CELF Technical Jamboree #9

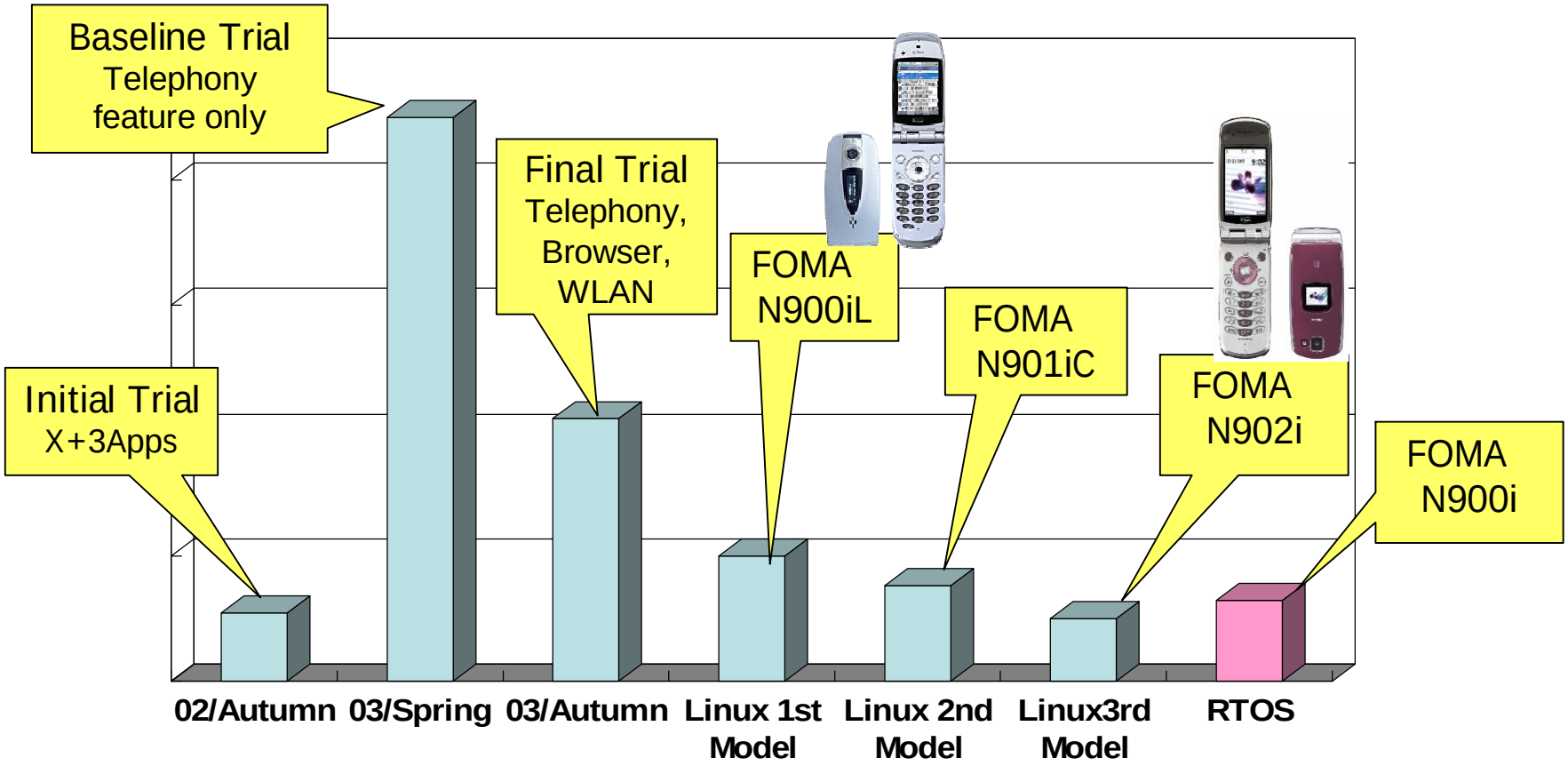
Linuxケータイ 性能改善への取り組み

2006年7月13日
NEC
モバイルターミナル開発事業部
技術マネージャ　福永　雅次郎

目次

1. NECケータイの性能改善の歩み

- **性能：　システム起動性能は約10倍**
- **機能：　音声通話のみ→TV電話/ブラウザ/Java**

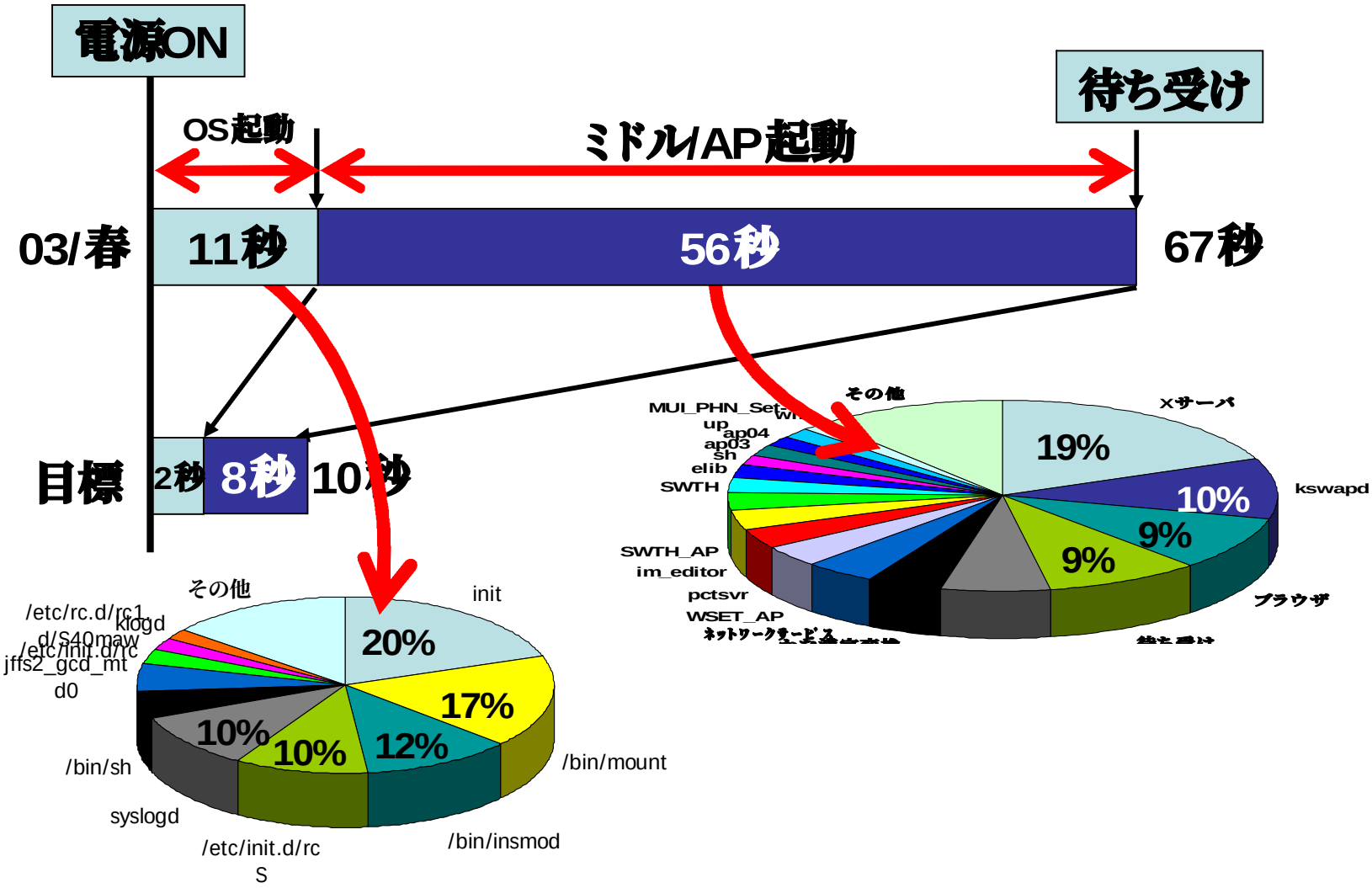


2. 性能上の課題

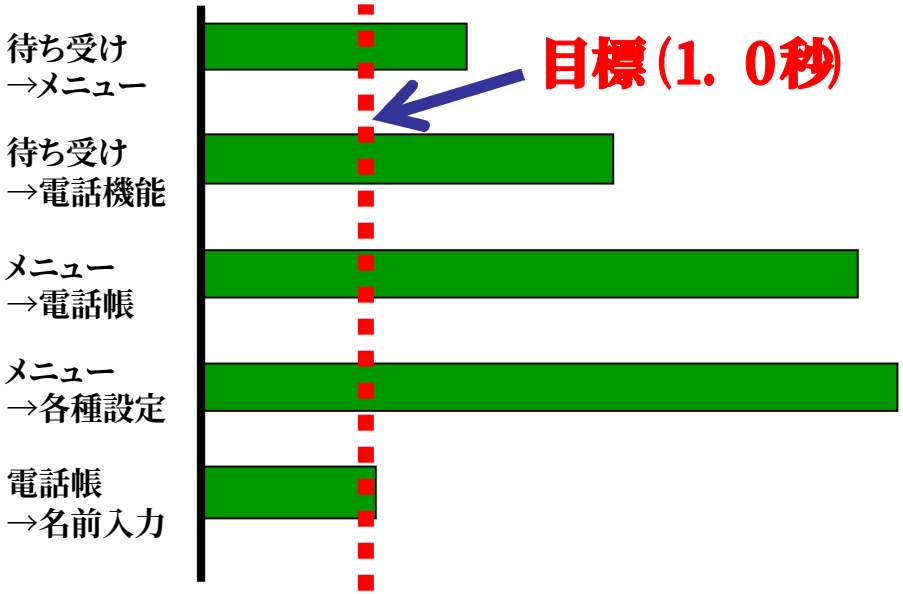
- **起動時間**
  - **当初は約90秒**(03/春)
  - **製品化のためには10秒程度への短縮が必要**

- **アプリ間の画面切り替え**
  - **当初は5～10秒**(03/春)
  - **製品化のためには1秒以内への短縮が必要**

2.1 システム起動時間

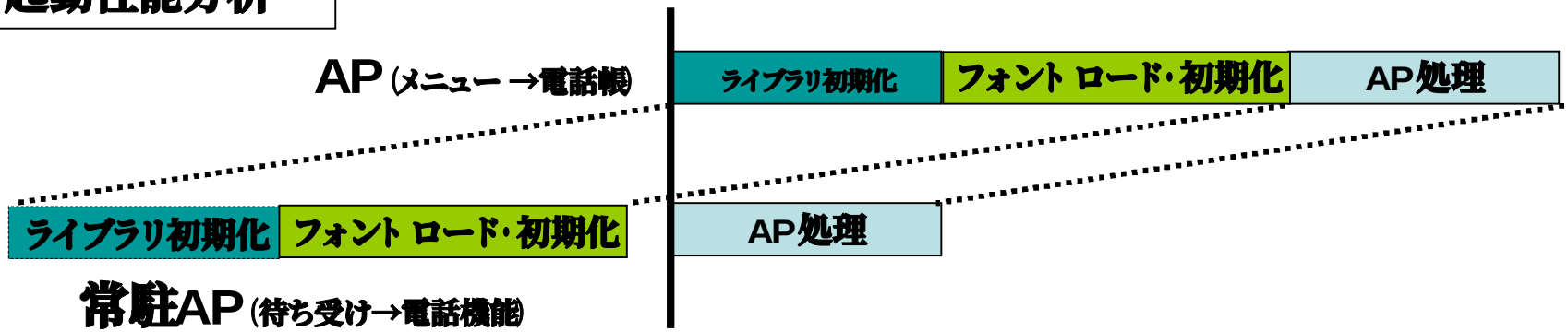


2.2 画面切替性能の検証



	時間
待ち受け→メニュー	1.7秒
待ち受け→電話機能	2.5秒
メニュー→電話帳	3.9秒
メニュー→各種設定	4.2秒
電話帳 → 名前入力	1.1秒

AP起動性能分析



3. 性能改善への施策

3.1 性能改善策

- 起動時間（OS起動時）
  - カーネルチューニング、ログ抑止、
    ファイルシステムのマウント高速化

- 起動時間（ミドル／常駐アプリ起動時）
  - プロセス起動高速化、非常駐プロセス導入、
    スワップ撲滅、描画処理改善

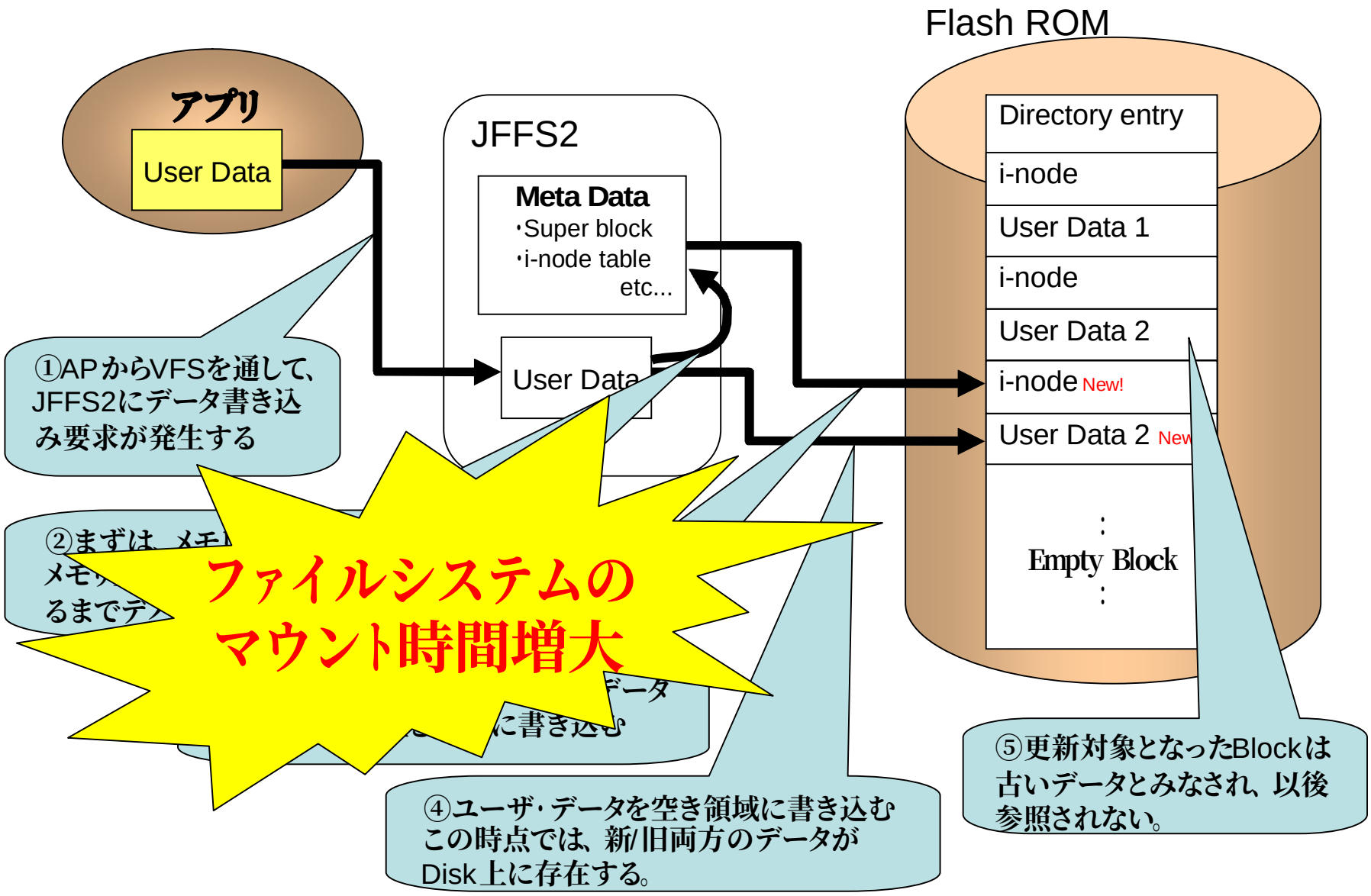
- 画面遷移時間（待受け→メニュ切替時）
  - ウィンドウ部品削減、描画処理改善

3.2 JFFS2マウント高速化

(1) JFFS2 ファイルシステム

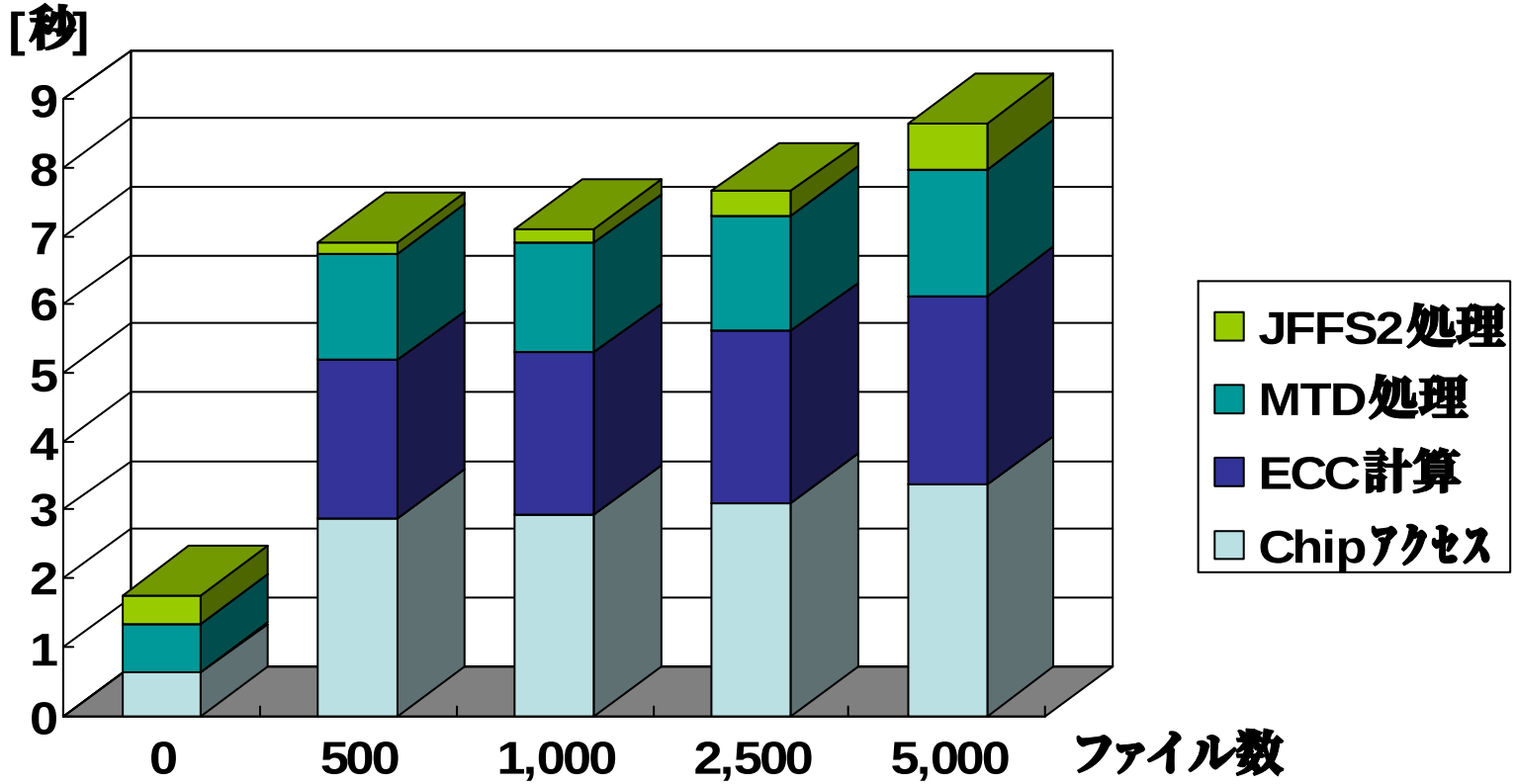
- **JFFS2の特徴**(http://www.linux-mtd.infradead.org/)
  - Flash ROM向けファイルシステム
  - ログ構造
    - 複数のノードをログとして記録し、構成
    - 各ノードは、ファイル情報(データ、ファイル名, 属性など)を管理
  - ファイルデータを圧縮して管理
    - ext2と比較して、データサイズが約40%減少
  - マウント時に、i-node等のテーブルを作成
    - マウント時にファイル管理領域をRAM上に作成
    - データ書き込み中に電源断が発生しても「ファイルシステム自体の破壊」を防止

(2) JFFS2によるファイルアクセス



(3) JFFS2のマウント時間

n ファイル数に依存し、マウント時間が増大

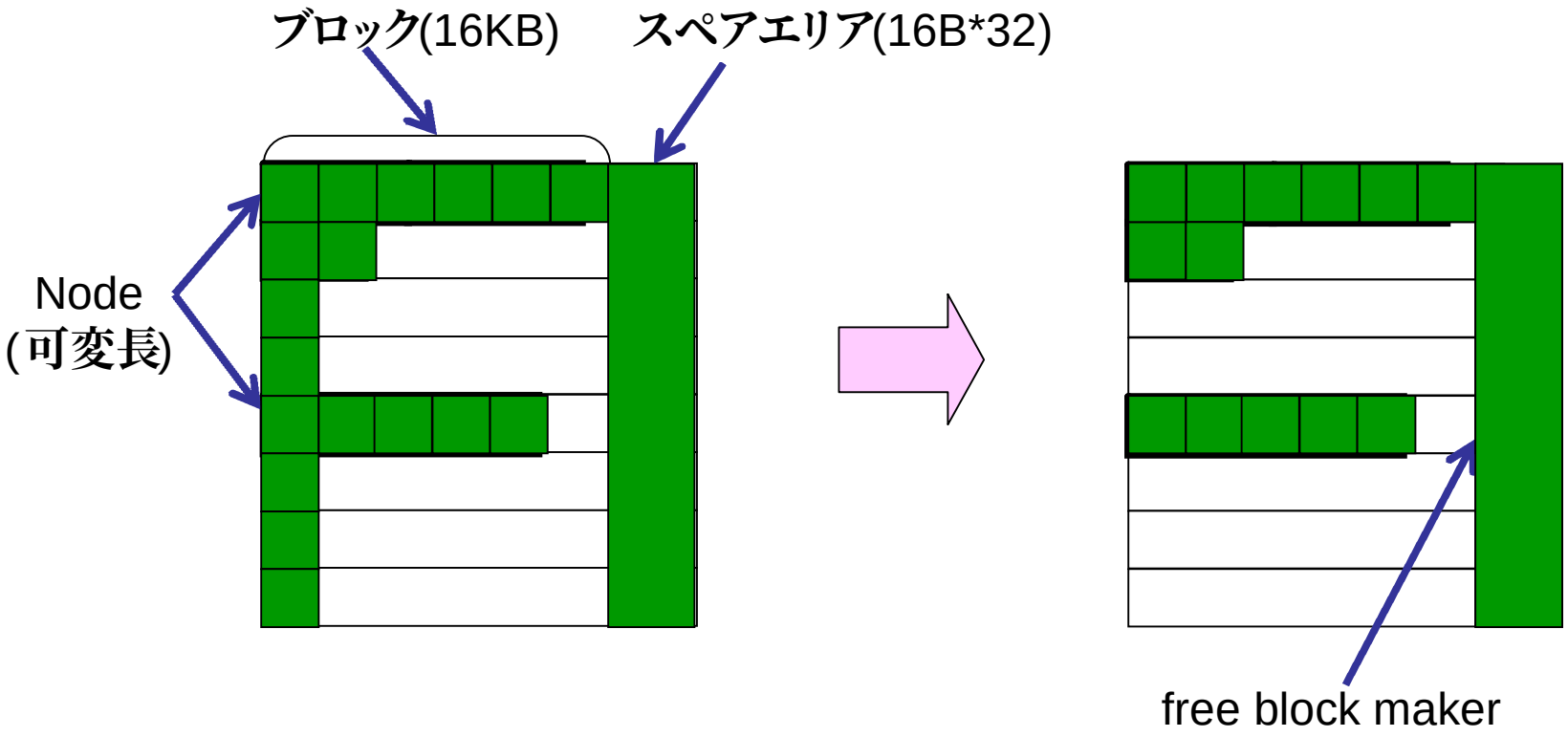


(4) JFFS2マウント処理の高速化

- n 未使用ブロック判断による最適化
- n ファイル管理情報のみ読み出し
- n エラー訂正の簡略化
- n エラー訂正ロジックの改善

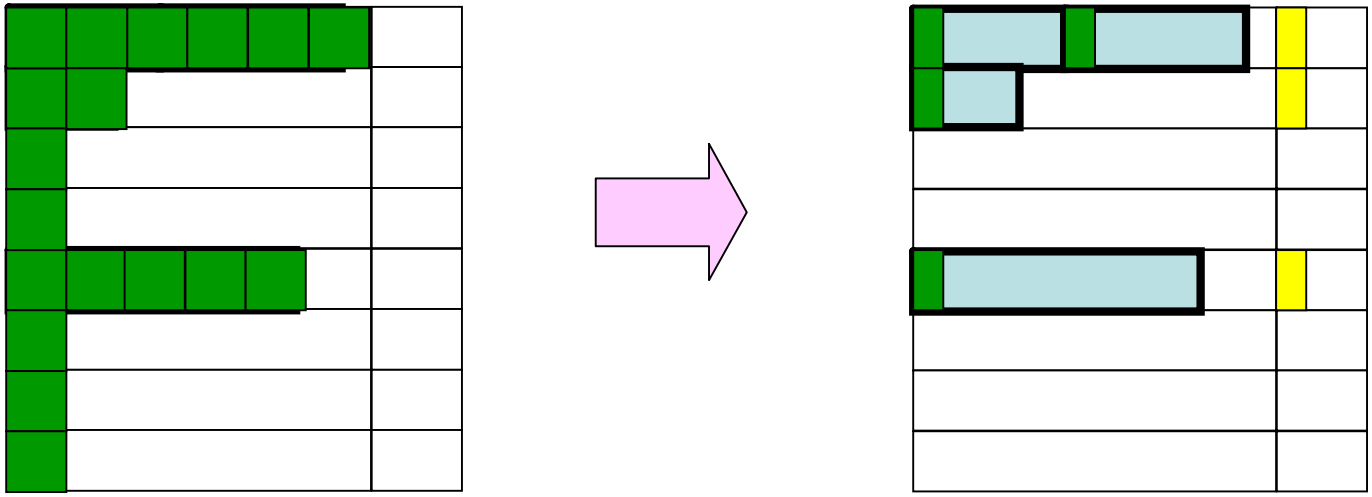
未使用ブロック判断による最適化

- n ブロック毎にNodeの存在有無を示すフラグ（free block maker）を設け、Nodeの存在しないブロックのリードを省略



Node Headerのみ読み出し

- **マウントに必要なNode Headerのみリード**
  - **JFFS2ファイルシステムは、FROMのすべての領域をリードし、ファイル管理情報を構築していた。**
  - **本来マウント処理では、各Nodeの先頭にあるNode Headerを調べれば可能**



エラー訂正の簡略化

- MTDでのECCの読み出しと計算を省略
  - FROMはデータのエラー訂正のためスペアエリアにECCを格納、JFFS2自身もNode Header内にCRCがあり、二重チェックされていた
  - JFFS2はECCなしでデータをリード。万一、JFFS2でCRCエラーとなった場合は、ECC付で再度リード処理を行いデータ訂正

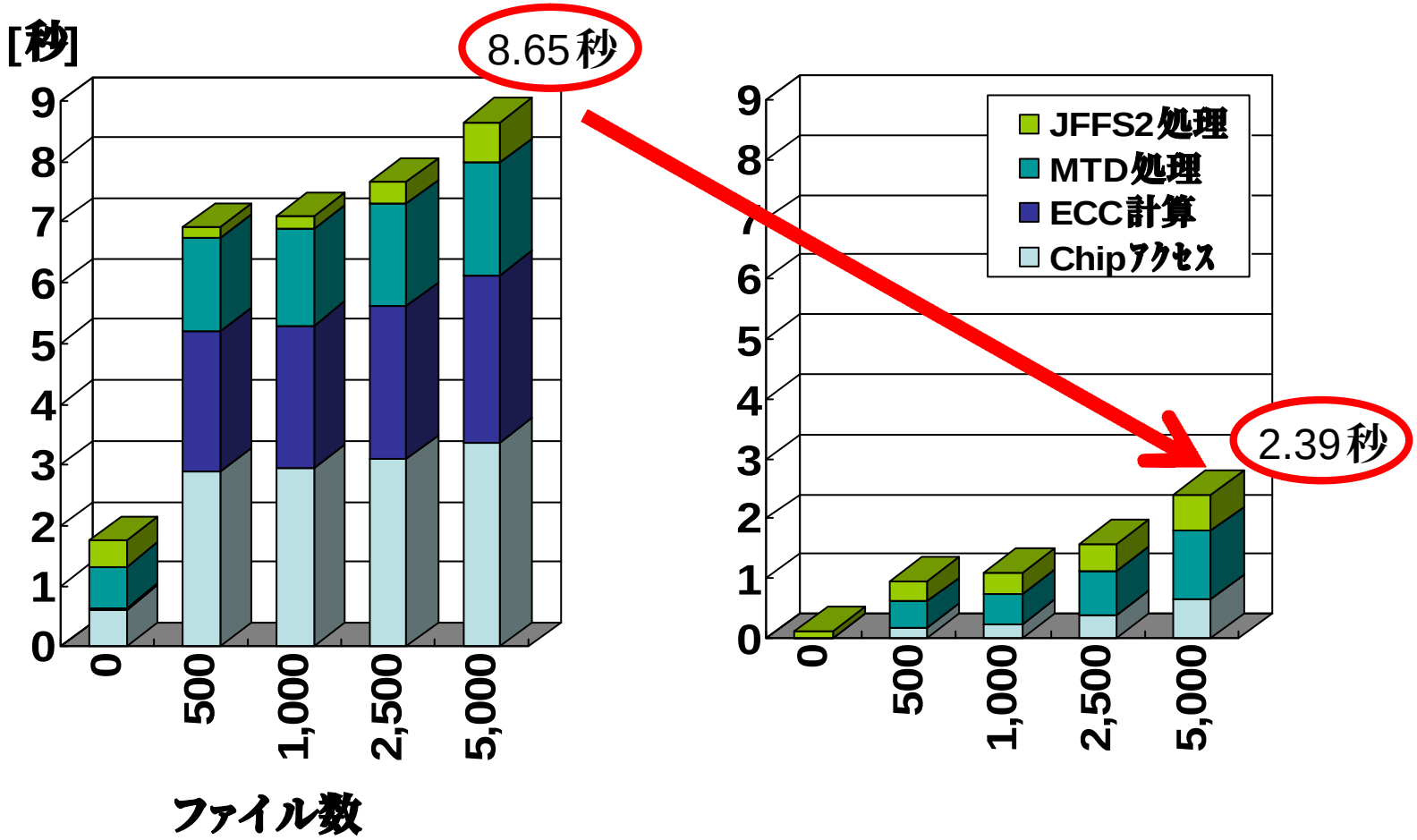
エラー訂正ロジックの改善

- ECC計算ロジックを高速なものに置換
  - MTDドライバのECC処理を、より高速なYAFFS ファイルシステムのECC処理に置換

- 計算処理の最適化
  - ECC計算処理自体もLoop処理の展開により最適化を実施
  - 種々のサンプル結果より、一番高速な展開方法を採用

(5) JFFS2マウント処理の改善結果

n ファイル数が5000個のとき



3.3 プロセス起動の高速化

(1)prelinkの適用

- prelinkとは(http://tree.celinuxforum.org/CelfPubWiki/PreLinking)
  - 実行ファイル作成時にDLLのリンク情報を解決
  - ローダによる未解決シンボルのリンクをスキップ
- 効果
  - fork　2,110mS→169mSに短縮(約80%改善)
  - アプリがリンクするDLL本数が多い場合に効果大
- 適用上の課題
  - Object作成時に“–fPIC” オプションが必須
  - DLL変更時は、それをリンクする実行ファイル すべての再prelinkが必要
  - prelinkによりObjectサイズが2～3割増大

(2)glibcの性能改善

- setlcaleの処理簡略化
  - 通常：　文字や時刻等のフォーマット定義などを
    ファイルから読み出してlocale tableに格納
  - 必要な情報のみをlocale tableに格納

- 効果
  - 処理時間が134mS→2mSに短縮(98%改善)

4. まとめ

対策と効果

- OS起動の改善
  - JFFS2のマウント性能に着目し改善を実施
  - ファイル5000個： 8.65秒 → 2.39秒

- プロセス起動の改善
  - prelink適用： 2,110m秒→169m秒 (fork)
  - setlocale簡略化： 134m秒→2m秒

Empowered by Innovation