不揮発性メモリ品型計算機用オペレーティングシステムの提案

谷口秀夫[†1]

概要：不揮発性メモリの性能向上は著しく、実メモリとしての利用も可能になっている。そこで、本稿では、不揮発性メモリのみを搭載する計算機ハードウェアにおいて、オペレーティングシステムが必要とする機能について提案する。具体的には、実メモリとプロセッサの関係の変化について述べる。次に、実メモリとして不揮発性メモリを利用することで外部記憶装置が不要であることを述べ、活性挿抜できる不揮発性メモリ品を提案し、これにより期待される効果を述べる。さらに、不揮発性メモリ品を搭載した計算機用のオペレーティングシステムについて、課題を示し、対処への方針を述べる。

キーワード：不揮発性メモリ、オペレーティングシステム、仮想記憶機構、要求時ページング、ページ例外処理

Operating System for Computer Equipped with Only Non-Volatile Memory Block

TANIGUCHI, Hideo[†1]

１．はじめに

不揮発性メモリ（Non-Volatile Memory：以降、NV メモリと略す）は、数十年前から存在し、揮発性メモリと同様にバイトアクセス可能であり、当然のことながら不揮発性（電源 OFF でもデータを保持）である。しかし、アクセス速度は、揮発性メモリに比べ非常に遅い。近年、NV メモリは、アクセス速度の向上、大容量化、消費電力低減、耐久性の向上、および価格低下といった性能や機能の向上が著しい。また、アクセスインタフェースも進み、例えば、Intel が 2019 年 4 月に発表した Optane DC Persistent Memory は、アクセスインタフェースとして DDR4 のデータ信号で利用でき、従来の NV メモリに比べアクセス速度が速い。

NV メモリの特徴を生かし、実メモリが揮発性ではなく不揮発性であれば、データの操作や格納の方法が大きく変わることを想定し、実メモリを NV メモリとして仮想的に扱う研究[1-3]がある。また、ファイルシステムでの有効利用に関する研究、消費電力に着目した研究、およびメモリとして扱う研究がある。さらに、プログラムの実行に関する研究として、文献[4][5][6]がある。文献[4]は、NV メモリをログ構造化ファイルシステムの一部に組込むことで強い一貫性保証と高速化を実現する手法を提案している。文献[5]は、NV メモリを主記憶として利用する際に揮発性メモリを少量だけ混載し、データ構造の配置を工夫することで、インメモリ処理を行う応用プログラムの性能低下を大幅に抑える方法を述べている。文献[6] は、揮発性メモリと NV メモリが混載された計算機において、仮想記憶機構が 2 つのメモリの特長を生かしたプログラムの実行をできるように、新しい実行プログラムのファイル形式を提案している。具体的には、プログラムをメモリ上に配置したときのアクセス形態に着目し、２つのファイルからなる実行ファイル形式（OFF2F：Object File Format consisting of 2 Files）である。文献[4] は、ファイルシステムに大幅な改変を加える必要があるが、OFF2F は、従来のファイルシステムを維持しつつ、ページ例外処理の一部を改変するだけでNV メモリに配置したファイルを実行可能にしている。また、文献[5] は、読み込み遅延の大きい大容量低価格のNV メモリを想定しているが、OFF2F は、読み込み遅延がDRAM と同等で少容量高価格のNV メモリを想定している。

本稿では、実メモリとプロセッサの関係の変化について述べる。次に、実メモリとしてNVメモリを利用することで外部記憶装置が不要であることを述べ、活性挿抜できるNV メモリ（不揮発性メモリ品）を提案し、これにより期待される効果を述べる。さらに、不揮発性メモリ品を搭載した計算機用のオペレーティングシステム（OS）について、課題を示し、対処への方針を述べる。

２．実メモリとプロセッサ

計算機を構成する実メモリとプロセッサの関係に、以下のような変化が生じている。

（１）実メモリとしての NV メモリ

NV メモリは、数十年前から存在し、揮発性メモリと同様にバイトアクセス可能であり、さらに不揮発性である。しかし、アクセス速度が揮発性メモリに比べ非常に遅い。しかし、最近のハードウェア技術の向上により、アクセス速度は向上しており、大容量化とともに低価格化も進んでいる。例えば、Intel が 2019 年 4 月に発表した Optane DC Persistent Memory は、アクセスインタフェースとして DDR4 のデータ信号で利用でき、従来の NV メモリに比べアクセス速度が速い。

†1　岡山大学
Okayama University

（２）メモリキャッシュの重要性
プロセッサの動作速度は非常に速くなっている。これに対し、実メモリが DRAM のような揮発性メモリであっても十分なアクセス性能ではない。このため、メモリキャッシュの役割が非常に重要になっている。メモリキャッシュは、高速化や大容量化とともに高機能化により、高いキャッシュヒット率を生み出している。例えば、命令キャッシュとデータキャッシュの構成がある。また、多くの OS が有する仮想記憶機構では、TLB（Translation Lookaside Buffer）のヒット率がプログラム実行速度に大きく影響する。つまり、現在の計算機では、メモリキャッシュのヒット率が計算機の性能を左右しているといっても過言ではない。逆に言えば、実メモリのアクセス速度の重要度は低下している。以上のことから、アクセス速度について、NV メモリは揮発性メモリと同等であることが必須とは限らない。なお、将来においても、NV メモリは、ハードウェア構造やエネルギー効率の性質上、揮発性メモリに比べ相対的にアクセス速度（特に、書き込み速度）は低速であると考えられる。
（３）実メモリ空間の拡大
64 ビットプロセッサの登場により、プロセッサが扱えるメモリ空間は広がっている。これに伴い、仮想メモリ空間の 64 ビット化だけではなく、実メモリ空間の 64 ビット化も可能である。また、仮想記憶機構では、実メモリ空間と仮想メモリ空間をページ単位で対応付けるため、実メモリ空間が全て連続アドレスである必要はない。
以上のことから、ヒット率が高いメモリキャッシュを有した 64 ビットプロセッサを搭載し、NV メモリを実メモリとする計算機を実現できる。ただし、メモリキャッシュ機能が効果的に動作するには、その容量は実メモリのアクセス速度と容量に大きく影響される。したがって、NV メモリのアクセス速度と容量を考慮したメモリキャッシュの搭載が必要である。

３．不揮発性メモリ品
３．１　外部記憶装置の必要性と代替え
現在の計算機において、不揮発性の記憶媒体である外部記憶装置が必要である要因は、大きく以下の３つである。ここで、情報とは、プログラムとプログラムが扱うデータである。
（１）電源 OFF 時の情報保存：実メモリは、揮発性メモリであるため、電源 OFF とともに実行しているプログラムやデータの情報を消失する。例えば、電源 ON から OFF までの間に、全ての処理（計算）を終えることは難しいため、データの参照と更新を行いながら継続的に処理を進めるには、電源 OFF 時のデータの保存が必要である。もちろん、プログラムの保存も必須である。このため、プログラムやデータを外部記憶装置に格納して保存する必要がある。
（２）多くの情報保存：様々なプログラムが多くのデータを扱う処理を同時に実行しようとすると、実メモリの大きさを超えるプログラムやデータの量を扱える必要がある。つまり、多くのプログラムやデータの保存が必要である。例えば、仮想記憶機構は、プログラムやデータを格納した外部記憶装置を利用し、膨大な実メモリが存在するように制御することで、様々なプログラムの実行を支援している。
（３）情報の可搬：一つの計算機で全ての処理を行うのではなく、プログラムやデータを可搬し、いろいろな計算機で利用できることが望まれる。また、多くの利用者で情報を共有するには、プログラムやデータを分配し各利用者が利用できることが求められる。さらに、情報のバックアップの意味でも、可搬つまり計算機外に情報を取り出せることが必要である。このため、プログラムやデータを可搬できる必要がある。
ここで、ネットワーク利用が常時可能な計算機の場合、ネットワークを介して遠隔にデータを保存することで上記に対応可能である。しかし、非常に高速なネットワーク環境が全ての計算機に必要になってしまう。
上記の外部記憶装置が必要である要因に対し、実メモリを NV メモリにすることで、以下の対処ができ、外部記憶装置を必要としない。
（１）電源 OFF 時の情報保存：NV メモリは不揮発性であるので、電源 OFF 時の情報保存ができる。このため、外部記憶装置は不要である。
（２）多くの情報の保存：NV メモリは、大容量化とともに低価格化も進んでおり、多くの情報保存ができる。さらに、後述する不揮発性メモリ品で、多くの情報の保存を可能にできる。
（３）情報の可搬：NV メモリに格納した情報を可搬にするには、後述する不揮発性メモリ品で可能にできる。
したがって、これらの対処により、外部記憶装置は必要でなくなる。言い換えれば、不揮発性メモリ品を実メモリかつ外部記憶媒体として扱う。なお、情報のバックアップにおいては、その量が膨大になるため、磁気ディスク装置や磁気テープ装置との連携は必要である。

３．２　不揮発性メモリ品の考え方
NV メモリのアクセスインタフェースは進化の途中にある。例えば、Intel が 2019 年 4 月に発表した Optane DC Persistent Memory は、アクセスインタフェースとして DDR4 のデータ信号で利用できる。今後、USB インタフェースのように活性挿抜ができることも考えられる。そこで、NV メモリを実メモリとして搭載し、その一部を活性挿抜できれば、次のような機構が考えられる。
（１）実メモリ空間を適当な大きさに分割する。これを NV メモリ単位（NVM ユニット）と名付ける。例えば、NV ユニットは 1 テラバイト（40 ビット空間）とする。
（２）NVM ユニットでの活性挿抜（装着と脱着）を可能にする。
具体的には、64 ビットプロセッサにおいて、実メモリ空間（64 ビット空間）を 1 テラバイト空間単位で 16 メガ個（24 ビット分）構成とする。また、この活性挿抜できる NVM ユ

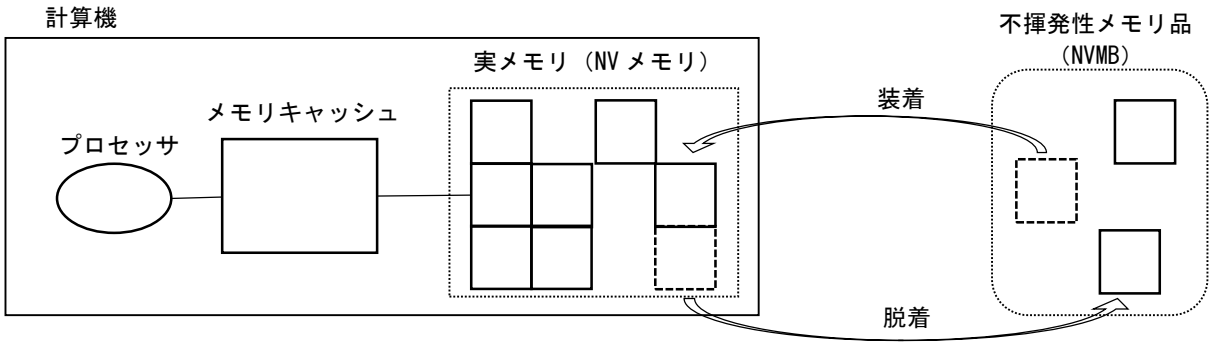

図１　不揮発性メモリ品型計算機

ニットを不揮発性メモリ品（NVMB：Non-Volatile Memory Block）と名付ける。この NVMB の装着と脱着により、実メモリ空間に存在する実メモリ量を自由に増減する。

NVMB の様子を図１に示す。NVMB は、装着により実メモリの一部となり、脱着により実メモリから切り離される。このような計算機を不揮発性メモリ品型計算機（Computer Equipped with Only Non-Volatile Memory Block）、つまり NVMB 型計算機と名付ける。

３．３　期待される効果

NVMB 型計算機に期待される効果として、大きく２つある。

ひとつは、「データの複写」の減少である。古くから、計算機システムにおける OS の本質的なオーバヘッドは「データの複写」といわれている。この「データの複写」には、実メモリ間のデータ複写、および実メモリと外部記憶装置間のデータ入出力がある。NVMB の導入により、以下の示すように「データの複写」は大きく減少する。

（１）実メモリ間のデータ複写は、大きく減少する。例えば、プログラムを実行する際、テキスト部は読み込みのみであるためデータ複写は発生せず、データ部書き込み時のみ発生する。この実メモリ間のデータ複写の減少の程度は、OS のプログラム実行機構に大きく依存する。これについては、4 章で詳しく述べる。

（２）実メモリと外部記憶装置間のデータ入出力は、全く発生しない。

これらにより、「データの複写」の OS オーバヘッドは大きく減少する。

これらの制御よる利用上の効果例を以下に示す。

（例１）データ入出力不要、かつ実メモリ間複写はデータ部のみであるため、高速なプログラム実行が可能になる。

（例２）高速なプログラム実行とともに、環境設定ファイル等の参照でのデータ入力不要であるため、計算機の立ち上げ時間が高速になる。

（例３）ファイル操作の処理において、データ入出力不要であるため、その処理が簡易かつ高速になる。

上記（例３）の具体的事例を述べる。データベース（DB）処理では、処理時に発生する外部記憶装置との入出力を避けるため、データをメモリ上に一括して読み込んで処理を行うこと（インメモリ処理）により、DB 処理を高速化している。しかし、この方法においても、実メモリと外部記憶装置間のデータ入出力は避けられない。これに対し、NVMB 型計算機では、実メモリと外部記憶装置間のデータ入出力は不要であり、メモリ上での処理だけとすることができる。

もうひとつは、電源の ON/OFF に対する即時動作機能を実現できることである。つまり、電源 ON で即時に処理を開始し、電源 OFF で即時に処理を停止できる。NVMB 型計算機では、電源 OFF で揮発する情報はレジスタとメモリキャッシュ上の情報のみである。このため、電源 OFF の感知によりレジスタとメモリキャッシュを NV メモリに書き戻すだけで、現在の状態を保存できる。この書き戻し処理時間は短いため、利用者にとっては即時停止である。また、電源 ON 時の処理開始は、大きく２つに分類できる。ひとつは、動作継続である。レジスタの情報復元を行うだけで、実メモリ（NV メモリ）上のプログラム実行は可能であり、即時に電源 OFF 時の状態から動作を継続開始できる。もうひとつは、初期開始である。レジスタの情報復元を行なわず、OS 立ち上げから開始することで、計算機を開始できる。これらにより、予期しない電源断（つまり電源 OFF）に対し堅牢な計算機を実現できる。また、計算機でも TV（テレビ）のような簡単操作が可能になる。

なお、現在の計算機の動作を一時停止する機能として、スリープ（休止状態）やハイバネーションがある。スリープは動作を停止させ状態を実メモリ（揮発性メモリ）上に保有している。このため、電源供給の停止とともにその内容は失われ、回復できなくなる。しかし、この技術は、NVMB 型計算機で生かされると考える。一方、ハイバネーションは、動作を停止させ状態を外部記憶装置に書き出している。このため、電源供給が停止しても回復可能であるが、実メモリと外部記憶装置間のデータ入出力は避けられないため、即時動作はできない。

４．オペレーティングシステムの課題と対処

４．１　課題

外部憶装置を有しない NVMB 型計算機では、OS が NVMB に

関して以下の課題に対処する必要がある。
（１）データ格納形式：NVMB のボリューム構造やファイルシステム構造といった形式である。現計算機との親和性を重視すると、現在の形式を流用することが考えられる。一方、NV メモリの特徴を生かした新たな形式の考案も必要である。
（２）活性挿抜機能：現在の USB メモリの装着や脱着のような NVMB の活性挿抜機能である。この機能の実現は、OS の実メモリ管理機能であり、ハードウェアが実現する活性挿抜機構に大きく影響を受ける。
（３）プログラム実行法：NVMB 上のプログラムの実行方法とともに、プログラムが参照や更新する NVMB 上のデータ（ファイル相当）へアクセスする方法である。仮想記憶機構を基本とし、仮想メモリ空間を利用したコピーオンライト（CoW：Copy on Write）を駆使する必要がある。
（４）ファイルの扱い：実メモリ空間のファイルを仮想メモリ空間にマッピングする方法、およびファイルシステムのマウントやアンマウントの処理、ファイルの作成や削除といった操作の処理法である。
さらに、NVMB 型計算機は立ち上げのために少量の NV メモリを搭載する。そこで、以下の課題への対処も必要である。
（５）OS 起動法：先に述べた初期開始の方法、および動作継続機能である。

４．２　データ格納形式

NVMBのデータ格納形式においては、ボリューム構造とファイルシステム構造の形式を示す必要がある。

ボリューム構造は、NVユニットの大小により、その必要性が異なる。例えば、NVユニットが１テラバイト（40ビット空間）の場合は複数ボリュームであるが、１ギガバイト（30ビット空間）の場合は単一ボリュームとして扱える。また、劣化や破損によるファイル損傷を考慮すると、NVユニットの大きさを小さくして単一ボリュームの扱いが好ましい。なお、単一ボリュームであっても、当該NVMBの大きさ（メモリサイズ）および下記のブロックサイズを格納した領域（以降、NVボリューム情報と呼ぶ）が必要である。

ファイルシステム構造は、現計算機とNVMB型計算機が共存する社会（環境）を考えると、親和性を重視し現在の形式を流用することが考えられる。一方、NVメモリの特徴を生かした新たな形式も考えられる。そもそも、現在の形式は、以下の考えに基づいて設計されている。
（１）外部記憶装置が磁気ディスク装置であるため、その平均回転待ち時間とシーク時間は各々数ミリ秒である。これに対し、入出力データ転送時間は１ミリ秒以下であることが多い。このため、高い性能を得るには、入出力データ長を短くすることではなく、入出力回数の削減が重要である。そこで、初期のファイルシステムでは、プログラムやデータを連続領域に格納していた。

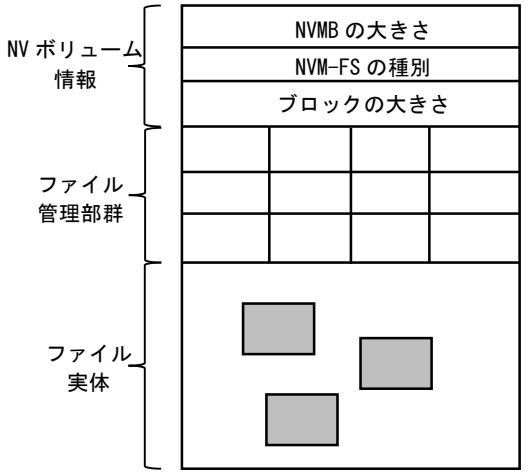

図２　NVMB の構成

（２）しかし、ハードウェアの進歩により、ファイル管理のバッファキャッシュ機能や磁気ディスク装置のキャッシュ機能が高機能化し、平均回転待ち時間とシーク時間を隠蔽できる場合が増加した。また、様々なプログラムが動作するようになり、ファイルの作成削除や大きさの拡縮が頻繁に起こるようになった。これらを受け、ファイルの実体を格納する領域は、固定長分割された領域を用いる現在の形式になっている。

一方、NVMBは、回転待ちやシークが起こらない。したがって、ファイルシステム構造の形式（以降、NVM-FSと略す）は、ファイルの実体を格納する領域を固定長分割された領域（以降、ブロックと略す）とする現在の形式で問題ない。つまり、ファイルの所有者情報などを保持するファイル管理部（Linuxのiノード相当）の集まり（群）とファイルの内容を保持するファイル実体から構成される。NVMBを構成する様子を図２に示す。NVボリューム情報は、NVMBの大きさ、NVMB-FSの種別、およびブロックの大きさについて情報を有する。ファイル管理部群は、固定長のファイル管理部の集まりであり、使用と未使用を管理する領域も有する。ファイル実体は、ファイルの内容を格納するブロック群であり、使用と未使用を管理する領域も有する。

ただし、NVMB型計算機のOSでは、仮想記憶機構が必須である。このため、仮想メモリ空間を構築する際のマッピング表の総量に留意が必要である。NVM-FS上のプログラムやデータを仮想メモリ空間にマッピングするため、マッピングの単位（以降、ページと略す）はブロックの大きさ以下にする必要がある。ここで、両者が同じ大きさと仮定する。このとき、ページ（ブロック）の大きさが大きい場合、マッピング表の総量を小さくできるものの、ブロックの大きさが大きいためNVM-FSの内部断片化が発生する。一方、ページ（ブロック）の大きさが小さい場合、NVMBの内部断片化を抑制できるものの、マッピング表の総量が大きくなる。

４．３　活性挿抜機能

ハードウェアとして NVMB が USB メモリのように活性挿抜できる機能については､ハードウェアの進歩に期待する。なお、利用者の誤操作によるファイルシステム破壊といったシステムへの悪影響を防ぐため、利用者の誤った脱着動作を感知しソフトウェアへ通知する仕組みが好ましい。例えば、NVMB を押すと飛び出す仕組みとし、押された際に脱着動作と感知しソフトウェアへ通知する。

以降では、ハードウェアとして NVMB の活性挿抜が可能であると仮定し、OS の実メモリ管理機能について述べる。

装着処理を以下に述べる。

（１）ハードウェアを監視し、装着されたことを検知し、装着位置（装着された実メモリの先頭アドレス：sraddr）を得る。

（２）装着位置を基に、先頭１ページを仮想メモリ空間にマッピングする。

（３）sraddr と NV ボリューム情報に格納されている NVMB の大きさ情報から、末尾アドレス（eraddr）を算出する。

（４）マッピングを解除する。

（５）実メモリ空間管理として、sraddr から eraddr までを拡張する。

また、脱着処理の内容を以下に記述する。

（１）当該 NVMB のファイルシステムがアンマウントされていることを確認する。これにより、当該 NVMB の実メモリが仮想メモリ空間にマッピングされていない。

（２）実メモリ空間管理として、sraddr から eraddr までを削除する。

（３）脱着可能であることを利用者に通知する。

なお、利用者の誤った脱着動作をハードウェアから通知された場合は、当該 NVMB のファイルシステムがアンマウント状態になるように処理を行う。

４．４　プログラム実行法

４．４．１　実メモリ空間と仮想メモリ空間の関係

NVMBは、装着により実メモリ空間に配置される。実メモリ空間は、計算機が搭載するNVメモリ上に構築されたルートNVM-FS、および装着により実メモリ空間に配置されたNVMB上に構築された各NVM-FSで構成される。仮想メモリ空間の大きさは、実メモリ空間上のファイルを仮想メモリ空間にマッピングしてバイト単位で操作できるために、実メモリ空間の大きさ以上とする。また、仮想メモリ空間は、当該仮想メモリ空間を利用するプロセスが必要とするファイルなどで構成され、基本的に仮想アドレスは実アドレスと同じになるようにマッピングする。

実メモリ空間と仮想メモリ空間の関係として、実メモリ空間の大きさがXYZである場合を図３に示す。各プロセスは、ルートNVM-FSのカーネル、および操作するファイルが格納されたNVMB上のNVM-FSのファイルをマッピングする。プロセスAはNVMB1とNVMB2のファイルをマッピング

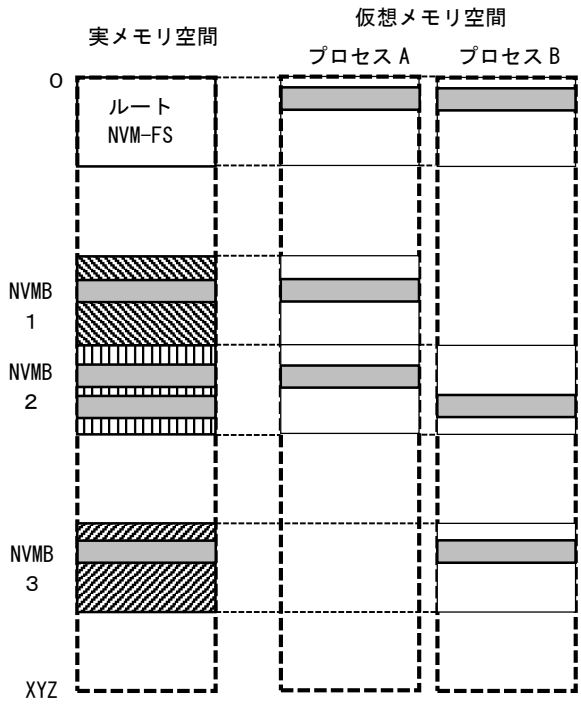

図３　実メモリ空間と仮想メモリ空間の関係

し、プロセスBはNVMB2とNVMB3のファイルをマッピングしている。

４．４．２　プログラムの実行

NV メモリ上のプログラムの実行として、仮想記憶機構を基本とし、仮想メモリ空間を利用した CoW 機能を駆使する方法の基本的考え方を以下に述べる。この方法では、実メモリ（NV メモリ）間のデータ複写を最小限に抑制できる。プロセス生成時にマッピング表を作成し、以下の処理を行う。

（１）参照された実メモリのみを仮想メモリ空間にマッピングする。

（２）更新された実メモリのみを更新可能な実メモリ域に複写し、仮想メモリ空間にマッピングする。

この方法におけるページ例外処理の流れを図４に示し、以下に説明する。下記において、NV メモリは実メモリである。

（１）テキスト部の場合、NV メモリのページをマッピング表に登録する。

（２）データ部の場合、NV メモリのページをマッピング表に登録する。このとき、アクセス権は読み込みのみ設定とする。次に、アクセス権例外（書き込み）によるページ例外が発生すると、ルート NVMB に空ページを確保し、当該ページのデータを複写し、マッピング表を更新する。

（３）BSS 部の場合、ルート NVMB に空ページを確保し、マッピング表に登録する。

（４）データ部と BSS 部については、プロセスの複製が行われた場合、CoW によりルート NVMB の空ページ確保とデータ複写が起こり得る。

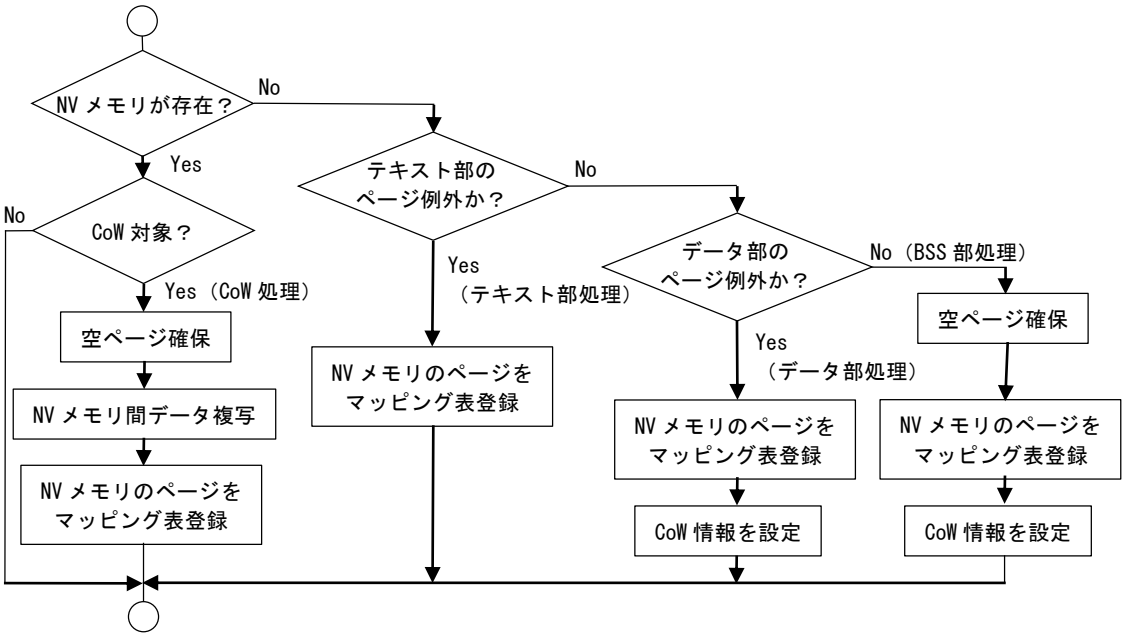

図４　ページ例外処理の流れ

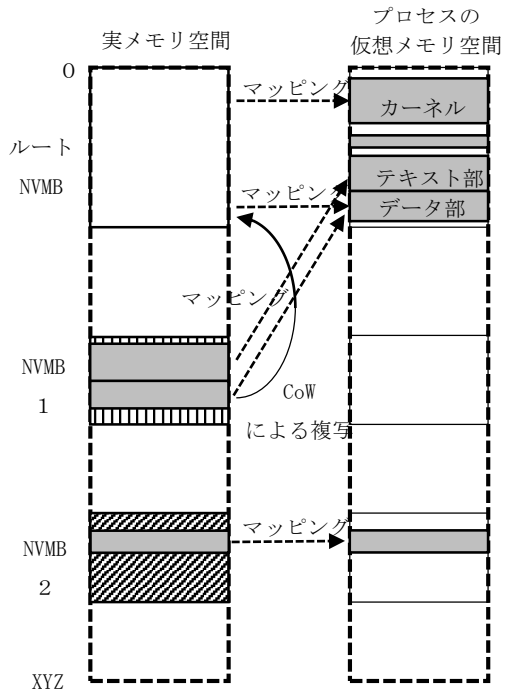

図５　プロセス実行時の各メモリ空間の関係

（５）スタックはルート NVMB に確保する。

したがって、テキスト部については、データ複写はなく、マッピング処理は必要最小限である。また、データ部と BSS 部については、空ページ確保＆データ複写とマッピング処理は必要最小限である。

本方法におけるプロセス実行時の実メモリ空間と仮想メモリ空間の関係を図５に示す。MVNP1 のプログラムの実行が進むと、テキスト部とデータ部は若番の仮想メモリ空間上にマッピングされる。なお、スタックもこの付近に用意する。NVMB2 のファイルのマッピングはプロセスが操作するファイルである。なお、MVNP1 のプログラムのテキスト部とデータ部を NVMB1 の実メモリ空間に相当する仮想メモリ空間上にマッピングする方法も考えられる。しかし、この方法では、NVMB の実メモリ空間の位置が計算期毎に異なる可能性が高いため、プログラムを位置独立コード（PIC：Position-Independent Code）として作成する必要がある。

４．５　ファイルの扱い

４．５．１　ファイルのマッピング

実メモリ空間のファイルは、仮想メモリ空間にマッピングしてバイト単位で操作できる必要がある。そこで、以下の考え方でファイルを仮想メモリ空間にマッピングする。

（１）仮想メモリ空間上のファイルの先頭は、実メモリ空間のファイル実体の先頭ブロックと同じ位置で仮想メモリ空間にマッピングする。

（２）２番目以降のブロックは、先頭ブロックから連続した仮想メモリ空間にマッピングする。

以上により、実メモリ空間のファイルは、ファイル実体の先頭ブロックの先頭実アドレスと同じ仮想アドレスから仮想メモリ空間上の連続領域にマッピングされる。この様子を図６に示す。ファイル実体１（先頭ブロック）は、先頭実アドレスと同じ位置で仮想メモリ空間にマッピングされ、ファイル実体２とファイル実体３のブロックは、ファイル実体１に連続してマッピングされる。このため、プロ

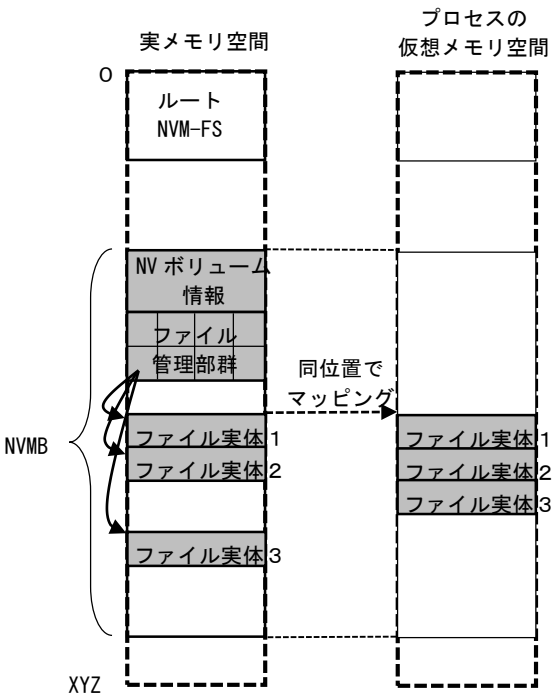

図６　ファイルのマッピング

グラムのファイル操作は、現在の外部記憶装置からメモリへのファイル入力後のメモリ上の操作と同様に行える。

上記を実現するには、各ファイルについて、「仮想メモリ空間内のファイル実体の先頭ブロックの位置が、他ファイルの仮想メモリ空間内に含まれない」ことが求められる。これを実現する方法として、大きく以下の２つある。

（方法１）実メモリ空間において、新規作成ファイルの先頭ブロックの配置は、他ファイルの先頭ブロックからファイルの大きさを基に算出される最終ブロック相当の位置の間に含まれない位置にする方法

（方法２）定期的にコンパクション処理を行い、例えば図６のファイル実体１とファイル実体２のように、ブロックを連続して配置するように変換する方法

もちろん、上記２つの方法を共に利用することも可能である。

４．５．２　ファイルの操作

ファイルシステムのマウントとアンマウントについて、以下に述べる。装着処理時に得た装着位置（sraddr）を用いて処理を行う。

(1)mount(sraddr, mount_point)：実アドレス sraddr で指定される NVMB の NVM-FS について、固定長サイズの NV ボリューム情報からファイル管理部群の先頭実アドレスを算出し、ファイル管理部群を仮想メモリ空間にマッピングする。マッピング位置は、実アドレスと仮想アドレスが同じとする。また、この領域へのアクセス権は、OS のみに設定する。さらに、mount_point で指定される場所へのマウント処理を行う。

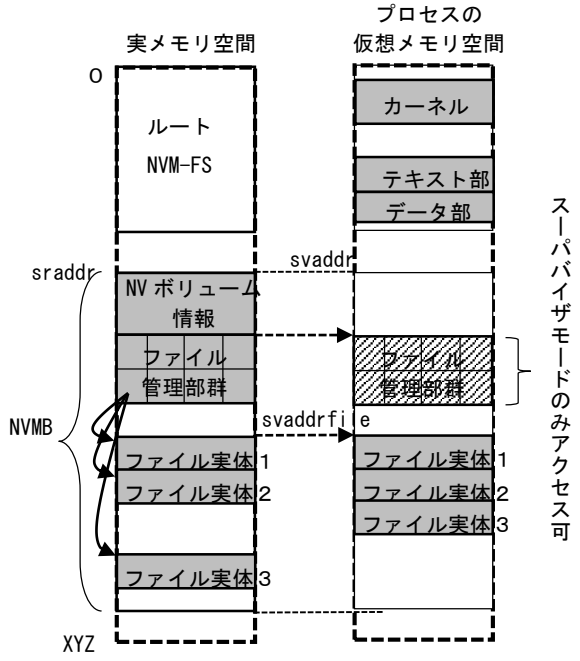

図７　ファイルの操作

(2)umount(svaddr)：仮想アドレス svaddr で指定される NVMB の NVM-FS について、アンマウント処理を行う。また、仮想アドレス svaddr で指定される NVMB の NVM-FS について、ファイル管理部群の先頭仮想アドレスを算出し、ファイル管理部群を仮想メモリ空間からアンマッピングする。ファイルシステムをマウントとした様子を図７に示す。仮想メモリ空間にマッピングされたファイル管理部群は、スーパバイザモードのみでアクセス可能、つまりカーネルのみアクセス可能とする。また、同じ NVMB の場合、sraddr と svaddr は同じ値である。

ファイルの作成と削除について、以下に述べる。

(3)create(filename, tsize, op)：ファイル名 filename のファイルを大きさ tsize で作成する。op では、アクセス権などを指定する。

(4)delete(filename)：ファイル名 filename のファイルを削除する。

ファイルのデータ操作について、以下に述べる。

(5)svaddrfile =open(filename, op):ファイル名 filename のファイルを仮想メモリ空間にマッピングする。op では、アクセス権などを指定する。上記において、マッピング処理には、２つの方法がある。ひとつは、最初はページテーブルを用意するのみとし、ページ例外発生により NV メモリの当該ページをマッピング表に登録する方法である。もう一つは、最初に、ページテーブルを用意し、かつ NV メモリの各ページをマッピング表に登録する方法である。前者の方法は、ファイルの大きさが大きく、操作がファイルの一部分で行われる場合に有効である。一方、後者の方法は、ファイルの大きさが小さく、操作がファイル全体に行われる場合に有効である。

(6)データへの操作:仮想メモリ空間上でバイト単位の操作ができる。
(7)close(svaddrfile):仮想アドレス svaddrfile から始まるファイルを仮想メモリ空間から剥がす。

ファイルの大きさの拡縮について、以下に述べる。
(8)拡大：ファイルの大きさ（tsize）を超えるアドレスへの書き込みは、ページ例外を利用してファイルの大きさを拡大する。
(9)縮小：ファイルサイズ縮小は、tsize=shrink(vaddr)による。仮想アドレス vaddr をファイルの末尾とする。

４．６　OS 起動法

電源 ON による OS の起動法は、3.3 節に記したように大きく２つに分類できる。動作継続と初期開始である。ここでは、初期開始について述べる。

初期開始は、現在の OS 起動手順である boot プログラム実行から OS（カーネル）プログラム実行までを踏襲して実現できる。ただし、boot やカーネルのプログラムはルート NVMB 上にあるため、仮想メモリ空間にマッピングするだけで実行できる。例えば、以下の手順で OS の起動ができる。
（１）リアルモードで boot プログラムの実行を開始する。
（２）boot プログラムは、仮想メモリ空間を作成し、boot プログラムとカーネルプログラムを仮想メモリ空間にマッピングし、プロテクトモードに移行し、カーネルプログラムの実行を開始する。
（３）カーネルは、OS の初期化処理を行う。

５．おわりに

外部記憶装置を有せず実メモリとして不揮発性メモリのみを搭載した計算機、つまり不揮発性メモリ品（NVMB）型計算機を提案し、オペレーティングシステムに求められる機能を述べた。

具体的には、実メモリとプロセッサの関係の変化について述べ、ヒット率が高いメモリキャッシュを有した 64 ビットプロセッサを搭載し、不揮発性メモリを実メモリとする計算機の実現を提案した。次に、実メモリとして不揮発性メモリを利用することで外部記憶装置が不要であることを述べ、活性挿抜できる不揮発性メモリ品を提案し、これにより期待される効果として「データ複写の削減」と「即時動作機能の実現」を述べた。さらに、不揮発性メモリ品を搭載した計算機用のオペレーティングシステムについて、５つの課題（データ格納形式、活性挿抜機能、プログラム実行法、ファイルの扱い、OS 起動法）を述べ、対処への方針を述べた。

残された課題として、OS の各課題の詳細検討、および動作継続機能の検討がある。

[謝辞]本原稿の執筆にあたり、ご協力いただいた富士通研究所の中島耕太氏に感謝します。なお、本研究の一部は、JSPS KAKENHI 18K11244、および共同研究（株式会社富士通研究所）による。

＜参考文献＞

[1] 谷口秀夫, “分散指向永続オペレーティングシステム ***Tender***”, 情報処理学会コンピュータシステムシンポジウム, シンポジウム論文集, Vol.95, No.7, pp.47-54 (1995).
[2] 谷口秀夫, 市川正也, “***Tender*** オペレーティングシステムにおける資源の永続化機構、” 情処研報、Vol.00,No.32,pp.7-12（1999）.
[3] Toshihiro Yamauchi, Yuta Yamamoto, Kengo Nagai, Tsukasa Matono, Shinji Inamoto, Masaya Ichikawa, Masataka Goto, Hideo Taniguchi, “Plate: Persistent Memory Management for Nonvolatile Main Memory,” Proceedings of 31st ACM Symposium on Applied Computing (SAC 2016), pp.1885-1892 ACM (2016.04).
[4]Xu, J. and Swanson, S. , “NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories,” *14th USENIX Conference on File and Storage Technologies* (*FAST 16*), pp.323 - 338, USENIX Association (2016) (online), available from _https://www.usenix.org/conference/fast16/technical-sessions/presentation/xu_.
[5] Dulloor, S.R., Roy, A., Zhao, Z., Sundaram, N., Satish, N., Sankaran, R., Jackson, J. and Schwan, K., “Data Tiering in Heterogeneous Memory Systems,” *Proc. Eleventh European Conference on Computer Systems*, *EuroSys ’16*, pp.15:1-15:16, ACM (online), DOI: 10.1145/2901318.2901344 (2016).
[6] Sato, M. and Taniguchi, H., “OFF2F: A New Object File Format for Virtual Memory Systems to Support Volatile/Non-volatile Memory-Mixed Environment, *International Journal of Machine Learning and Computing*, Vol.9, No.4, pp.387-392 (2019).