フラッシュEEPROMを用いたUNIXファイルシステムの実現

1F－2

攝津 敦　菅井 尚人　山口 義一
三菱電機（株）情報技術総合研究所

1. はじめに

近年、UNIX計算機は、高機能／高信頼の要求から産業用システムに適用されてきている。産業用システムにおいて、屋外設置される制御装置などでは、高い耐環境性を満足するため、可動部分がなくかつ大容量／低コストであるフラッシュEEPROMを補助記憶装置として使用するものがある。

本稿では、フラッシュEEPROMを用いたUNIXファイルシステムの実現方式について述べる。

2. 課題と解決策

2．1 データの消失

高信頼性の観点から不意の電源断でもデータが消失しないようにする必要がある。

[問題点1] UNIXバッファキャッシュ

不意の電源断の発生により、バッファキャッシュに保持されていた書込みデータは失われてしまう。

[問題点2] フラッシュEEPROMの書き込み

フラッシュEEPROMは、書き込む領域を初期化（消去）しないと、データを書き込むことができない。この領域は一定のサイズ（以降、セクタ）を持ち、かつファイルシステムが管理するファイルブロック（以降、ブロック）のサイズよりも大きい。このため、1ブロックのデータの更新では、セクタ内にある他のブロックのデータを一旦他の領域にコピーしデータを更新した後、セクタを初期化し、そのセクタに再書き込みを行わなければならない。ここで、セクタ初期化中に電源断が発生すると、セクタ内のデータは失われてしまう。

[解決策]

UNIXバッファキャッシュは同期書き込みとし、フラッシュEEPROM更新時のセクタデータを保持する領域（セクタバッファ）をバッテリバックアップメモリに割り当てる。

2．2 書き込み性能の劣化

読み込みはフラッシュEEPROMのH/W性能により、ほぼHDDと同じ性能を達成することができる。今回適用予定の制御装置では、書き込みはインストールまたはメンテナンス時のみとなるため、その性能に対する重要度は低い。しかし、下記問題点により、上位からの書き込み要求毎にフラッシュEEPROMに書き込みを行う単純方式の場合、数MBのアーカイブファイルの展開作業に10～20時間程度掛かってしまう。よって書き込み性能を向上させる必要がある。

[問題点1] フラッシュEEPROM H/W特性

フラッシュEEPROMへの書き込みにはセクタの消去が必要である。消去に要する時間はチップにもよるが約1～2秒程度である。上位層から来るデータを常に書き込んでいた場合、1ブロック毎にセクタの消去時間を要することになる。

[問題点2] UNIXファイルシステムの書き込み特性

図1は新規ファイル作成における、ブロック書き込み要求の様子を示している。

1．スーパブロック(i-node数更新)
2．親ディレクトリファイル(コピー先ファイル名の登録)
3．親ディレクトリ i-node(時刻の更新)
4．新規ファイル用i-node（データ設定）
5．フリーブロック管理ブロック（新規ファイル用ブロックの確保）
6．スーパブロック（フリーブロック数更新）
7．確保したブロック（データ書き込み）
8．新規ファイル用i-node（確保したブロックを登録）
以降、5～8をファイルサイズ分繰り返す

図1．新規ファイル作成におけるブロック書き込み

図のようにUNIXファイルシステムでは、スーパブロック、i-node、ファイルブロックの各領域に分散して書き込み要求が発生し、かつその同じ領域に対する書き込みも多い。このため、UNIXバッファキャッシュを同期書き込みとした場合、上位からの書き込み要求はそのままフラッシュEEPROMに対する書き込み要求となるので、小

An Implementation of Unix File System using of Flash EEPROM.
Atsushi SETTSU, Naoto SUGAI,
Yoshikazu YAMAGUCHI
Mitsubishi Electric Corp.

さなファイルの作成でもフラッシュ EEPROM へ多数の書き込みが発生することが予想される。

[解決策]

バッテリバックアップメモリ上に複数のバッファを持ち、バッファリングを行う。

2．3 バッテリバックアップメモリの容量

[問題点]

バッテリバックアップメモリは、容量が限られている。そのため、セクタバッファのような大きなサイズのバッファを大量に取ることはできない。

[解決策]

セクタバッファを1個のみとし、ファイルシステムのファイルブロックと同じサイズのバッファ（ファイルバッファ）を複数設ける構成にする。

3． 実現方式

上述した解決策から以下の実現方式に決定した。

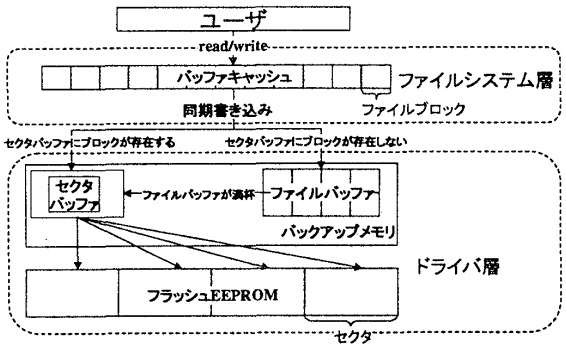

図2．実現方式

ファイルシステム層は、バッファキャッシュを同期書き込みにすることにより、ユーザデータの消失を防ぐ。

ドライバ層では、上位から来たブロックがセクタバッファに含まれていれば、セクタバッファを更新する。セクタバッファに存在せず、ファイルバッファに存在していれば、ファイルバッファを更新する。ファイルバッファが満杯の場合は、セクタバッファの内容をフラッシュ EEPROM に書き戻した後、ファイルバッファの内容をセクタバッファに移すことによりファイルバッファに空きを作る。これにより、フラッシュ EEPROM への書き込み回数を低減し書き込み性能の劣化を防ぐ。またセクタバッファ、ファイルバッファともにバッテリバックアップメモリに配置し、セクタバッファを1個、ファイルバッファを数個という構成にすることにより、バッテリバックアップメモリの使用容量を抑え、かつ不意の電源断でデータが失われることを防ぐ。

4． 性能試算

ファイルバッファの個数を決定するために、バッファ数を変化させた場合のフラッシュ EEPROM への書き込み回数を試算した（図3）。

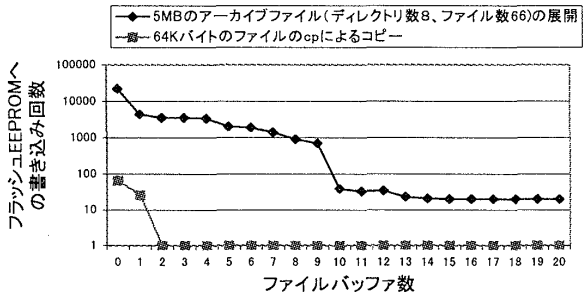

図3．ファイルバッファの効果

結果、ファイルバッファを 10～16 個程度備えることで、書き込み回数が格段に低減できることが判明し、今回の実現方式では 14 個にすることに決定した。表1に 5MB のアーカイブ展開時の HDD 性能値、および上位からの書き込み要求毎にフラッシュ EEPROM に書き込みを行う単純方式と本実現方式の場合での性能試算値を示す。

表1．5MB アーカイブ展開時の性能試算結果

HDD	10 秒
フラッシュ EEPROM(単純方式)	18 時間 54 分 10 秒
フラッシュ EEPROM(実現方式)	**2 分 10 秒**

本結果より、HDD の 1/12 の性能ながら、単純方式に比べ 500 倍程度の書き込み性能が確保できる見込みが得られた。

5. おわりに

本稿では、フラッシュ EEPROM を用いた UNIX のファイルシステムの実現方式について述べた。

バッファキャッシュを同期書き込みにし、セクタバッファ・ファイルバッファを小容量バッテリバックアップメモリに配置することで、データ消失を防ぎ、実用レベルの書き込み性能を確保できる見込みが得られた。

今後は、試作・検証を行っていく予定である。

参考文献

[1]川口 敦生，西岡 真吾，元田 浩，フラッシュメモリを用いた UNIX ファイルシステム，情報処理学会第 49 回全国大会予稿集，5U-5，1995