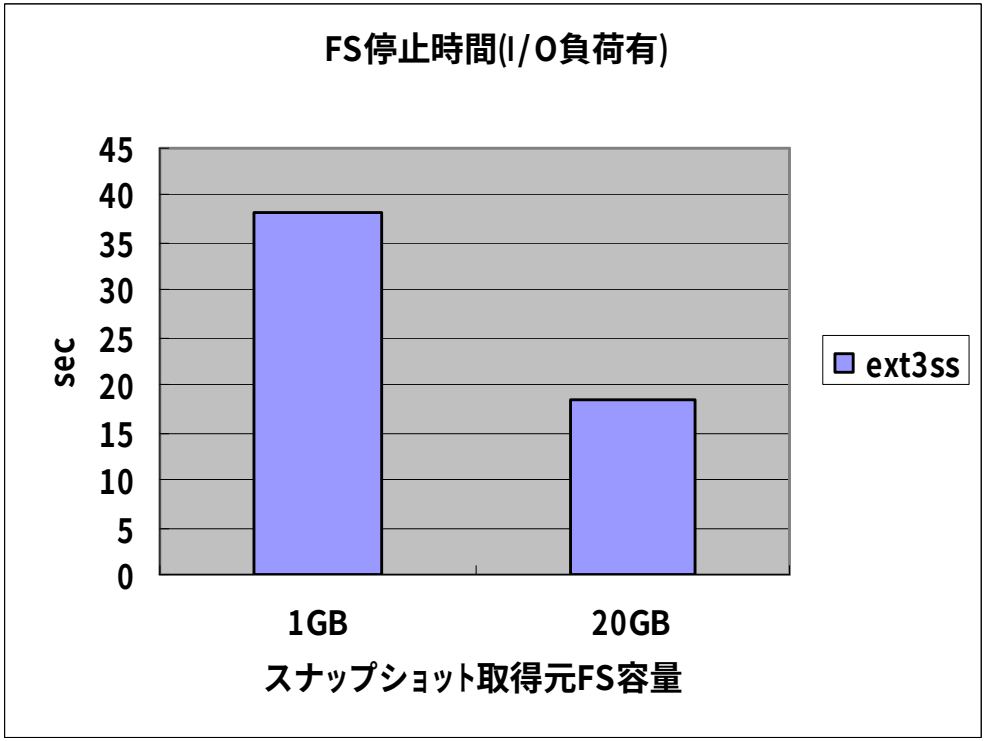
- 両者とも数十秒以上掛かっている
  - LVMはI/O高負荷時スナップショット取得が遅いためかddコマンドによるデータの作成が速いためか

スナップショット取得時のFS停止の時間

スナップショット取得時FS停止時間

ddコマンドで新規データ作成しながらスナップショットを取得



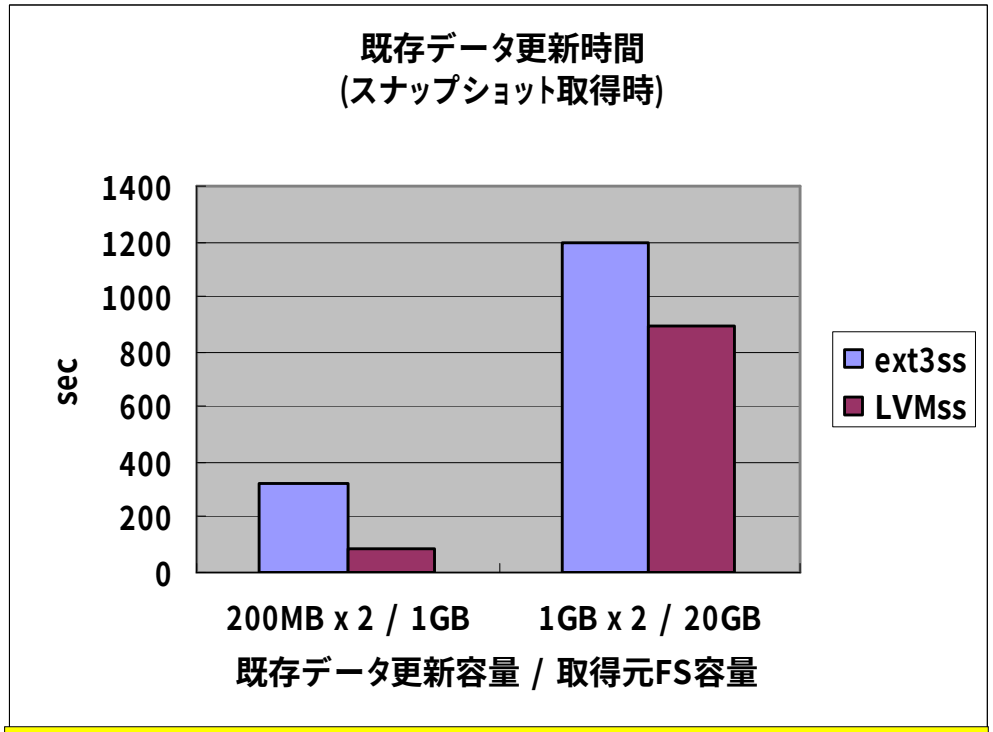


- I/O負荷が無い時は問題無い

- I/O負荷が有る時は数十秒もFS停止していて問題有り
  - もしくは やむを得ない(かも)

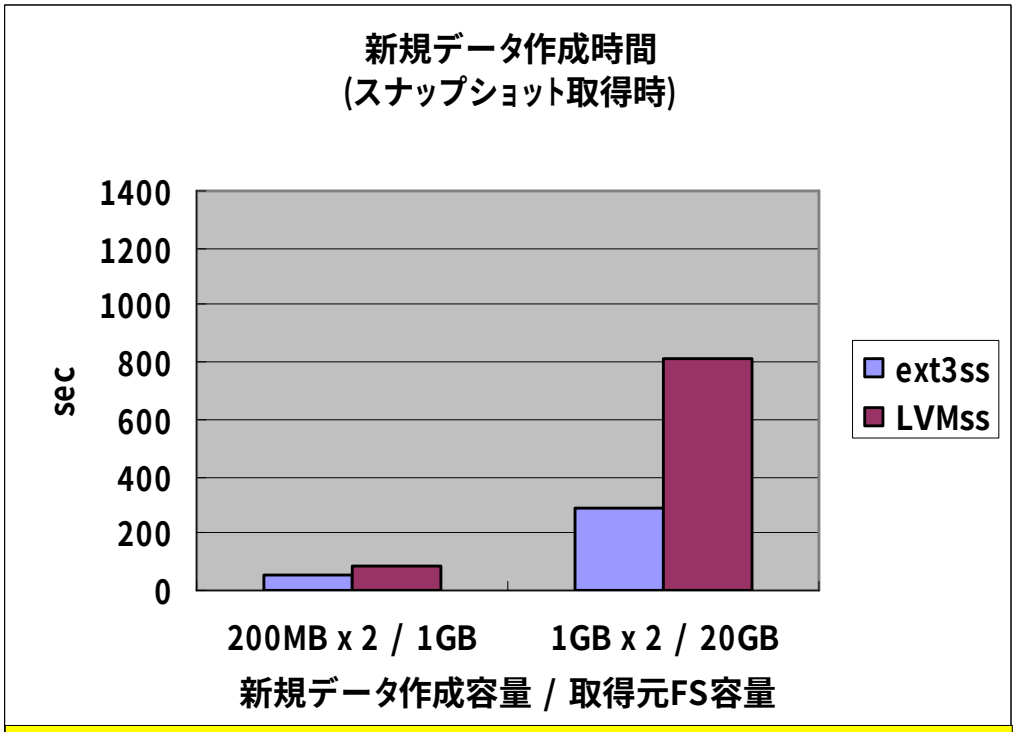
スナップショット取得後の既存データ更新と新規データ作成の時間

既存データ更新時間



- ext3の方が1.5〜4倍程度遅い
  - LVMの方がCopy-On-Write単位が大きいため、効率が良い
  - 少し遅すぎる(かも)

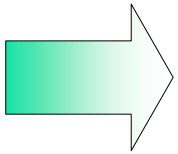
新規データ作成時間



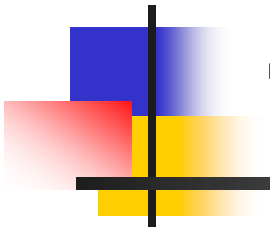
- ext3の方が1.5倍〜3倍程度速い
  - LVMの方がCopy-On-Write単位が大きいため、余分なCopy-On-Writeが発生している

現時点の実装の評価のまとめ(LVMとの比較)

- 利点
  - スナップショット取得後の新規データ作成が速い
  - 更新されないデータはCopy-On-Writeされないので、余分なディスク容量の消費が無い
- 欠点(=今後の課題)
  - スナップショット取得後の既存データ更新が遅い
  - スナップショット取得時間やI/O高負荷下のFS停止が長い



•課題はあるもののFSレイヤで実装する事により、LVMスナップショットと異なる性能特性を示し、ext3スナップショットの存在意義が認められる
  •ただし実装途中なので数値は参考程度に



ext3スナップショット デモ

時間があれば...

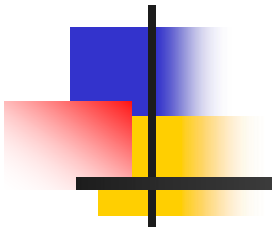
ext3スナップショットの利用の流れ

- 利用例
  - スナップショットの取得から利用まで

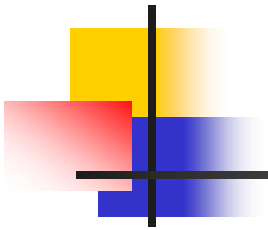
```
# e3snapconfig  –c  /mnt/src/snapshot.img                  … ①
# ssdevconfig  –c  /mnt/src/snapshot.img  /dev/ssdev/0     … ②
# mount  /dev/ssdev/0  /mnt/ss                             … ③
# cd  /mnt/ss                                              … ④
      …
```

  - スナップショットの利用終了から解放まで

```
      …
# umount  /mnt/ss                                         … ⑤
# ssdevconfig  –d  /dev/ssdev/0                            … ⑥
# e3snapconfig  –d  /mnt/src/snapshot.img                  … ⑦
```

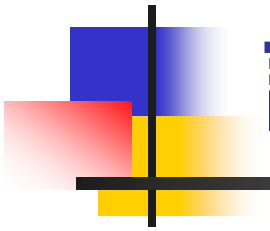


まとめ



まとめ

- ext3FSへのスナップショット機能追加の設計と実装について述べ、その存在意義を示した
  - 現時点のソースコード及び利用手順を以下で公開している
    - http://sourceforge.net/projects/ext3snapshot/
- 今後の課題
  - スナップショット取得の性能向上
  - 未実装部分の実装
    - スナップショットの永続化
      - beforeイメージジャーナルの実装
      - クラッシュリカバリの実装
    - 複数のスナップショットの取得
      - スナップショットファイルのdelayed allocationの実装



ご静聴頂きまして
誠にありがとうございました

前野真輝＇ 松尾隆利＇ 山幡為佐久＂

所属: ＇= NTTコムウェア株式会社, ＂= VA Linux Systems Japan株式会社