Vol.2019-OS-146 No.12
2019/5/31

擬似不揮発性メモリを用いた OFF2F プログラムの実行方式

高杉 頌[1]　佐藤 将也[1]　谷口 秀夫[1]

概要：不揮発性メモリのアクセス速度が速くなっている．そこで，揮発性メモリと不揮発性メモリが混載された計算機を想定し，プログラム実行を高速化する手法として，新たな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．OFF2F は，プログラムをメモリ上で実行する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である．本稿では，混載環境として，揮発性メモリのみを搭載した計算機で不揮発性メモリを擬似的に実現する手法を述べる．また，この擬似不揮発性メモリを用いて，OFF2F プログラムを実行する方式を述べる．

1. はじめに

従来，計算機の実メモリは揮発性メモリであり，シャットダウンと共に保持している内容は失われる．これに対し，シャットダウン時にもデータを継続して保持できる不揮発性メモリ（Non-Volatile Memory，以降，NV メモリ）が登場している．NV メモリを有効利用するソフトウェア技術の研究として，NV メモリにメタデータを格納する FSMAC [1] や，高速な NV メモリと低速な SSD の 2 つを階層型のストレージとして扱う ATSMF [2] がある．また，NV メモリをキャッシュ階層として扱うことで平均メモリアクセス時間を改善する研究 [3]，CPU のキャッシュと NV メモリ上のデータの一貫性を最適化する研究 [4]，NV メモリを利用したプログラミング言語の拡張の研究 [5] がある．今後の技術革新により，NV メモリは，揮発性メモリ並みの読み出し速度が見込まれるものの，書き込み速度が揮発性メモリと同等になることは困難である．また，NV メモリは少容量かつ高価格である．このため，実メモリが全て NV メモリ搭載になる構成ではなく，揮発性メモリと NV メモリが混載する実メモリ構成のプロセッサ環境（以降，混載環境）が主流になる．

文献 [6] では，混載環境を対象に，新たな実行ファイルの形式（OFF2F: Object File Format consisting of 2 Files）が提案されている．この形式では，仮想記憶機構が 2 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，実行ファイルを 2 つのファイルから構成している．具体的には，テキスト部を 1 つのファイルとし，テキスト部以外を別のファイルにする．NV メモリのアクセス速度の特徴を利用して，OFF2F を用いて読み出しのみ行われるテキスト部を NV メモリ上に格納し，仮想記憶を利用してそのまま仮想記憶にマッピングすることで，ページ例外処理時間を短縮できる．文献 [7] [8] では，FreeBSD 11.0-RELEASE（以降，FreeBSD）の OS 初期化処理に OFF2F を適用した際の効果予測が示されている．また，文献 [9] [10] では，CoW（Copy on Write）機能を考慮した OFF2F プログラムのページ例外の評価が行われている．しかし，現在のところ混載環境の計算機は身近に存在せず，OFF2F の効果を検証できない．

そこで，本稿では，OFF2F の効果を検証する支援環境として，揮発性メモリのみで構成される既存計算機を用いて，擬似 NV メモリを実現する方式を述べる．具体的には，搭載されている揮発性メモリの一部を利用して擬似 NV メモリを構築する．また，擬似 NV メモリにファイルシステムを構築し，OFF2F プログラムのテキスト部を配置する方法を示す．さらに，FreeBSD において，仮想メモリ空間の作成処理流れとページ例外処理流れを変更することで，擬似不揮発性メモリを用いて OFF2F プログラムを実行する方式を述べる．

2. OFF2F

2.1 従来の実行ファイル形式の問題点

実行ファイル形式の例を**図 1** に示す．既存の実行ファイル形式は，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領域

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

図 1 実行ファイル形式の例

- 関係情報部：外部変数に関する情報
- ヘッダ部：上記情報の格納位置などを保持

FreeBSD における既存の代表的な実行ファイル形式には，a.out 形式，COFF（Common Object File Format），および ELF（Executable and Linkable Format）などがある．いずれの形式においても，上記の 4 つの内容は 1 つのファイルに格納されている．このため，仮想記憶機構の要求時ページング（On Demand Paging，以降，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）実行ファイルをディスク装置（以降，DK）などの外部記憶装置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出し時間（ページイン処理時間）が長い．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である．しかし，SSD はメモリ間複写に比べるとその入出力時間は長い．

（2）実行ファイルを 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 メモリが必要となり，高価になってしまう．

2.2 考え方

揮発性メモリには以下の特徴がある．

- バイト単位アクセスが可能
- 読み書きともに高速

一方，NV メモリは揮発性メモリには以下の特徴がある．

- バイト単位アクセスが可能
- 読み出しは高速
- 書き込みは低速

したがって，読み出しのみが行われるデータをそのまま仮想メモリ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（ヘッダ部，テキスト部，データ部，および関係情報部）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．1 つは，読み出しのみが行われる部分であり，ヘッダ部，テキスト部，および関係情報部である．なお，ヘッダ部の読み出しは，主にプログラムを実行するときに発生する．このとき，関係情報部の読み出しは発生しない．一方，テキスト部の読み出しは，プログラム実行により頻発する．もう 1 つは，頻繁に読み書きされるデータ部である．

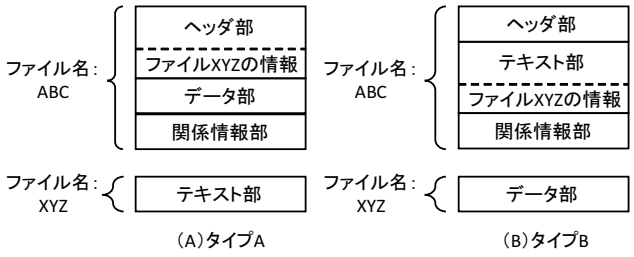

図 2 OFF2F の形式

そこで，実行ファイルの内容のアクセス形態に着目し，複数のファイルからなる実行ファイル形式（OFF2F）が提案された [6]．実行ファイルは 4 つの内容から構成されているため，最大 4 つのファイルに分割し格納できる．しかし，構造の複雑化を防ぎ，またファイルシステムでの占有領域を抑制するため，構成するファイル数を最小限とする．つまり，OFF2F は実行ファイルを 2 つのファイルに分割し格納する実行ファイル形式である．

2.3 OFF2F の形式と比較

OFF2F の形式では，実行ファイルを構成する 4 つの内容から，アクセス形態を考慮し，2 つの形式が提案された．これを**図 2** に示す．

タイプ A は，ヘッダ部，データ部，および関係情報部を 1 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のファイルとする形式である．タイプ B は，ヘッダ部，テキスト部，および関係情報部を 1 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の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 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に保持され，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．

（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．

以降，OFF2F についてはタイプ A の場合を述べる．

2.4 OFF2F の利用

仮想記憶機構において，OFF2F プログラムを実行するときのマッピングの様子を**図 3** に示す．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

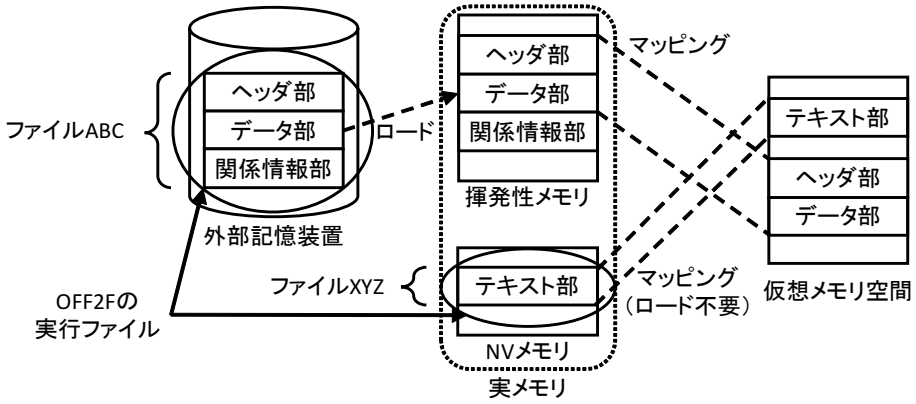

図 3　OFF2F プログラム実行時のマッピング

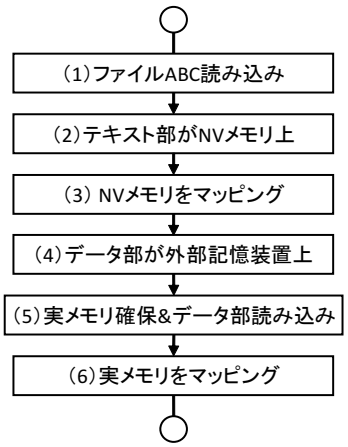

図 4　仮想メモリ空間の作成処理流れ

ト部とデータ部を仮想メモリ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．

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と比べて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，容量の大きな 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．

3. 擬似不揮発性メモリ

3.1 要求

揮発性メモリのみで構成される既存計算機を用いて，擬似 NV メモリを実現する．擬似 NV メモリ実現への要求を以下に示す．

（要求 1）擬似 NV メモリはバイト単位アクセスできること

NV メモリと同様に，擬似 NV メモリはバイト単位でアクセスできる必要がある．

（要求 2）シャットダウン時に NV メモリの内容が失われない特性を再現すること

NV メモリは不揮発性であるため，計算機をシャットダウンした際にメモリの内容が失われない特性を持つ．この特性を擬似 NV メモリで再現する必要がある．

（要求 3）擬似 NV メモリ上にテキスト部を配置して仮想メモリ空間にマッピングできること

OFF2F では，テキスト部を NV メモリ上に配置する．また，OFF2F プログラムを実行したときのテキスト部に対するページ例外処理では，NV メモリ上のページを仮想メモリ空間にマッピングする．このため，擬似 NV メモリ上に OFF2F のテキスト部を配置し，マッピングできる必要がある．

3.2 対処

各要求を満足する各対処を以下に述べる．

（対処 1）擬似 NV メモリとして，実メモリの一部を使用する．これにより，（要求 1）を満足できる．擬似 NV メモリを実現した際の実メモリ構成を図 5 に示す．カーネルが使用する実メモリの大きさは，カーネルの起動パラメータにより設定できる．これにより，実メモリの一部をカーネルが使用しない領域（以降，OS 管理外領域）とし，この領域を擬似 NV メモリとして使用する．

（対処 2）上記の（対処 1）により，カーネルが使用する実

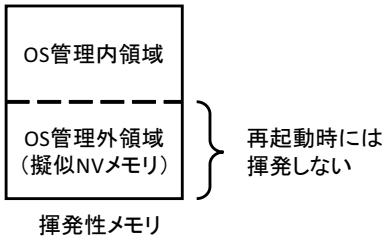

図 5　擬似 NV メモリを実現した際の実メモリ構成

メモリの大きさは制限され，OS 管理外領域は初期化処理に使用されない．また，この領域の内容は OS 管理外領域のため，電源を供給していればソフトウェアリセット時に失われることはない．ただし，ソフトウェアリセット時，OS 管理外領域も含めメモリの内容が初期化され上書きされる可能性があり，対処が必要である．簡易な対処として，初期化され上書きされるメモリ領域を使用しない方式がある．

（対処 3）擬似 NV メモリにテキスト部を配置するためには，擬似 NV メモリにアクセスできる必要がある．しかし，（対処 1）により，擬似 NV メモリは OS 管理外領域であるため，既存のインタフェースを用いてアクセスできない．そこで，擬似 NV メモリへアクセスできる機能を実現する．この機能については後述する．

4. 擬似不揮発性メモリアクセス機能

4.1 基本機能

擬似 NV メモリは OS 管理外領域であるため，既存のインタフェースを用いてアクセスできない．そこで，擬似 NV メモリアクセス機能として，OS 管理外領域である擬似 NV メモリへのアクセスを可能にする機能，具体的には，擬似 NV メモリを仮想メモリ空間にマッピング可能にする機能を実現する．

4.2 実現方式

仮想記憶機構では，仮想メモリ空間に実メモリをマッピングすることで実メモリへアクセスできる．また，実メモリはページに分割して管理され，アクセスはページ単位で行われる．

NV メモリへのアクセスも，実メモリ同様にページ単位で行う．擬似 NV メモリをページ単位に分割し，仮想メモリ空間にマッピングする．実現において，擬似 NV メモリをページ単位に分割する処理は，既存処理を改変して利用する．また，擬似 NV メモリのページを仮想メモリ空間にマッピングする処理は，既存処理をそのまま利用する．

4.3 インタフェースと処理内容

擬似 NV メモリアクセス機能を提供する nvm_mapping()

表 1　擬似 NV メモリアクセス機能を提供するシステムコール

形式	`nvm_mapping(paddr, vaddr, pages)`
引数	`paddr`：アクセス可能にする擬似 NV メモリの先頭実アドレス `vaddr`：マッピング先の仮想アドレス `pages`：カーネルに追加するページ数
戻り値	成功：1 失敗：0
機能	呼出元プロセスの仮想アドレス `vaddr` と擬似 NV メモリの実アドレス `paddr` を `pages` 分対応付け，`vaddr` から擬似 NV メモリへのアクセスを可能にする．

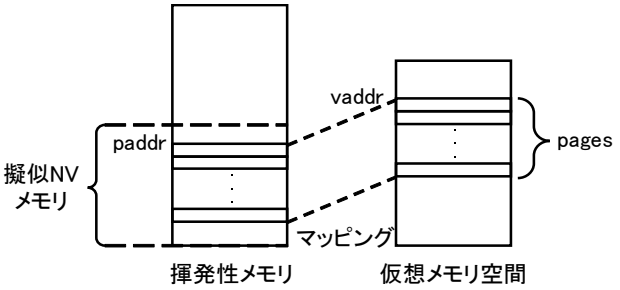

図 6　nvm_mapping() システムコールによる空間構成

システムコールの形式を**表 1** に示す．nvm_mapping() システムコールは，アクセス可能にする擬似 NV メモリの先頭実アドレス，マッピング先の仮想アドレス，およびカーネルに追加するページ数を指定する．引数に用いるアドレスは，どちらもページサイズの倍数を指定する．ここで，擬似 NV メモリ上の実アドレスは，（対処 1）で OS 管理外領域としたアドレスをもとに指定する．

nvm_mapping() システムコールに基づく空間構成を**図 6** に示し，処理内容を以下に説明する．

（1）OS が管理する全ページを保持するカーネル内変数を pages だけ増やす．

（2）擬似 NV メモリの実アドレス paddr を先頭とする pages 分のページをカーネル内の既存のページキューに追加する．

（3）追加したページとして与えられた仮想アドレス vaddr にマッピングする．

上記の処理により，プロセスは，指定した仮想アドレスで擬似 NV メモリにアクセスできる．

4.4 利用方法

混載環境において OFF2F プログラムを実行するために，テキスト部のファイル（XYZ）を NV メモリ，データ部などのファイル（ABC）を外部記憶装置に格納する．このために，NV メモリ上にファイルシステムを構築し，外部記憶装置上のファイルシステムと連携する．つまり，擬似 NV メモリにファイルシステムを構築する必要がある．

この構築の様子を**図 7** に示し，以下に説明する．

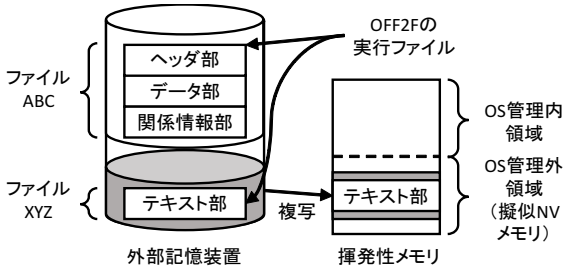

図 7 外部記憶装置の内容を擬似 NV メモリへ複写

（1）外部記憶装置のパーティションを 2 個用意し，各パーティションにファイルシステムを構築する．
（2）1 個のファイルシステムにはデータ部などのファイル（ABC）を格納し，もう 1 個にはテキスト部のファイル（XYZ） を格納する．
（3）nvm_mapping() システムコールを用いて，擬似 NV メモリを仮想メモリ空間にマッピングする．ここでは，このマッピングされた仮想メモリ空間領域を NVM と呼ぶ．
（4）テキスト部のファイル（XYZ） を格納したファイルシステムのパーティションを RAW デバイス（ここでは，RAW と略す）として扱い，RAW の データを NVM に複写する．

以上により，擬似 NV メモリ上にファイルシステムを構築でき，テキスト部のファイル（XYZ）を格納できる．

5. 擬似不揮発性メモリを用いた OFF2F プログラムの実行

5.1 従来の仮想メモリ空間作成処理

既存の実行ファイル形式として ELF がある．FreeBSD について，ELF のプログラム実行時の仮想メモリ空間の主な作成処理流れを図 **8** に示し，以下に説明する．
（1）実行するプログラムのファイルを選定する．
（2）ファイルのサイズ分のページを確保する．ただし，確保するページ数はファイルのサイズに依存し，最大 16 ページである．
（3）処理（2）で確保したそれぞれのページにファイルの内容を読み込む．
（4）ヘッダ部読み取りのために，先頭のページをカーネル空間のマッピング表に登録する．
（5）プロセスの仮想メモリ空間を新たに作成する．
（6）ヘッダ部の情報をもとに，仮想メモリ空間にテキスト部とデータ部のエントリを作成する．プログラム実行時にページ例外が発生した場合，エントリの持つ情報をもとに外部記憶装置から読み込む．
なお，処理（2）（3）では，64KB（1 ページ 4KB）以下のサイズのファイルであれば全体を読み込む．64KB より大きいサイズであれば，先頭から 64KB を読み込む．

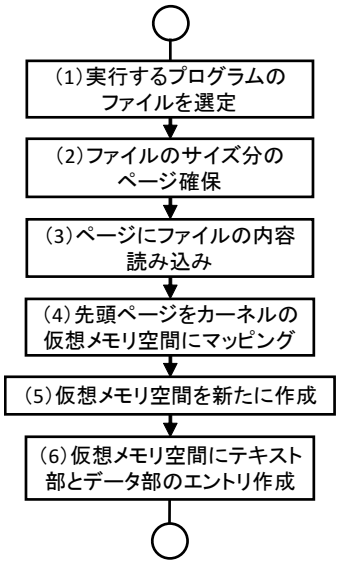

図 8 ELF プログラム実行時の仮想メモリ空間作成処理流れ

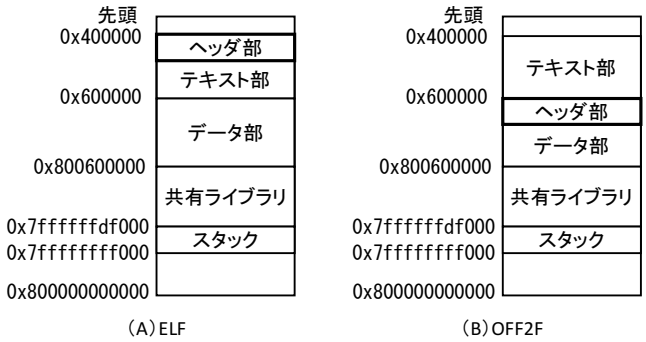

図 9 プログラム実行時の仮想メモリ空間のメモリマップ

5.2 OFF2F プログラム実行時の仮想メモリ空間作成処理

プログラム実行時，ヘッダ部読み取りの後にファイルの先頭 4 バイトの内容をもとに，実行ファイルが OFF2F であることを認識する．このため，OFF2F プログラムのヘッダ部は，ELF プログラムのヘッダ部のうち，以下を変更する．
（1）ファイルの先頭 2〜4 バイトを “ELF” から “OFF” に変更
（2）テキスト部のファイル（XYZ）のパスを追加
（3）テキスト部のオフセットの情報を削除
（4）データ部のオフセットの情報を変更

ELF と OFF2F について，プログラム実行時に作成される仮想メモリ空間のメモリマップを図 **9** に示す．ELF ファイルは，ヘッダ部とテキスト部が連続して存在する．このため，仮想メモリ空間のメモリマップでは，ヘッダ部の直後にテキスト部を配置する．これに対し，OFF2F ファイルではヘッダ部とデータ部が連続して存在する．このため，仮想メモリ空間のメモリマップでは，ヘッダ部の直後にデータ部を配置する．

図 2（A）に示したように，データ部などのファイル

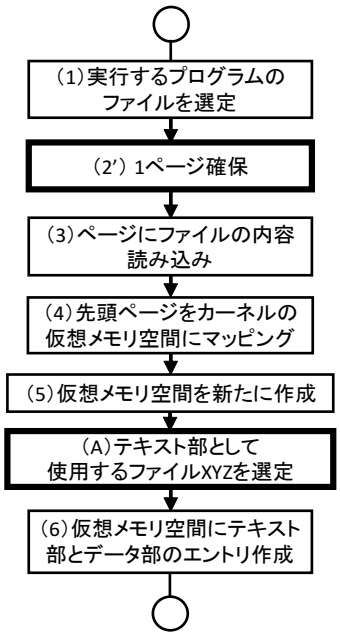

図 10 OFF2F プログラム実行時の仮想メモリ空間作成処理流れ

（ABC）は外部記憶装置上に存在し，テキスト部のファイル（XYZ）は擬似 NV メモリ上に存在する．OFF2F プログラム実行時の仮想メモリ空間の作成処理流れを**図 10** に示し，ELF（図 8）の場合との違いを以下に説明する．

多くのプログラムにおいて，テキスト部に比べデータ部は小さい．例えば，文献 [6] では，FreeBSD の/bin 下のファイル群において，全プログラムのテキスト部の総和は，データ部の総和の約 20 倍の大きさであることが示されている．したがって，図 8 の処理（2）では，ヘッダ部とテキスト部を含めたファイルの先頭 64KB が読み込まれるが，OFF2F プログラム実行時には，ファイルの先頭 4KB（1 ページ）を読み込むこととした．つまり，(2) を以下のように改造した．

（2'）ヘッダ部とデータ部を含めた先頭 4KB（1 ページ）を読み込む．

また，OFF2F は 2 つのファイルで構成されるため，図 8 の処理 (5) と処理 (6) の間に，テキスト部のファイル（XYZ）を選定する処理を追加する必要がある．つまり，以下の処理の追加が必要である．

（A）テキスト部のファイル（XYZ）を選定する．

上記の変更のみで OFF2F プログラム実行時の仮想メモリ空間の作成ができる．ただし，例えば処理（6）について，テキスト部のエントリ作成の際に ELF では実行ファイルの情報を格納するが，OFF2F ではテキスト部のファイル（XYZ）の情報を格納する．

5.3 ページ例外処理

擬似 NV メモリを利用したページ例外処理の流れを**図 11** に示し，以下に説明する．

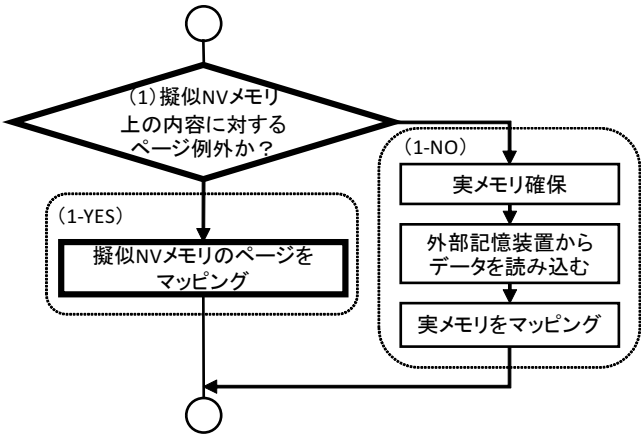

図 11 ページ例外処理の流れ

（1）擬似 NV メモリ上の内容に対するページ例外であるか否かを判定する．

（1-YES）擬似 NV メモリ上の内容に対するページ例外である場合，擬似 NV メモリ上を探索し，対応する実アドレスをマッピングする．

（1-NO）擬似 NV メモリ上の内容に対するページ例外ではない場合，既存の ODP 処理を行う．

なお，擬似 NV メモリ上の内容に対するページ例外であるか否かは，ページ例外の発生した仮想アドレスにより判別する．エントリの持つ情報をもとに擬似 NV メモリの実アドレスを算出し仮想メモリ空間にマッピングする．

6. おわりに

揮発性メモリと NV メモリが混載された計算機環境を対象とする実行ファイル形式 OFF2F の効果を検証する支援環境として，揮発性メモリのみで構成される既存計算機を用いて擬似 NV メモリを実現する方式を述べた．

この実現方式は，擬似 NV メモリとして揮発性メモリの一部を利用する．具体的には，カーネルが使用する実メモリの大きさを制限し，カーネルが使用しない領域を NV メモリとして擬似する．また，擬似 NV メモリの領域をページ単位に分割し，仮想メモリ空間にマッピングする機能を提供することにより，擬似 NV メモリにアクセスできる．この機能を用いて擬似 NV メモリにファイルシステムを構築し，OFF2F プログラムのテキスト部を配置する．さらに，FreeBSD において，仮想メモリ空間の作成処理とページ例外処理を変更することで，擬似不揮発性メモリを用いて OFF2F プログラムを実行する方式について述べた．

残された課題として，擬似 NV メモリ上に構築したファイルシステムとの連携と OFF2F の実装による評価がある．

謝辞 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] Qingsong Wei, Jianxi Chen, Cheng Chen : Accelerating File System Metadata Access with Byte-Addressable Nonvolatile Memory, ACM Transactions on Storage (TOS), Vol.11, Issue 3, pp.1–28 (2015).
[2] Kazuichi Oe, Mitsuru Sato, Takeshi Nanri : Automated tiered storage system consisting of memory and flash storage to improve response time with input-output (IO) concentration workloads, 2017 Fifth International Symposium on Computing and Networking (2017).
[3] Doe Hyun Yoon, Tobin Gonzalez, Parthasarathy Ranganathan, Robert S. Schreiber : Exploring latency-power tradeoffs in deep nonvolatile memory hierarchies, Proceedings of the 9th conference on Computing Frontiers, pp.95–102 (2012).
[4] Yiying Zhang, Steven Swanson : A Study of Application Performance with Non-Volatile Main Memory, Proceedings of 2015 31st Symposium on Mass Storage Systems and Technologies (MSST), pp.1–10 (2015).
[5] Xiaochen Guo, Aviral Shrivastava, Michael Spear, Gang Tan : Languages Must Expose Memory Heterogeneity, Proceedings of the Second International Symposium on Memory Systems, pp.251–256 (2016).
[6] 谷口秀夫：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, コンピュータシステム・シンポジウム論文集，Vol.2017，pp.35–40 (2017)．
[7] 河辺誠弥, 谷口秀夫, 佐藤将也：FreeBSD の初期化処理における OFF2F の効果予測，情報処理学会研究報告, vol.2018-OS-142, no.2, pp.1–8 (2018.02)．
[8] 河辺誠弥, 佐藤将也, 谷口秀夫：FreeBSD の初期化処理におけるページ例外に着目した OFF2F の効果予測，コンピュータシステム・シンポジウム (ComSys2018) 論文集, pp.57–64 (2018.11)．
[9] 谷口秀夫：OFF2F プログラムのページ例外処理における CoW 機能を考慮した評価，第 17 回情報科学技術フォーラム（FIT2018）講演論文集, vol. 第 1 分冊, pp.149–150 (2018.09)．
[10] 谷口秀夫, 佐藤将也 : CoW 機能を考慮した OFF2F プログラムのページ例外処理の評価，平成 30 年度（第 69 回）電気・情報関連学会中国支部連合大会, 電子媒体 (2018.10)．