社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－MBL－36（55）
2006－UBI－10（55）
2006／2／17

不揮発性記憶装置の効率的利用の研究

三浦史光

日本電信電話株式会社　サイバースペース研究所

あらまし 近年の不揮発性メモリ素子の発達は、近い将来に不揮発性主記憶を可能にしつつある。しかし現在の OS やアプリケーションは不揮発な主記憶を利用するようになっておらず、主な利用形態はファイルシステムとしての利用にとどまっている。ファイルシステムとしての利用は互換性の観点では望ましい。しかし、ファイルの操作は OS のカーネルを介してしか操作できないので、コンテクストスイッチを必要とする。アプリケーションは直接主記憶を操作できるので、OS のカーネルを介さないで永続的主記憶を利用できる可能性がある。本研究は不揮発性記メモリ素子を主記憶として使う方法を検討し、オーバーヘッドを削減する。

Non-volatile memory element and its effective utilization

Fumiaki Miura

Nippon Telegraph and Telephone Corporation Cyber Space laboratories

Abstract: Non-volatile memory device technologies have developed. Now we expect main memories constructed with non-volatile memory devices. On the other hand, OS and applications are still designed on the assumption that main memories are volatile. So, the typical usage of non-volatile memory devices are filesystems at present. The usage for filesystem is good from the view point of API compatibility. But only the kernel can handle filesystem, and other programs must make requests against the kernel. It requires context switches. Applications can directly handle main memories as the nature. We studied that applications directly handle non-volatile memories.

1. はじめに

初期の計算機では主記憶に様々な装置が使われていた。その中には磁気ドラムやコアメモリなどの不揮発性メモリ装置も含まれていた[1]。不揮発性メモリは給電を断った後でも情報を保持することが可能である。しかし当時はバッチ処理が中心であったため、それぞれのプログラムがメモリ全体を占有していた。すなわち、メモリ中の情報のライフタイムはプログラムのライフタイムを超えることがないので、プログラムより長いライフタイムの情報を保持できる不揮発性メモリを活かす機会は多くなかった。また、昔の OS には充分な保護機構がないことが多く[2]、ブート過程を一部省略できる場合がある以外の利益はなかったようである。

現在の計算機は MMU を中心とする技術によって充分な保護機構を備えるようになっている。が、主記憶は SRAM や DRAM などの揮発性メモリを使うことがほとんどなので、現在の OS を始めとするプログラムは依然として主記憶が不揮発性メモリで構成されている場合の考慮をせずに設計・運用されている。

しかし、近年の MRAM・FeRAM などの不揮発性メモリ素子の研究により、近い将来に主記憶の一部を不揮発メモリ素子で構成することが現実的になると予見されている[3]。

本研究は、この不揮発性主記憶を有効に利用する方法を提案する。

2. 関連研究

1990 年ぐらいまでは不揮発性メモリを主記憶の一部に使うデータベースの研究が散見される[9,10]。これらの研究での不揮発性メモリは主としてバッテリバックアップつき DRAM である。DRAM にはリフレッシュ動作が不可欠であるが、性能を犠牲にせず、安定してバッテリバックアップできるようにするのが困難であったのか、その後はあまり見られない。たとえばメモリデータベースにも使われていない[8]。

最近はフラッシュメモリの登場で新たな可能性が出てきた。が、フラッシュメモリは使い方が特殊[1]であるため、そのまま主記憶として使うには好適でない。このためやはりファイルシステムに仕上げ、フラッシュメモリを直接操作するのはカーネルだけに限定している。標準の Linux カーネルに入っている JFFS や JFFS2 がその例である。MRAM や FeRAM ならば主記憶と同様に操作できるのでファイルシステムにしない選択肢もあるはずだが、多くの研究ではファイルシステムを作っており[4,5,6,7]、ファイルシステムではない研究[11]は少なく、主記憶としてそのまま用いる研究は更に少ない[12]。本研究では不揮発性メモリをアプリケーションが直接操作して速度を向上させることを検討する。

3. 不揮発性メモリをファイルシステムに使う場合の問題点

ディスク装置へのアクセスは CPU から複数のコントローラを操り、SCSI などのバスを経由して操作する必要がある。その後機械装置の動作を待ち合わせ、バスを経由して結果が返り、それを割り込みとして処理する(図 1)。その応答時間は DRAM と比較して 10,000 倍以上である。現在の多重プログラミング環境でディスク装置を操作したり割り込みを処理したりできるのは OS のカーネルだけである[2]から、アプリケーションがディスク(ファイル)操作を要求すると、カーネルへのコンテクストスイッチとカーネルからのコンテクストスイッチが必要である。コンテクストスイッチはパイプラインを止め、キャッシュの有効性を減らしたりするので、速度低下の大きな要因となる。

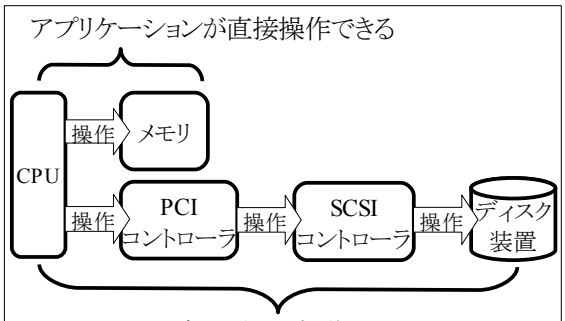

図 1: メモリとディスクのハード的な違いの模式図

ディスク装置を操作する場合には以上のコストは不可欠なので容認するしかない。

一方、現在は様々なファイルシステムが存在し、その中にはディスク装置を用いないものもある。バッキングストアとしてフラッシュメモリを用いる JFFS や JFFS2、MRAM を用いる[3]HeRMES[5]、不揮発性メモリを用いる PRAMFS[7]、不揮発性メモリとディスク装置を併用する Conquest[6]などがある。

ファイルシステムに仕上げる利点として、新たな API を必要としない、すなわち既存のプログラムがそのまま動作する点がある。ファイルシステムに対する API の基本的な部分は変化していないことがその理由であるが、ファイルシステムはディスク装置を念頭に置いて設計され、ディスク装置以外に用いる場合にもその基本的設計方針は受け継がれているので、バッキングストアにメモリを用いる場合には効率や操作性に問題がある。つまり、これらのファイルシステムを用いる場合は主記憶にあるデータの操作を OS のカーネルに依頼し、コピーやブロッキングなどを繰り返して読み書きすることになる(図 2)。し

1 書き方が異なる(NAND 型では読み方も異なる)だけでなく、特定箇所ばかり書き換えないようにするにはカーネルによる管理が望ましい。

2 任意のアプリケーションにディスク装置の操作を許すと、ディスク装置を破壊してしまう虞があるし、アクセス制御ができない不都合などもある。

3 ソフトウェアから見た MRAM と FeRAM の大きな違いは後者が破壊読出しであることである。すなわち、FeRAM を読み出している時に給電が断たれると情報が消えてしまう可能性があるので、場合によっては冗長符号化などの対策が必要である。

かし主記憶の操作はカーネルに依頼しないでできるし、その際ブロッキングなどは必ずしも必要ないはずである。古典的なファイルシステムの API を用いる限り、この問題は避けがたい。本研究では、API の互換性を犠牲にする替わりに[4]、このようなオーバーヘッドを避けることを検討する。

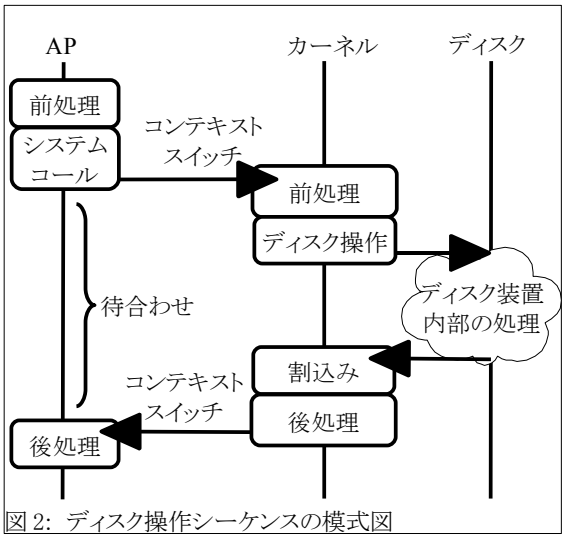

図 2: ディスク操作シーケンスの模式図

4. メモリセグメント

現在の OS はプロセスに対して仮想メモリを提供する機能を持つことが多い。Linux の場合はセグメントという単位で仮想メモリを提供する。たとえばテキストセグメントにはプログラムが入っており、read only かつ executable という属性を付与している。そしてメモリセグメントを利用しているプロセスからはどのアドレスがどのセグメントなのか意識することなく利用できる。メモリセグメントの種類を追加し、不揮発性メモリを扱えるようにできれば、メモリセグメントをプロセスに割り付ける初期化以外はカーネルに操作を依頼する必要がなくなる。このセグメントを作り出すために mmap システムコールを用いることにした。

mmap はメモリセグメントを一つ作り出し、ファイルなどをそのセグメントと対応付けるものであるが、ここでは mmap で不揮発性メモリの領域とプロセスのメモリセグメントを対応付ける働きをさせる。「対応」といっても、ファイルとの対応の場合はファイルに対する読み書きが随時発生するが、本研究の mmap は最初に PTE[5]を設定するだけでデマンドページングを起こさないので初期化時以降はカーネルの介入を必要としない。

なお、ファイルが mmap できるかどうかはファイルシステムにも依存する。JFFS や JFFS2、PRAMFS は書換え可能になるように mmap することができない。JFFS や JFFS2 はフラッシュメモリの書換え回数を抑えたいのが理由だと思われる。PRAMFS はファイルシステムとしての一貫性を保つためである[7]。主記憶としてアクセスできるメモリを用いているのにこれらのファイルシステム上のファイルをメモリと同じ方法でアクセスすることは困難である。

5. 永続性を持つメモリセグメントの特徴

永続性メモリセグメントの検討の結果、永続的メモリセグメントに固有な問題がいくつかあることがわかった。

5.1 アプリケーションの問題

アプリケーションはアドレスのどの範囲が永続性を持つかを知っておく必要がある。そしてこのアドレス範囲のメモリの書換えは、いつプログラムが停止しても問題ない順序にする必要がある。たとえばリスト構造を操作する場合、参照できる必要があるデータが永続的なデータ構造から参照されていない瞬間があってはならない(図 3)。言い換えると、既存のプログラムを改造することなく不揮発性主記憶の恩恵に浴することはできない。このような操作手順はライブラリにすることが好ましいと考えられるので、現在は共通的な操作体系を模索している。たとえば malloc を永続性を扱えるように改造することも考えられるが、malloc の一時的に必要なメモリを確保する用途などは利用シーンが一致しない。また、メモリ領域の細分化は、永続的メモリにおいては深刻

4 ラップするライブラリを使う方法も取れるが、対象のファイルシステムごとに内容を切り替える必要があるため、今回は見送った。

5 page table entry の略

な問題となる。

データベース等ではバッファを固定長の「ページ」単位に分割して使うことが多いので、まずはこのような利用形態を検討することとした。

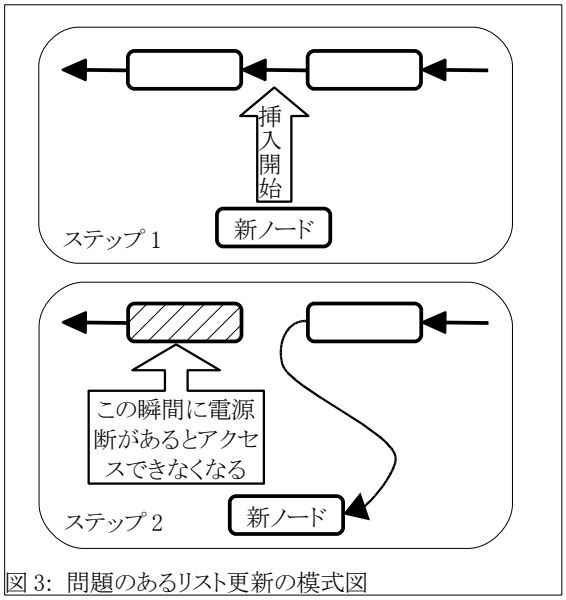

図 3: 問題のあるリスト更新の模式図

5.2 ハードウェアの問題

現在の PC の主記憶は DIMM である。現在普通に入手できるすべての DIMM は synchronous DRAM を使っているようである。仕様を入手できた範囲では、synchronous DRAM と同じメモリサイクルで動作できる MRAM も FeRAM も発見できなかった。つまり、不揮発性 DIMM は当面入手困難である。CPU から主記憶としてアクセスできる他の候補は PCI 族である。DIMM と比べると CPU から PCI 族へのアクセスは遅い。つまり、揮発性主記憶と比較して不揮発性主記憶のアクセス速度は遅くならざるを得ない。この問題を評価するため、PCI -X にメモリボードをさして評価中である。このボードはメモリに書いた内容を通信線を通じて他のボードにコピーする特殊ボード[6]であり、その通信内容を他の機材に保持しておけば、速度だけでなく機能的にも不揮発性主記憶に近いものになる。

なお、組込み用途では主記憶をアドレス範囲ごとに異なるメモリコントローラを適用できるので、そういう場合はペナルティは小さくなるかもしれない。

6 http://www.avaldata.com/products/solution/comunication_01.html

5.3 OS の問題

永続的メモリ装置に対するカーネルの関与は不揮発性メモリの領域ごとにアクセス権を管理することと、セグメントの管理の 2 点である。前者はファイルシステム等と基本的に同じであるので後者について述べる。

セグメントの管理は基本的に PTE の設定である。永続的メモリ装置に対する書込み順序は保存されなければならない[7]ので、write back キャッシュは利用できない。すると write through キャッシュが候補になるが、これは利用しないことにした。その理由はプラットホームとして採用した Linux の PTE の状態管理の中に write through キャッシュがないことである。5.2 で述べたように、不揮発性メモリは少なくともハードウェア的に遅くなる傾向があることが判明しているので、典型的な使い方は、演算等を揮発性メモリセグメントで行い、得られた結果を不揮発性メモリセグメントに書くことになると思われる。従って、write through キャッシュが使えないことはそれほど大きいペナルティではないかもしれない。このペナルティ評価の精密化は今後の課題である。

6. 不揮発性メモリセグメントの使い方

6.1 利用例: 携帯電話

現在入手できる携帯電話のほとんどは、書換え可能な永続的メモリをフラッシュメモリに負うている[8]ようである。ここで利用シーンとして次の場合を考える。

1. 利用者はゲームしている。
2. 電話がかかってくる。
3. ゲームを再開するに必要な情報を永続的に保持[9]し、ゲームを停止する。

7 そうしないとアプリケーションからカーネルに対して同期指示を頻繁に与える必要が生じる。

8 マイクロディスクを備えた携帯電話もあるが、このディスクは間欠給電されていて、頻繁あるいは緊急に必要になるデータを保持するのには使われていないと思われる。

9 通話からこのゲームの再開までの間に電池が空になる可能性を考えると、永続的な保持が望ましい。

4. 通知された発信者番号を元に、電話帳を検索する。
5. 発信者情報と共に着信があることを利用者に通知する。

これら一連の操作を迅速に行わないと、電話をかけてきた人が待ちきれないで呼の切断操作をしてしまうことがある。ここで、ステップ 3 とステップ 4 には永続的メモリの操作が必要である。ファイルシステムではなく永続的主記憶を用いる場合はカーネルに操作を依頼し結果を待つコスト[10]をほぼ 0 にすることができるので、応答速度の向上が期待できる。

6.2 利用例: メモリデータベース

メモリデータベースはデータベースで扱うデータを主記憶に保持して処理を行うものである[8]。検索処理においてはディスク操作が必要ないため高速に処理できる。しかし、更新処理ではディスク操作を必要とするため、更新が多い場合にはメモリデータベースはあまり有効ではない。しかし、永続的主記憶がある場合は更新処理の速度向上が見込める。永続的主記憶の容量が充分であれば単にメモリに書くだけでコミットできる。そうでない場合でも一般に負荷には変動があるため、永続化すべきデータを一旦永続的主記憶に保持し、その後順次ディスク装置に移すスプーリングを行えば、永続的主記憶が一杯にならない限り、メモリに書いた段階で「コミット完了」としてよい。もちろんこの方法はメモリデータベース専用ではない。現在このシステムを設計中である。

7. 結論

現在の計算機での主記憶の使われ方と、不揮発性メモリの開発にミスマッチがあることがわかった。次に、不揮発性メモリをファイルシステムとして使う場合の得失について検討し、オーバーヘッドの観点からはファイルシステムにしないほうが良いことがわかった。そこで不揮発性メモリを主記憶のメモリセグメントとして扱うこととし、ユースケースを用いて有用性を検討した。性能の点では有用だが、アプリケーションや OS およびハードウェアに課題があることがわかった。

8. 今後の課題

残っている課題は多い。まず、malloc に相当する機構を構築しなければならない。ゲームデータの書込みの場合は複雑な書込み順序制御は必要ないと思われるが、メモリデータベースのような場合は複雑な管理が必要になるだろう。これらの扱い方が課題である。この課題はまず、永続的スプーリングの詳細検討で対処したい。

次に、キャッシュ関係のペナルティの評価が必要である。この評価によって、永続的メモリの使い方が変わってくるからである。

そして、最大の課題は不揮発性メモリを「普通に」利用できないことである。その理由の 1 つに、不揮発性メモリを利用するソフトウェアがないことがあると考えられる。本研究が不揮発性メモリの普及に役立てば幸いである。

10 主なコストはコンテクストスイッチ 4 回分である。

参 考 文 献

[1] コンピュータが計算機と呼ばれた時代, 財団法人 C&C 振興財団, ASCII, 2005 年, ISBN4-7561-4677-5.

[2] Per Brinch Hansen, “バッチ処理システム, ”オペレーティング・システムの原理, 田中穂積・真子ユリ子・有沢誠共訳, pp.8-9, 近代科学社, 昭和 51 年.

[3] 「FeRAM 製造技術の開発(フォーカス 21、助成事業)」事後評価報告書, 独立情勢法人新エネルギー・産業技術総合開発機構研究評価委員会, pp.4-5, 平成 17 年, http://www.nedo.go.jp/iinkai/hyouka/houkoku/16h/jigo/30.pdf.

[4] Ethan L.Miller, Scott A.Brandt , and Darrell D.E. Long, “HeRMES: High-Performance Reliable MRAM-Enabled Storage, ”Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, May 2001,

http://www.soe.ucsc.edu/~elm/Papers/hotos01.pdf.

[5] Nathan K.Edel, Deepa Tuteja, Ethan L.Miller, and Scott A.Brandt, “MRAMFS: A compressing file system for non-volatile RAM,” Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications System, 2004.

[6] An-I A. Wang, Peter Reiher, Gerald J.Popek, and Geoffrey H.Kuenning, “Conquest; Better Performance Through a Disk/Persistent-RAM Hybrid File System, ” USENIX, 2002.

[7] “ Protected and persistent RAM Filesystem,” http://pramfs.sourceforge.net/.

[8] Takashi Honishi, Nobuyuki Kobayashi, and Jinnosuke Nakamura, “Design and implementation of an enhanced relational database management system for telecommunication and network applications, ” pp.698-703, Proceedings of Pacific Telecommunications Council Eighteenth Annual Conference, 1996.

[9] Kenneth Salem, Hector Garcia-Molina, “Checkpointing Memory-Resident Databases,” pp.452-462, ICDE, 1989.

[10] George Copeland, Tom Keller, Ravi Krishnamurthy, and Marc Smith, “The Case for Safe RAM,” pp.327-335, VLDB, 1989.

[11] Michael Wu, Willy Zwaenepoel, “eNVy: A Non-Volatile, Main Memory Storage System, ”pp.76-85, VLDB, 1997.

[12] 大竹廉、山崎信行、安西祐一郎、“主記憶に不揮発メモリを用いたシステムの実行状態復元手法”、pp.88-99、情報処理学会誌 Vol.45 No.SIG01 、2004.