揮発/不揮発メモリ混載環境を支援する仮想記憶機構向け実行ファイル形式：OFF2F の提案

谷口秀夫[†1]

概要：近年、電源 OFF でも保存データが消失しない不揮発性メモリの研究が進んでいる。まだ実用計算機としては登場していないものの、まもなく登場が期待できる。そこで、本稿では、揮発性メモリと不揮発性メモリが混載された計算機において、仮想記憶機構が二つのメモリの特徴を生かしたプログラム実行の支援を可能にするために、新しい実行プログラムのファイル形式を提案する。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、二つのファイルから成る実行ファイル形式（OFF2F: Object File Format consisting of 2 Files）を提案する。また、評価として、ページ例外処理の時間短縮効果を示す。

キーワード：不揮発性メモリ、実行ファイル形式、仮想記憶機構、要求時ページング、ページ例外処理

OFF2F: A New Object File Format for Virtual Storage Mechanism to Support Mixed Volatile/Nonvolatile Memory Environment

TANIGUCHI, Hideo[†1]

１．はじめに

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、メモリを不揮発性メモリ（Non-Volatile Memory：以降、NV メモリと略す）として仮想的に扱う研究[1-3]がある。

一方、最近のハードウェア技術の進歩により、不揮発性メモリが登場している。この登場を受け、不揮発性メモリを有効利用するソフトウェア技術も研究を開始している。ファイルシステムでの有効利用に関する研究として、揮発性の実メモリと外部記憶装置の利用に合わせて NV メモリの利用形態を変更させる方法[4]、NV メモリと外部記憶装置を組み合わせて NV メモリの容量を仮想拡張する手法 VEMS[5]、ファイルシステムのメタデータに NV メモリを利用する研究[6]がある。また、データの書き出しに NV メモリをキャッシュとして利用する研究[7]がある。消費電力に着目し、HPC において NV メモリと揮発性メモリの搭載比率が性能と省電力に与える影響に関する研究[8]がある。メモリとして扱う研究として、NV メモリをメモリ階層のひとつとして利用する研究[9]がある。さらに、不揮発性メモリも含め様々な特徴を持つメモリを言語記述として見せる研究[10]がある。

本稿では、揮発性メモリと不揮発性メモリが混載された計算機において、仮想記憶機構が二つのメモリの特徴を生かしたプログラム実行の支援を可能にするために、新しい実行プログラムのファイル形式を提案する。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、二つのファイルから成る実行ファイル形式（OFF2F: Object File Format consisting of 2 Files）を提案する。また、評価として、ページ例外処理の時間短縮効果を示す。

†1　岡山大学
Okayama University

２．揮発/不揮発メモリ混載環境

従来、計算機の実メモリは揮発性メモリであり、電源 OFF と共に保持データは消失する。これに対し、電源 OFF 後においてもデータを継続保持できる NV メモリが登場している。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。このため、NV メモリの不揮発性という特徴を生かし、例えば、NV メモリにファイルシステムを構築して PDA で利用されている。

揮発性メモリは、アクセス速度が高速であり、これまでの技術革新により大容量かつ低価格である。これに対し、ハードウェア構造やエネルギー効率の性質上、相対的に見ると、NV メモリは、アクセス速度（特に、書き込み速度）が低速であり、少容量かつ高価格である。したがって、実メモリがすべて NV メモリ搭載になる構成ではなく、揮発性メモリと NV メモリが共存する実メモリ構成のプロセッサ環境が、今後の主流になる。

一方、不揮発性という共通の特徴を持つ外部記憶装置と NV メモリを比較すると、アクセス単位の面で大きな違いがある。外部記憶装置はブロック単位であるのに対し、NV メモリはバイト単位である。また、磁気ディスク装置（DK）やソリッドステートドライブ（SSD）といった外部記憶装置は、大容量、低価格および高耐久性である。

したがって、今後の計算機として、システム構成のイメージを図 1 に示す。NV メモリは、揮発性メモリと同様に実メモリを構成する。なお、揮発性メモリと NV メモリがアドレス連続に配置されている必要はない。外部記憶装置は、バス接続され、入出力装置の位置づけである。この構成を生かす例として、NV メモリ上にファイルシステム（NVM-FS）を構築し、外部記憶装置上の通常ファイルシステム（nFS）と連携することが有効[11]である。

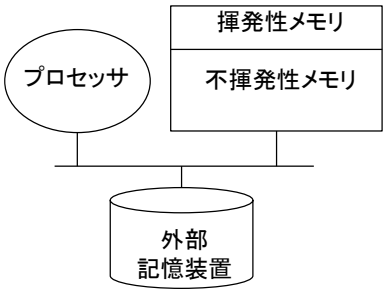


図1 システム構成（イメージ）

３．実行ファイルの形式

３．１　形式

実行できるプログラムの形式は、コンパイラと OS の連携で規定されている。「OS の要求に合わせてコンパイラが出力している」もしくは「コンパイラが出力した形式を OS がサポートする」のどちらでも構わないが、その連携は深い。いずれにせよ、実行できるプログラムは、以下の内容から構成されている。

・テキスト部：手続き（命令）の羅列
・データ部：初期値を持つデータの格納域
・関係情報部：外部変数に関する情報

これらの情報をファイル形式として格納するために、

・ヘッダ部：上記情報の格納位置などを保持

がある。実行ファイル形式の例を図２に示す。

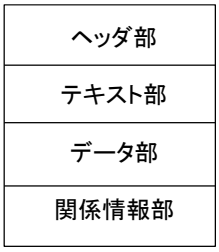


図2 実行ファイル形式の例

既存の代表的な実行ファイル形式には、UNIX の a.out 形式や COFF（Common Object File Format）がある。もちろん、いずれの形式も１つのファイルに格納されている。

３．２　プログラム実行時の問題

既存の実行ファイルは１ファイルに格納されているため、仮想記憶機構の要求時ページング（ODP）機能を利用したプログラム実行時には、次の問題がある。

（１）外部記憶装置（例えば DK）上のファイルとして格納した場合、DK から揮発性メモリへのページ読み込み時間（ページイン処理時間）が長い。また、ページアウト処理時間も長い。外部記憶装置として DK ではなく SSD を利用することも可能になっているが、メモリ間複写に比べるとその入出力時間は長い。

（２）NV メモリ上のファイルシステムとして格納した場合、NV メモリから揮発性メモリへのメモリ間複写により、ページイン処理時間は非常に短い。しかし、全ての実行ファイルを NV メモリ上にファイルとして格納する必要があるため、大きな NV メモリが必要になり、高価になってしまう。

４．新しい実行ファイル形式：OFF2F

４．１　考え方

揮発性メモリは、読み書き共に高速である。一方、NV メモリは、揮発性メモリと同様にバイト単位アクセスが可能であり、読み出しは高速である。しかし、書き込みは低速である。したがって、読み出しのみ行われるデータを NV メモリに格納し、仮想記憶を利用してそのまま仮想空間にマッピングして利用できれば、ODP 処理の時間を短縮できる。

実行ファイルは、４つの内容（テキスト部、データ部、関係情報部、ヘッダ部）から構成されている。これらの部分は、アクセス形態から大きく２つに分類できる。ひとつは、読み出し（リード）のみ行われる部分であり、テキスト部、関係情報部、およびヘッダ部である。なお、ヘッダ部の読み出しは、主にプログラムをプロセスとして実行する時に発生し、この時の関係情報部への読み出しは発生しない。一方、テキスト部への読み出しは、プログラム実行により頻繁に発生する。もうひとつは、頻繁に読み書き（リード&ライト）される部分であり、データ部である。

そこで、複数のファイルから成る実行ファイル形式を提案する。実行ファイルは４つの内容（テキスト部、データ部、関係情報部、ヘッダ部）から構成されているため、最大４ファイルに分割格納する案がある。しかし、構造の複雑化を防ぎ、またファイルシステムでの領域占有量を抑制するため、構成するファイル数を最小限とする。つまり、２ファイルに分割格納する実行ファイル形式（OFF2F: Object File Format consisting of 2 Files）を提案する。

４．２　形式と比較

OFF2F の形式は、実行ファイルが４つの内容から構成されていることから多くの組み合わせが考えられるが、アクセス形態の特徴を考慮すると、大きく２つある。この様子を図３に示す。

タイプ A は、ヘッダ部とデータ部および関係情報部をひとつのファイルとし、別ファイルであるテキスト部への情報を持つ。つまり、プログラム実行により頻繁に読み出しが発生するテキスト部のみを別ファイルとする形式である。タイプ B は、ヘッダ部とテキスト部および関係情報部をひとつのファイルとし、別ファイルであるデータ部への情報を持つ。つまり、プログラム実行により頻繁に読み書きが発生するデータ部のみを別ファイルとする形式である。タイプ A はファイル XYZ を NV メモリ上に置き、タイプ B

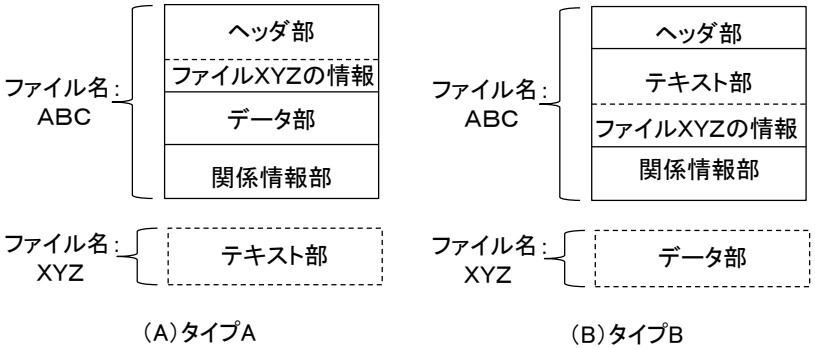


図3 OFF2F形式

はファイル ABC を NV メモリ上に置く。したがって、両者を比較すると、以下の理由によりタイプ A が好ましい。

（1）実行ファイルの名前はヘッダ部を保持する ABC であり、外部記憶装置上に存在することにより、既存の処理流れを利用できる。詳細は、後節で述べる。

（2）ファイル XYZ は、必ずしも NV メモリ上に存在する必要はないため、NV メモリの有無の影響を受けない。

5．OFF2F を利用する仮想記憶機構

5．1　利用法

仮想記憶機構における OFF2F の利用について、図 3（A）に示したタイプ A を例に、以下に述べる。ファイル ABC は外部記憶装置上に存在し、ファイル XYZ は NV メモリ上に存在する。ファイル ABC を利用してプロセスのテキスト部とデータ部を仮想メモリ空間に用意する処理流れを図 4 に示す。

（1）外部記憶装置上に存在するファイル ABC のヘッダ部を読み込む

（2）ヘッダ部の情報より、テキスト部が NV メモリ上に存在することを認識する

（3）NV メモリ上のテキスト部の各ページをマッピング表に登録する

（4）ヘッダ部の情報より、データ部が外部記憶装置上に存在することを認識する

（5）実メモリを確保し、外部記憶装置上に存在するデータ部を読み込む

（6）実メモリの各ページをマッピング表に登録する

したがって、3.2 節で述べた問題に対し、処理（3）により、テキスト部について、外部記憶装置上に存在する場合に比べ入出力を削減できる。また、NV メモリ上に存在する場合に比べメモリ間複写を削減でき、大きな NV メモリを必要としない。

また、ODP 機能を有する場合、プロセス生成時には処理（3）と（6）はページ例外フラグの設定処理となり、処理（5）は行わない。

5．2　ページ例外処理

ODP 機能を有する場合のページ例外処理について、図 5 に示す。

図 5 に示すように、NV メモリ上のテキスト部に対するページ例外の場合には、NV メモリの当該ページをマッピ

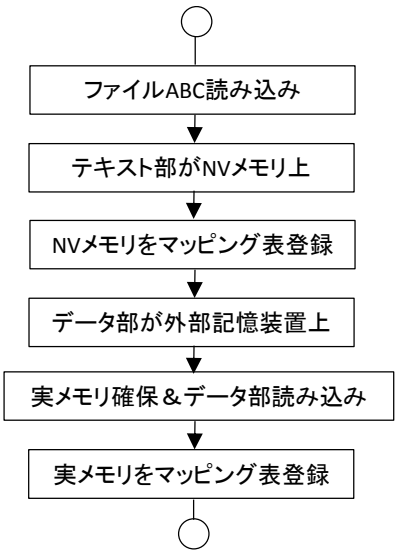


図4 仮想メモリ空間の作成処理流れ

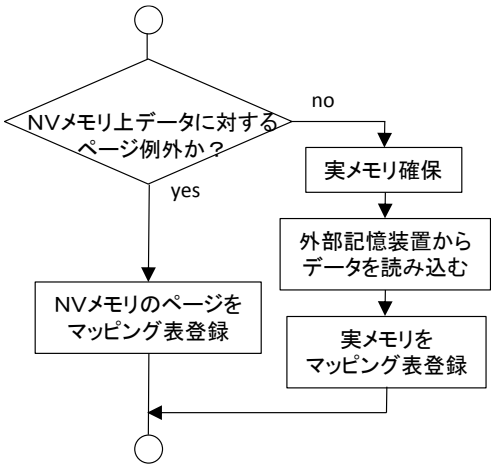


図5 ページ例外処理の流れ

ング表に登録するだけでよい。このため、ページイン処理やページアウト処理で発生する入出力は不要であり、実メモリ確保処理やメモリ間複写処理も割愛できる。

５．３　ページ例外処理時間の定式化

図５に示したページ例外処理の流れに基づき、以下の２つの場合について、ページ例外処理の時間を定式化する。

（従来）プログラムが全て外部記憶装置上に格納されている場合

（提案）OFF2F を利用して、テキスト部は NV メモリ上、データ部は外部記憶装置上に格納され、提案の処理（NV メモリをマッピング表に登録）を行う場合

以下のように各処理の時間を定義する。

・t1：実メモリ確保処理

・t2：外部記憶装置からデータ（１ページ：４KB）を読み込む処理

・t3：実メモリをマッピング表に登録する処理

・t4：NV メモリのページをマッピング表に登録する処理

さらに、

・P：プログラム（テキスト部＋データ部）の大きさ

・S：プログラムに占めるテキスト部の割合

とすると、各場合について、プログラム全体のページ例外（PF）処理時間の和、つまり、テキスト部とデータ部の全ページに一度だけページ例外が発生した時間の総和は、下式となる。

（従来）　(t1+t2+t3)P　　（１）

（提案）　(t1+t2+t3)P(1-S)+t4PS　　（２）

６．評価

６．１　観点

ODP 機能で OFF2F を利用することにより、ページ例外処理の時間を大きく短縮することが期待できる。ここでは、テキスト部とデータ部の大きさの割合に着目し、ページ例外処理の時間の定式を用い、基本性能と FreeBSD での性能予測を述べ、OFF2F の有効性を示す。

６．２　基本性能

以下の環境で、1 GB データについて 4 KB 単位のランダム読み込み時間を測定した。

・OS：FreeBSD 6.3-R

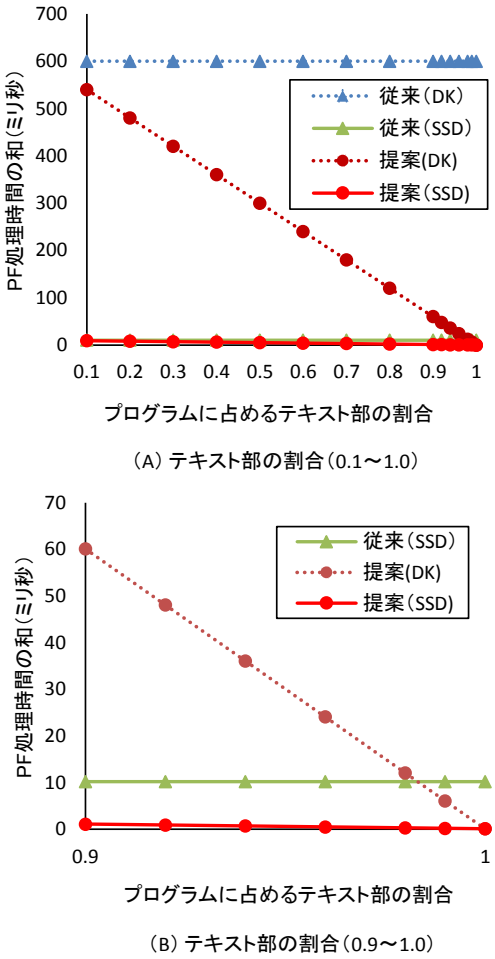
・プロセッサ：Intel® Core™ i7-2600 (3.4GHz)

・DK：Seagate ST500DM002(7200rpm)

・SSD：Intel SSD 540s Series

その結果、平均読み込み時間は、DK の場合 6.22 ミリ秒、SSD の場合 93.70 マイクロ秒であった。

そこで、外部記憶装置からデータ（１ページ：４KB）を読み込む処理時間（t2）として、DK は 6 ミリ秒、SDD は 0.1 ミリ秒と仮定する。5.3 節の式（１）と（２）を用い、プログラム全体のページ例外（PF）処理時間の和を算出し、その結果を図６に示す。横軸は、プログラムに占めるテキスト部の割合である。なお、プログラムの大きさ（P）は 100 ページとし、t2 以外の各処理時間は 0.001 ミリ秒（１μ秒）とした。図６より、以下のことがわかる。



（A）テキスト部の割合（0.1～1.0）

（B）テキスト部の割合（0.9～1.0）

図６　プログラム全体のPF処理時間の和

（t2(DK):6ミリ秒, t2(SSD):0.1ミリ秒, 他は0.001ミリ秒）

（１）（A）より、（従来）は、テキスト部とデータ部共に PF 処理の内容が同じであるため、プログラムに占めるテキスト部の割合に関係なく、PF 処理時間は一定である。なお、DK と SSD の差は、読み込み時間が大きく異なる（DK は SSD の 60 倍）ことに起因する。このことから、DK ではなく SSD を利用することにより、従来手法でも PF 処理時間を大きく短縮できるといえる。

（２）（A）より、（提案）は、プログラムに占めるテキスト部の割合が増加するにつれて、外部記憶装置からのデータ読み込み回数が削減するため、PF 処理時間は減少する。

（３）(B)より、プログラムに占めるテキスト部の割合が増加すると、提案（DK）の PF 処理時間は従来（SSD）より短くなる。したがって、プログラムに占めるテキスト部の割合が多い（具体的には、約 98.3%超）場合は、DK を SSD に変更するより、提案手法を利用したほうが PF 処理時間を短縮できるといえる。

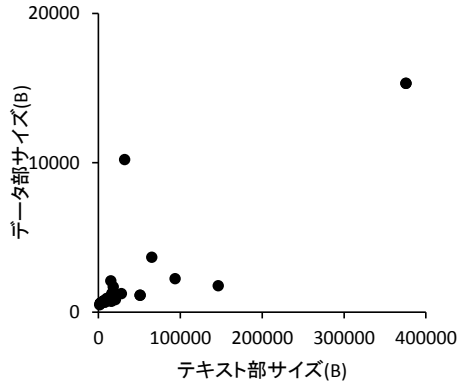


図7　FreeBSD(Ver.11) /bin下の各プログラムの大きさ

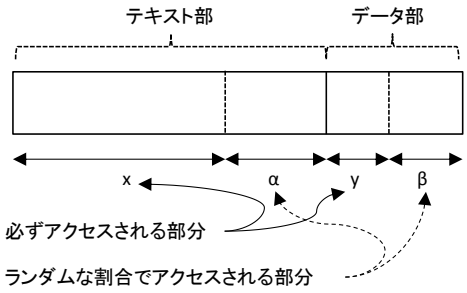


図8　プログラム実行時のアクセスの様子

６．３　FreeBSD での効果予測

６．３．１　実行プログラムの分析

FreeBSD(Ver.11)の実行プログラムとして、/bin と/sbin の下のファイルを分析した。分析の結果、/bin の下のファイル群と/sbin の下のファイル群は同様な性質であったため、以降では、/bin の下のファイル群について述べる。/bin 下の各ファイルの実行プログラムを図７に示す。図７に示すように、多くのプログラムにおいて、テキスト部サイズはデータ部サイズの約 20 倍である。全プログラムについても、/bin 下には、44 個のファイルがあり、テキスト部の総和は 1,518,784 バイト、データ部の総和は 80,136 バイトである。したがって、テキスト部はデータ部の約 20 倍の大きさである。

６．３．２　比較

/bin 下のプログラム群について、全てのプログラムが１回プロセスとして起動され、動作する際のページ例外処理時間の和を算出する。これにより、従来法と提案法のページ例外処理時間を比較する。

各プログラムの実行時のアクセス（読み書き）の様子を図８に示すように仮定する。具体的には、以下を定義する。

・x：テキスト部で必ずアクセスされる割合

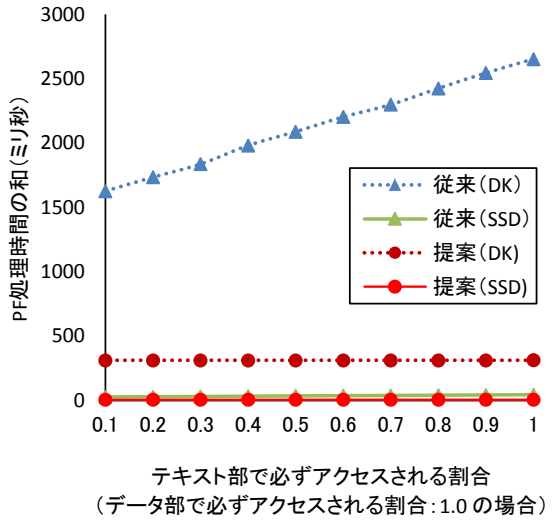
・y：データ部で必ずアクセスされる割合

さらに、各プログラムは、以下を仮定する。

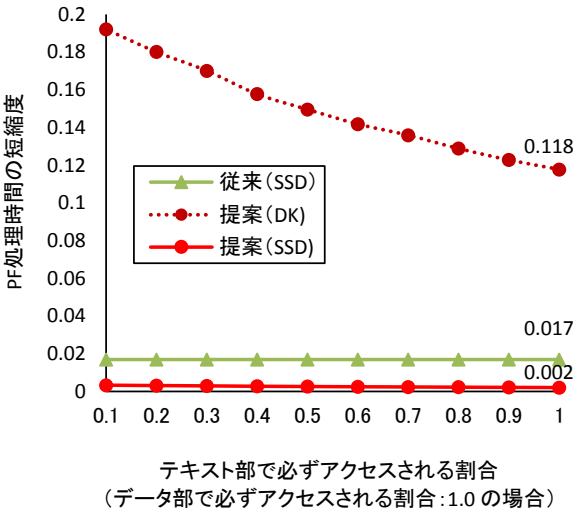
・テキスト部の（1-x）の割合の部分は、ランダムな割合 α でアクセスされる。

・データ部の（1-y）の割合の部分は、ランダムな割合 β でアクセスされる。

これらにより、プログラム i をプロセスとして起動し実行した際、ページ例外の発生回数は以下の式の和になる。



(A) F処理時間の和



(B) PF処理時間の短縮度（従来（DK)を 1.0 とした場合）

図９　FreeBSD(Ver.11) /bin下のプログラム群におけるPF処理時間の和

・テキスト部のページ例外回数

（Ti・x +Ti(1-x)・α）／（ページサイズ）+1　（３）

・データ部のページ例外回数

（Di・y +Di(1-y)・β）／（ページサイズ）+1　（４）

ここで、Ti と Di は、プログラム i のテキスト部サイズおよびデータ部サイズである。

上記の式（３）と（４）を/bin 下の各プログラムに適用し、かつ 5.3 節の式（１）と（２）を用いて、従来法と提案法のページ例外処理時間の和を算出した。/bin 下のプログラム群の和を図 9 に示す。（A）はページ例外処理時間の和であり、（B）は従来（DK）を 1 としたときの相対時間（つまりページ例外処理時間の短縮度）である。なお、t1,t2,t3,t4 は、6.2 節と同様（t2(DK):6 ミリ秒,t2(SSD):0.1 ミリ秒,他は 0.001 ミリ秒）とした。また、データ部で必ずアクセスされる割合（y）は、1.0 の場合である。なお、0.5 であっても PF 処理時間の和は同様であった。これは、次の原因による。一つは、テキスト部サイズに比べデータ部サイズが非常に小さいことである。もう一つは、データ部サイズが小さいため、ページを単位とするサイズ数は大半が 1 ページになってしまうためである。具体的には、データ部のアクセスされるページ数の和が、y=1.0～0.6 のとき 52 ページ、y=0.5～0.1 のとき 51 ページであった。

図 9 より、以下のことがわかる。

（１）（A）より、テキスト部で必ずアクセスされる割合が増加することは、ページ例外回数が増加することに等しいため、いずれの場合もページ例外処理時間の和は増加する。しかし、従来（DK）の増加は著しく、他の場合は少ない。したがって、（B）より、テキスト部で必ずアクセスされる割合が増加するにつれて、他の場合、特に提案（DK）の場合は短縮度が大きくなる。

（２）（B）より、ページ例外処理は、従来（DK）に比べ、従来（SSD）は約 60 倍、提案（DK）は約 8 倍、提案（SSD）は約 500 倍に高速化できる。したがって、DK あるいは SSD に関係なく、提案法は従来法に比べ約 8 倍高速化できる。

７．おわりに

揮発性メモリと不揮発性メモリが混載された計算機において、仮想記憶機構に基づくプログラム実行を高速化するために、新しい実行プログラムのファイル形式（OFF2F: Object File Format consisting of 2 Files）を提案した。また、OFF2F を利用したページ例外処理の流れを示した。

評価として、テキスト部とデータ部の大きさの割合に着目し、ページ例外処理の時間を定式化した。FreeBSD(Ver.11)の/bin 下の実行プログラムでは、外部記憶装置が DK あるいは SSD に関係なく、提案法は従来法に比べページ例外処理の時間を約 8 倍高速化できることを明らかにした。

残された課題として、提案手法の実装による評価がある。

＜謝辞＞

本研究の一部は、株式会社富士通研究所との共同研究による。

＜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] 追川修一：Non-Volatile メインメモリとファイルシステムの融合，情報処理学会論文誌,54(3),pp.1153-1164 (2013).

[5] 追川修一:ブロックストレージとの組み合わせによるメモリストレージ容量拡張手法，情報処理学会論文誌コンピューティングシステム,8(2),pp.15-24(2015).

[6] Qingsong Wei, Jianxi Chen, Cheng Chen, “Accelerating File System Metadata Access with Byte-Addressable Nonvolatile Memory,” July 2015 ACM Transactions on Storage (TOS): Volume 11 Issue 3, July 2015.

[7] Eunji Lee, Julie Kim, Hyokyung Bahn, Sunjin Lee, Sam H. Noh, “Reducing Write Amplification of Flash Storage through Cooperative Data Management with NVM,” May 2017 ACM Transactions on Storage (TOS) - Special Issue on MSST 2016 and Regular Papers: Volume 13 Issue 2, June 2017.

[8] Matthew Poremba, Itir Akgun, Jieming Yin, Onur Kayiran, Yuan Xie, Gabriel H. Loh, “There and Back Again: Optimizing the Interconnect in Networks of Memory Cubes,” June 2017 ISCA '17: Proceedings of the 44th Annual International Symposium on Computer Architecture.

[9] Doe Hyun Yoon, Tobin Gonzalez, Parthasarathy Ranganathan, Robert S. Schreiber, “Exploring latency-power tradeoffs in deep nonvolatile memory hierarchies,” May 2012 CF '12: Proceedings of the 9th conference on Computing Frontiers.

[10] Xiaochen Guo, Aviral Shrivastava, Michael Spear, Gang Tan, “Languages Must Expose Memory Heterogeneity,” October 2016 MEMSYS '16: Proceedings of the Second International Symposium on Memory Systems.

[11] 谷口秀夫, “不揮発性メモリと外部記憶装置を利用したファイルシステム,” 第 16 回情報科学技術フォーラム（FIT2017）,第 1 分冊, pp.179-180, (2017).