拡張可能OSのfail-safe機構

光来 健一*　　千葉 滋**　　益田 隆司*

* 東京大学大学院 理学系研究科 情報科学専攻
** 筑波大学 電子・情報工学系

fail-safe機構の必要性

- 拡張可能OSでは拡張モジュールを追加して機能の拡張ができる
  例：CD-ROMファイルシステム

- このようなOSにはfail-safe機構が必要である
  - 拡張モジュールにはエラーがあるかもしれない
  - そのエラーはOSの他の部分には影響を及ぼすべきではない

fail-safe機構が役に立つ例

- 開発者はファイルシステム・モジュールのデバッグがしやすくなる
  - エラーが正確に検出できる
  - エラーが検出されたモジュールをOSから安全に切り離せる
- ユーザは不安定なファイルシステム・モジュールを安心して動かせる
  - エラーによってOS全体が落ちることはない

fail-safe機構の問題

- fail-safe機構を強力にするとオーバヘッドが大きくなる
  - Machの拡張モジュール (ユーザプロセス)
- 性能を重視するとfail-safeをあきらめなければならない
  - NetBSDのloadable kernel module

fail-safe機構に対する考察

- 常に完全なfail-safe機構が必要なわけではない
  - 悪意のある拡張モジュールは組み込まれない
    - 管理者だけがOSを拡張できる
    - ソフトウェアのバグに対してのみfail-safe機構を考えればよい
  - ソフトウェアのバグは次第に減る

    例：開発時には多くのバグがあるが、リリース時にはほとんどなくなる

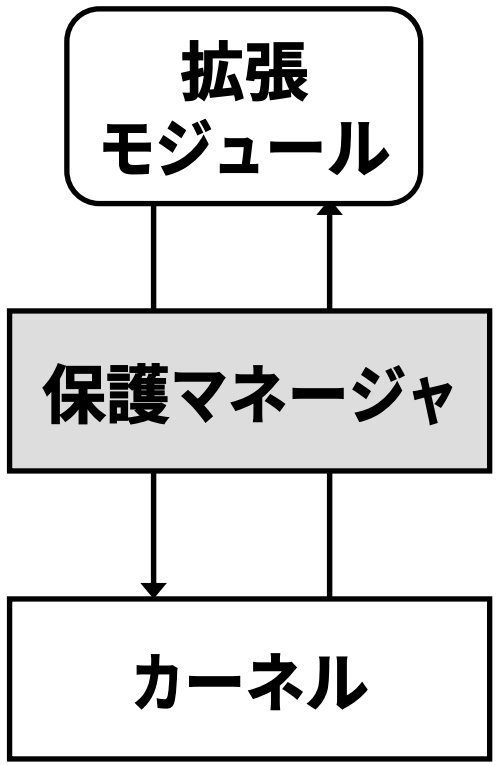
多段階保護機構の提案

- 拡張モジュールの保護レベルを変えてOSに組み込める
  - モジュールのソースコードを書き換えることなしに保護レベルを変えられる
  - fail-safeの能力と実行性能のトレードオフをとれるように様々な保護レベルを提供する
    - カーネルに組み込むこともできる

ファイルシステムについて設計･実装

システム構成

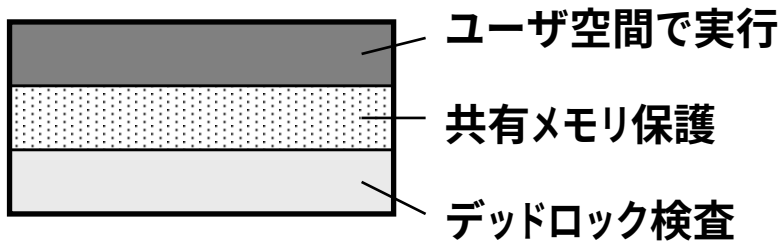
- 拡張モジュール
  - OSの拡張機能を記述する
- 保護マネージャ
  - 多段階保護機構を実現する
- カーネル
  - 拡張モジュールを必要に応じて呼び出す



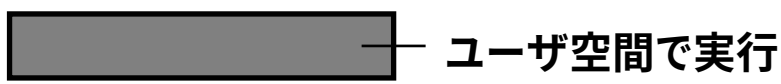
保護マネージャ

- 現在の実装では以下のエラーを検出可能
  - 不正メモリアクセス
  - デッドロック
- どの程度の検出の機能を使うかを選ぶことができる
  - 機能が少ない方が性能が良くなる

保護マネージャの例1



保護マネージャの例2

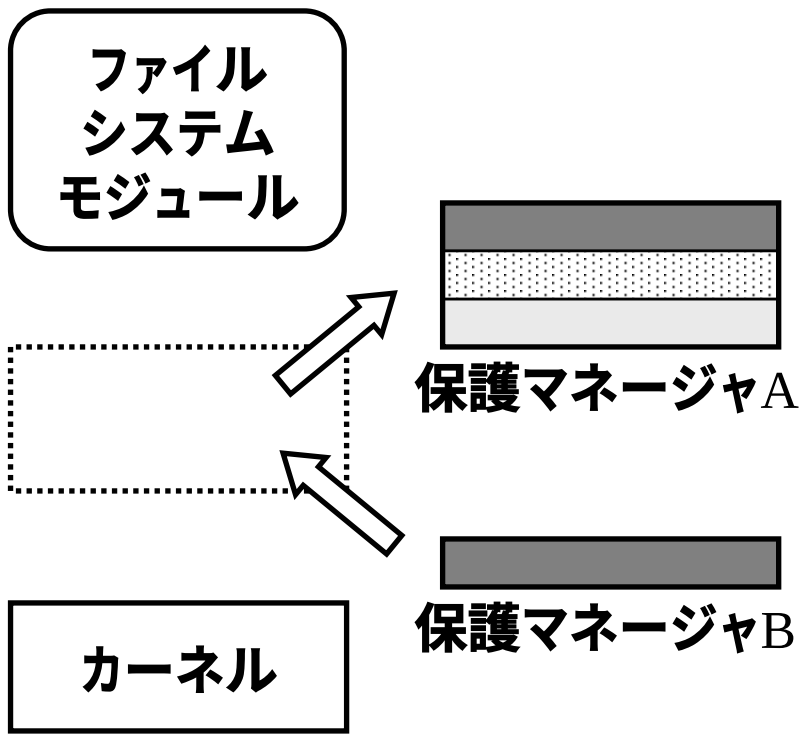


保護レベルの変更

手順

1. 保護マネージャを交換し拡張モジュールと再リンクする
2. 拡張モジュールを再起動する

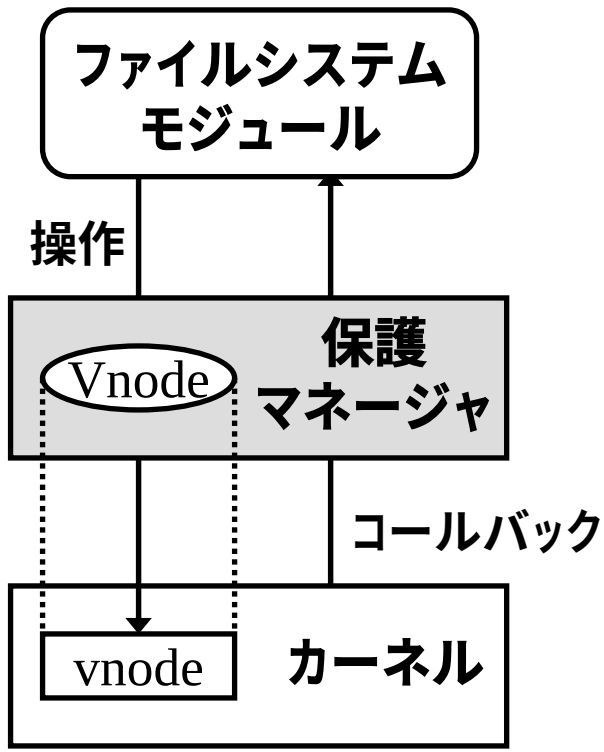
保護マネージャの実装の違いを隠す必要がある



保護マネージャのAPI

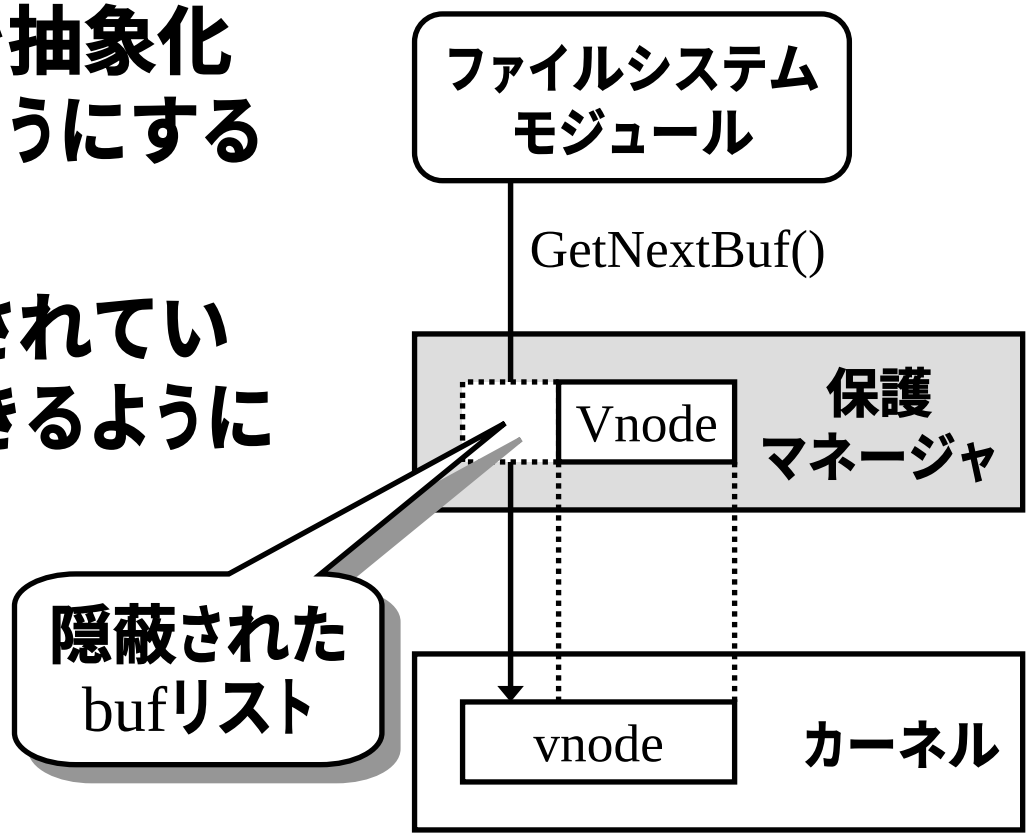
- 抽象度を高めたカーネルデータ構造を見せる
- 提供するAPI
  - カーネルデータを操作するAPI
  - コールバックを登録するAPI

保護レベル変更時に拡張モジュールのソースコード書き換え不要



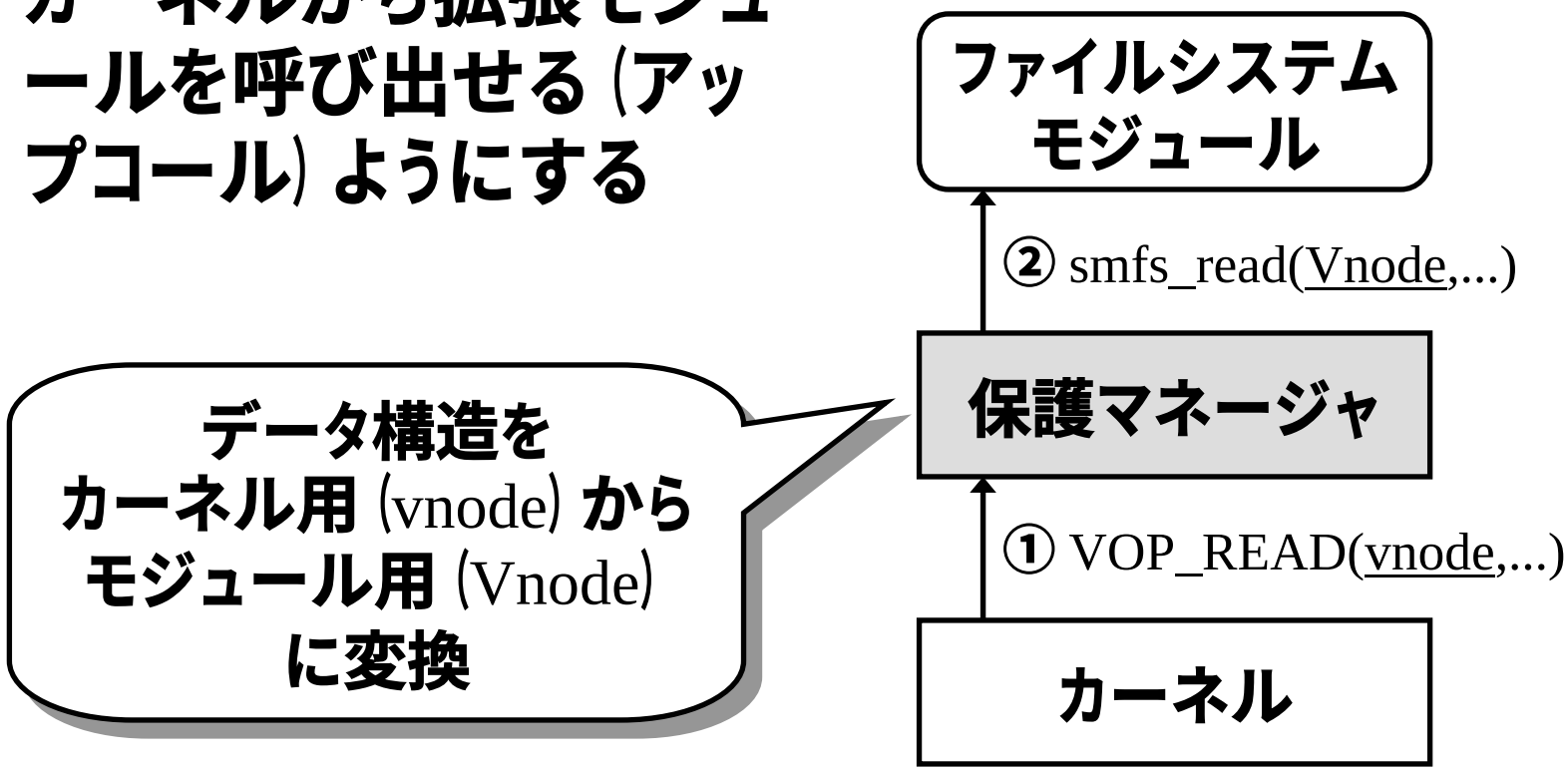
カーネルデータを操作するAPI

- カーネルデータを抽象化した形で扱えるようにする
  例：GetNextBuf()
- カーネルで提供されている機能を利用できるようにする
  例：Bread()



コールバックを登録するAPI

- カーネルから拡張モジュールを呼び出せる（アップコール）ようにする

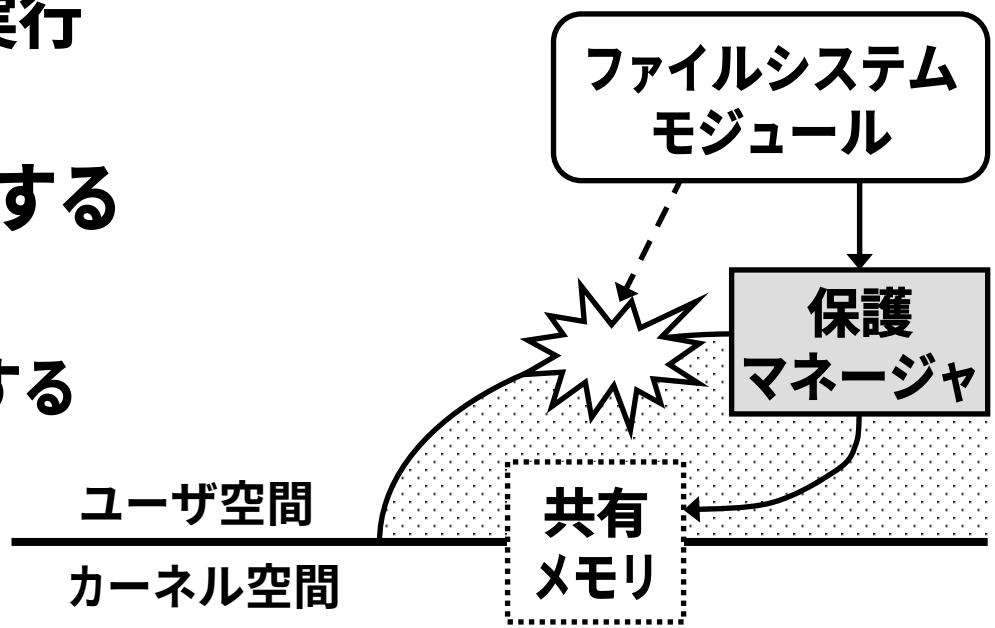


保護マネージャの実現

- 保護マネージャの実現可能性を示すために、本システムをNetBSD上に実装した
  - 以下のエラーを検出･回復することができる
    - 不正メモリアクセス
      - 例外が発生するもの
      - 意味的なもの
    - デッドロック
  - あるエラーの検出の機能が不要ならば、そのエラーに対する保護をやめることができる

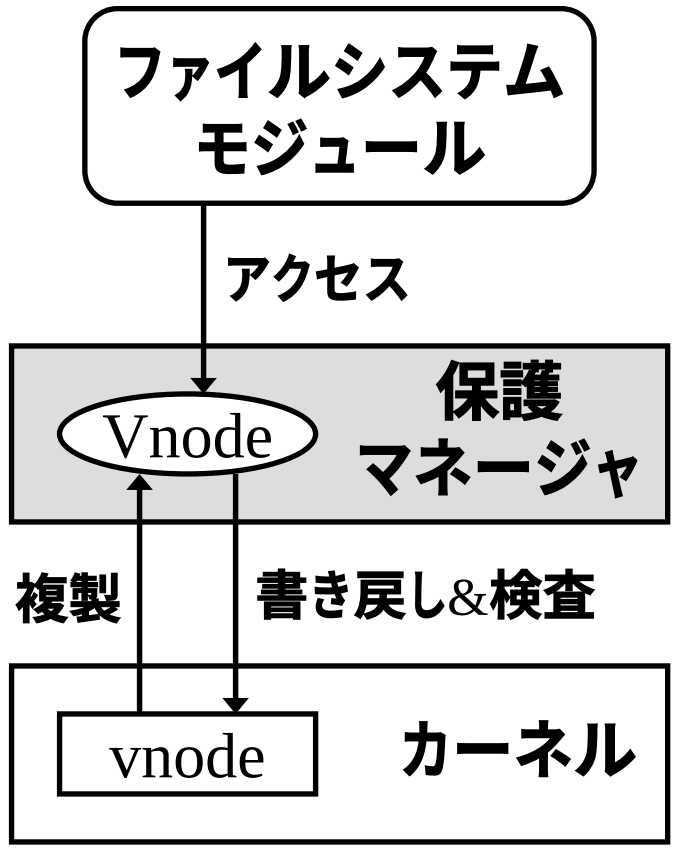
例外が発生するメモリアクセスの検出

- 拡張モジュールの実行環境を選ぶ
  - ユーザ空間で実行
    - 共有メモリを使ってカーネルと通信する
  - カーネル空間で実行
- 共有メモリを保護する
  - アンマップする
  - 読み出し専用にする
  - 保護しない



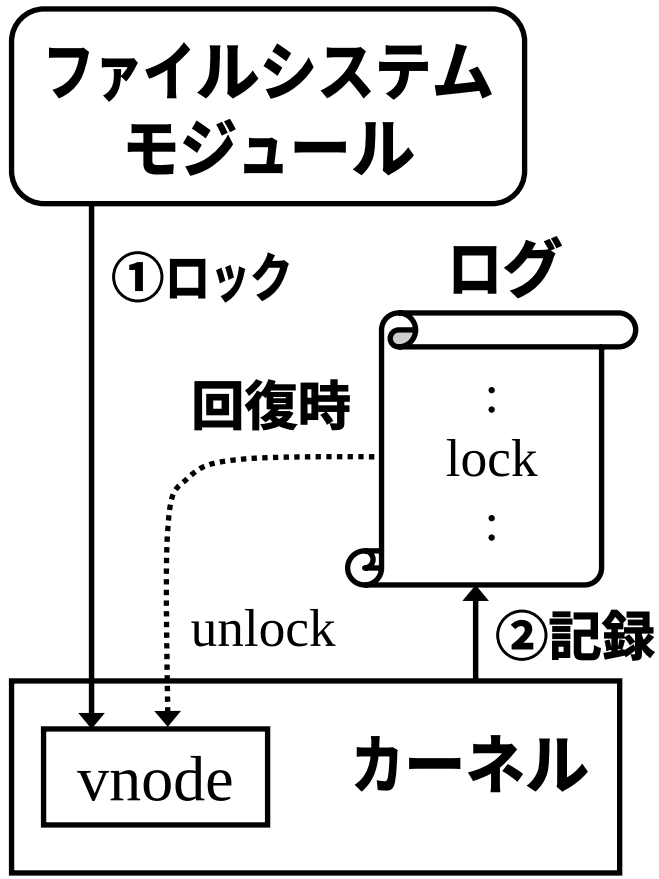
意味的に不正なデータの更新の検出

- カーネルデータを複製する
  - 書き戻し時にデータの内容を検査する
    例：参照回数は0以上
- どの種のデータをどの程度検査するかを変えられる
  例：ポインタを手繰って調べるかどうか



不正メモリアクセスからの回復

- カーネルデータの変更を元に戻す
  - 操作をログにとっておく
  - ログ中の操作を元に戻す操作をする
- 一部の操作を記録しないことができる
  例：メモリの取得・解放

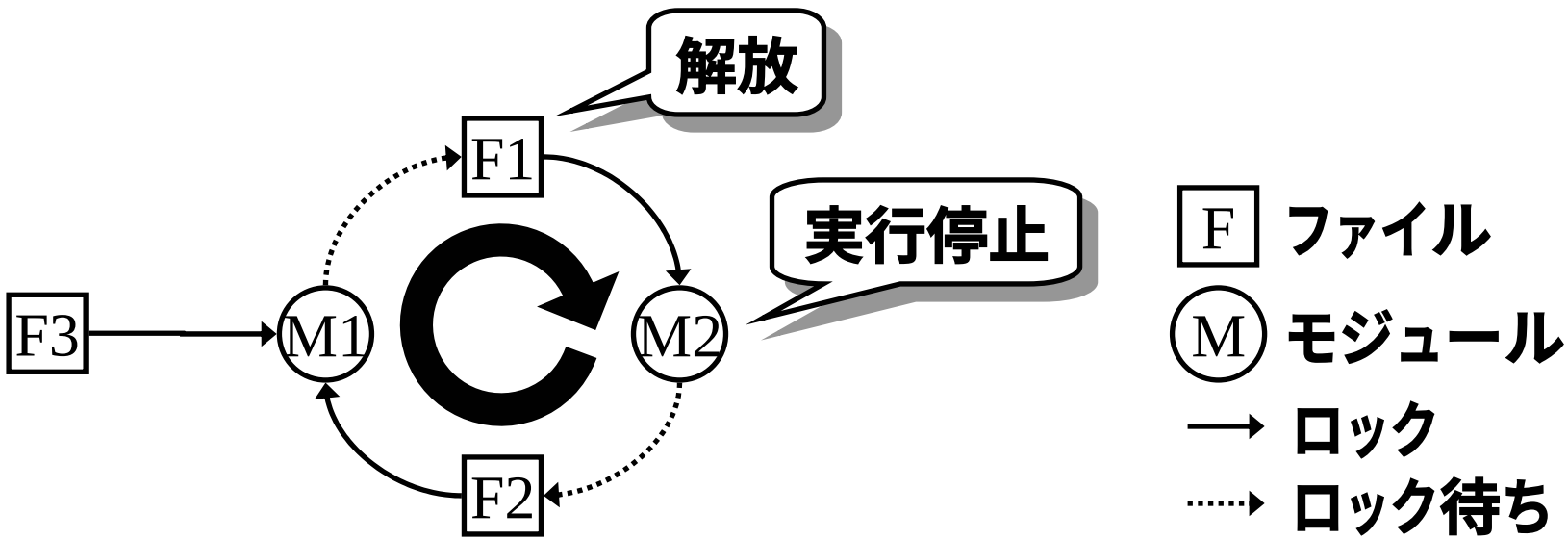


デッドロックの検出

- 定期的に検査する
  - ロック・ロック待ちを記録しておく
  - wait-for-graphで閉路があるかどうか検査する
- 検査の間隔を変えられる
  - オーバヘッドを減らすには間隔を長くする
  - デッドロックを早く検出するには間隔を短くする

デッドロックからの回復

- **デッドロックを解消する**
  1 wait-for-graph**のどこに閉路があるかを調べる**
  2 **閉路中のロックを一時的に解放する**



実験

- 保護レベルを変えて性能を改善できることを示すために実験を行なった
  - MFS(Memory File System)、NFSの2つのモジュールを作成した
  - それらの保護のオーバヘッドを測定した
    - 64KBのファイルをコピー（8KBブロック）
  - 比較のために最初からカーネルに作り込んだものについても測定した

環境：SPARCstation 5 (MicroSPARC2/85MHz)

実験結果 (1)

保護	*MFS*	*NFS*
複製	59	82
アドレス空間切替え	44	104
共有メモリ保護 (アンマップ)	317	433
共有メモリ保護 (読み出し専用)	197	326
デッドロック検査 (1 回当たり)	0.1	0.1
ログの記録	0	21

表1：保護のオーバヘッド(ms)

保護レベル	*MFS*	*NFS*
最大	459	1,150
最小 (カーネル内で動く)	38	516

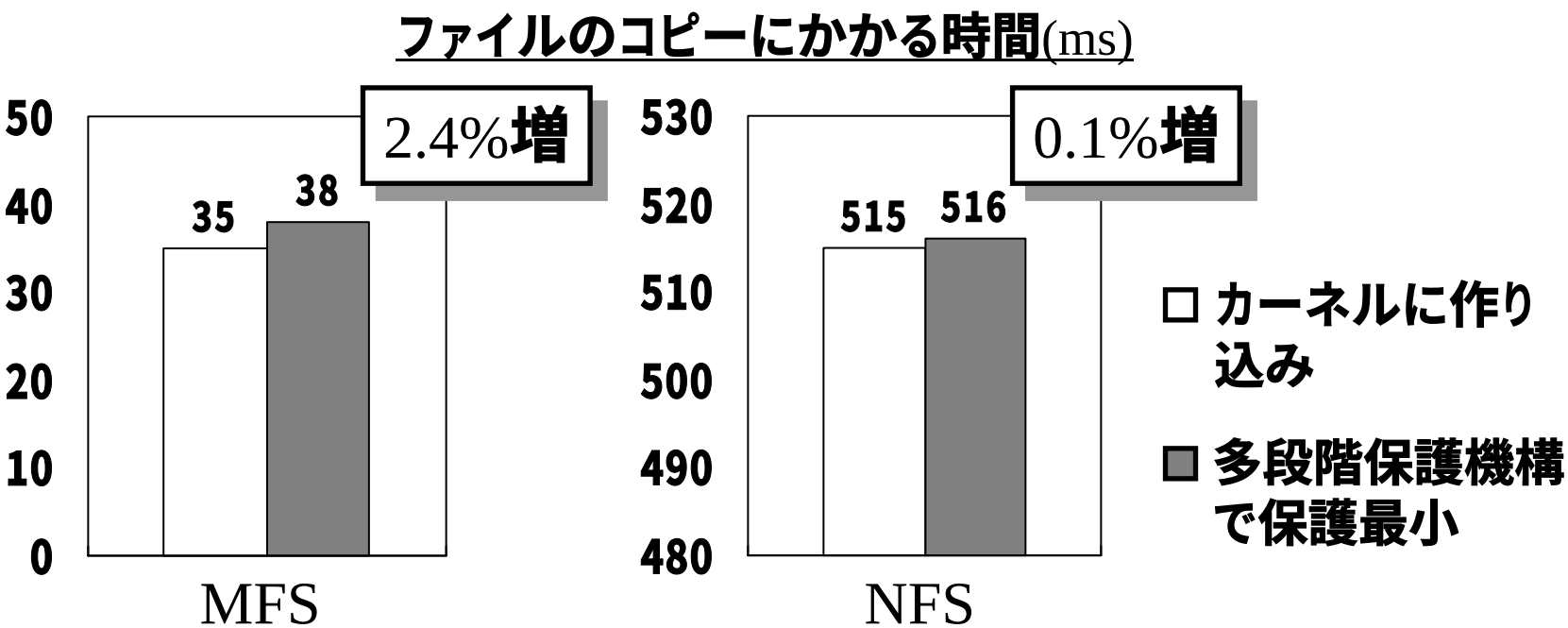
表2：コピーに要する時間(ms)

12倍

2.2倍

保護のオーバヘッドは実用に耐えられる程度である

実験結果 (2)



拡張モジュールの保護を最小にして
カーネルに組み込めば十分な性能が得られる

関連研究

- Mach[Accetta86]
  - **ユーザプロセス機構で**fail-safe**を実現するため実行性能が犠牲になる**
- SPIN[Bershad94]
  - **型安全な言語でメモリアクセス違反だけを検出することができる**
- VINO[Seltzer94]
  - SFI**やトランザクションで**fail-safe**を実現するが**fail-safe**の能力は変えられない**

まとめ

- 多段階保護機構を提案した
  - 保護マネージャにより拡張モジュールを変更なしに様々な保護レベルでOSに組み込める
- 保護マネージャが実装可能であり、保護レベルを変えて性能を改善できることを示した

今後の課題

- 多段階保護機構の汎用性を高める
  - ネットワークシステムなど他のサブシステムにもこの機構を実装する
- 拡張モジュールを停止させずに保護レベルを変えられるようにする