B-013

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

Evaluation considering CoW in page exception handling of OFF2F program

谷口秀夫†

TANIGUCHI, Hideo

１．はじめに

不揮発性メモリ（Non-Volatile Memory：以降、NV メモリと略す）の登場により、この NV メモリを利用した制御機構の研究が盛んに行われている。

オペレーティングシステムのプログラム実行機構において、NV メモリの特徴を生かすための新しい実行プログラム形式（OFF2F：Object File Format consisting of 2 Files）[1]が提案されている。OFF2F は、プログラムを実行する際の参照と更新の特徴、および不揮発性メモリの特徴に着目し、それらを生かしてプログラムを実行できる形式である。また、揮発性メモリと不揮発性メモリの混在した計算機環境だけではなく、揮発性メモリのみを搭載する現在の計算機環境でも利用できる形式である。文献[1]では、OFF2F を利用して、ページ例外（PF）処理への効果を述べている。しかし、PF 処理におけるもうひとつの主要処理である CoW(Copy on Write)機能の処理に触れていない。

そこで、本稿では、CoW 機能の処理も考慮した PF 処理について述べるとともに、その処理時間について考察し、OFF2F の効果を述べる。

２．OFF2F

実行プログラムは、a.out 形式などがあり、テキスト部とデータ部は一つのファイルに格納されている。しかし、プログラム実行時、テキスト部は参照のみ、データ部は読み書きされる。そこで、これらを別ファイルにする形式が OFF2F であり、テキスト部を NV メモリへ格納することで、プログラムの高速実行を目指している。NV メモリはバイト単位アクセスが可能であるため、NV メモリ上のテキスト部は、揮発性メモリ上に複写することなく、そのまま実行できる。したがって、仮想記憶機構の PF 処理において、処理時間を短縮可能である。

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

３．ページ例外処理

３．１　処理流れ

OFF2F における ODP(On Demand Paging)機能および CoW 機能を有する際の PF 処理を図２に示し、以下に説明する。

（１）実メモリが存在せず、かつ NV メモリ上データに対する例外で無い場合、通常の ODP 処理(A)を行う。この時、「実メモリ確保」において実メモリ不足が発生すればページアウト処理を行う。その後、CoW 処理の要否を確認し、CoW 処理(C)を行う。

（２）実メモリが存在せず、かつ NV メモリ上データに対する例外で有る場合、NV メモリ対応処理(B)として、当該の NV メモリのページをマッピング表に登録する。

（３）実メモリが存在する場合、CoW 処理が必要であれば、CoW 処理(C)を行う。なお、CoW 処理が必要なければ「プログラムのバグ」（パニック処理）である。ここでは、正常

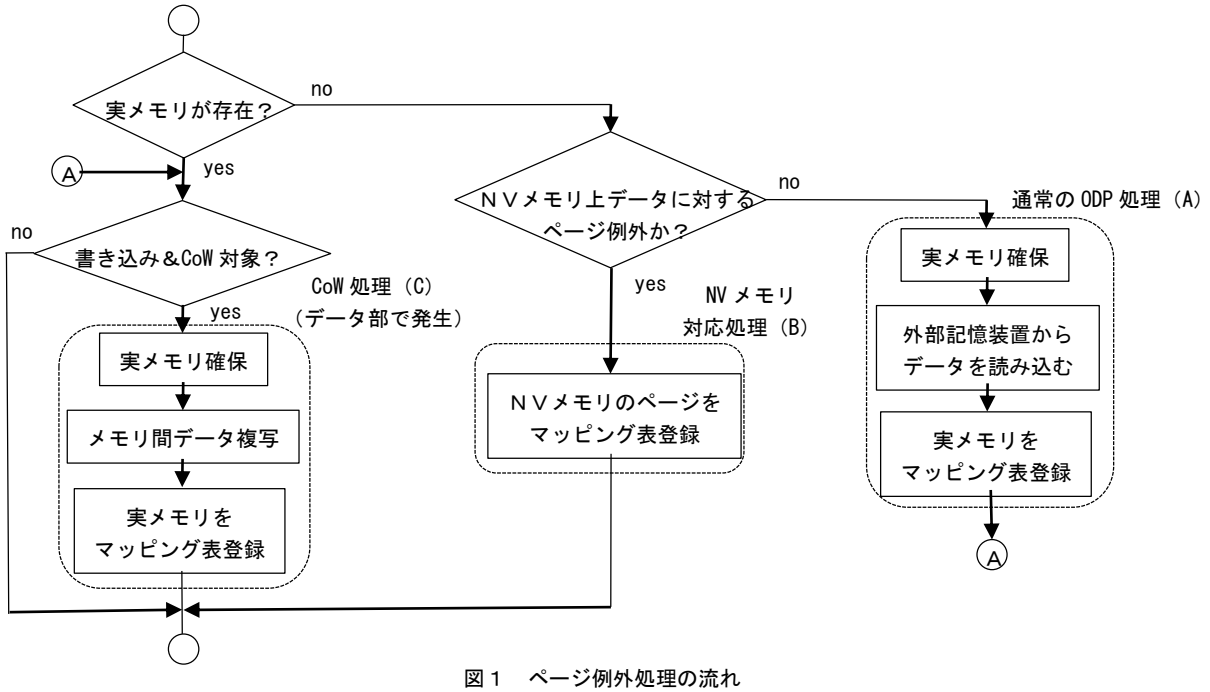


図１　ページ例外処理の流れ

な処理内容を中心に記したので、図では省略している。

３．２　ページ例外処理時間の定式化

図２に基づき、PF 処理の時間を定式化する。

以下のように各処理の時間を定義する。

・$t1$：実メモリ確保（１ページ：４KB）
・$t2$：外部記憶装置からデータ（４KB）を読み込む
・$t3$：実メモリをマッピング表に登録
・$t4$：NV メモリのページをマッピング表に登録
・$t5$：メモリ間のデータ複写（４KB）

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
・C：データ部に占める CoW 対象ページ数の割合

とする。

次に、プログラムが全て外部記憶装置に格納されている場合（従来）、およびテキスト部は NV メモリ上、他は外部記憶装置に格納されている場合（OFF2F）について定式化する。なお、定式化の条件として、プログラム全体がページ例外によりメモリ上にロードされ、１つのプログラムから n 個のプロセスを生成し実行する。

（従来）の場合、以下の式(1)となる。

$$(t1+t2+t3)P+\sum_{i=2}^{n}(t1+t5+t3)P(1-S)Ci$$

第１項はプログラム全体の PF 処理時間であり、第２項は２個目以降のプロセスで行われる CoW 処理時間の和である。当然ながら、CoW 処理はデータ部の PF で発生する。Ci は、i 番目に生成されたプロセスのデータ部に占める CoW 対象ページ数の割合であり、$C1$ は０とすることができる。

また、（OFF2F）の場合、以下の式(2)となる。

$$t4PS+(t1+t2+t3)P(1-S)+\sum_{i=2}^{n}(t1+t5+t3)P(1-S)Ci$$

第１項はテキスト部のページ例外処理時間であり、第２項はデータ部の PF 処理時間であり、第３項は２個目以降のプロセスで行われる CoW 処理時間の和である。

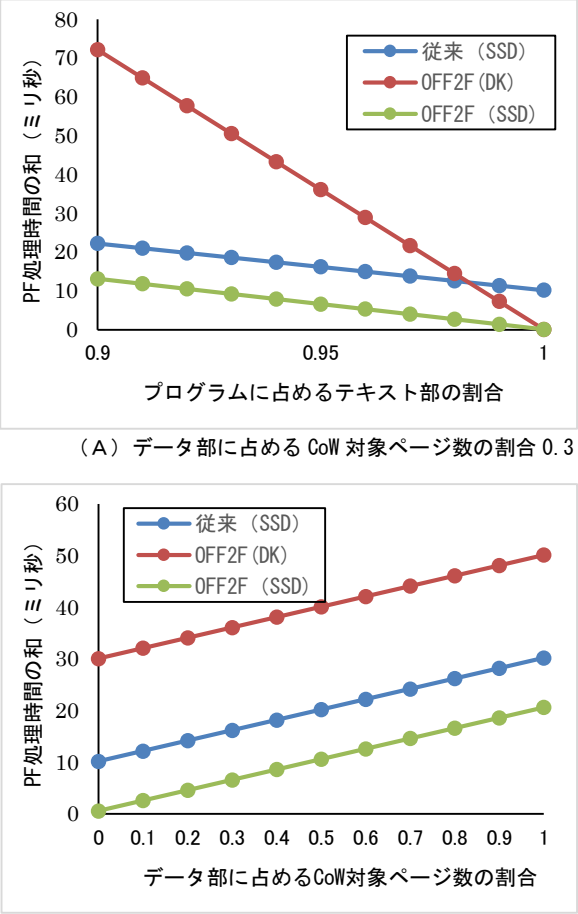
３．３　考察

式(1)(2)から、以下のことがわかる。

（１）式(1)の第２項および式(2)の第３項は、CoW 処理時間であり、同じ内容である。したがって、OFF2F による PF 処理時間短縮の効果は大きいものの、CoW 機能の処理を考慮すると、効果の割合は小さくなる。

（２）各式は、S に関し、１次関数であり、かつ S の係数は負であり、その絶対値は式(2)＞式(1)である。したがって、プログラムに占めるテキスト部の割合（S）が大きいほど、OFF2F の PF 処理時間短縮効果は大きくなる。

（３）各式は、Ci に関し、１次関数であり、かつ Ci の係数は正であり、両式とも同じ値である。したがって、データ部に占める CoW 対象ページ数の割合（Ci）に PF 処理時間は比例する。



（Ａ）データ部に占める CoW 対象ページ数の割合 0.3

（Ｂ）プログラムに占めるテキスト部の割合 0.95

図２　プログラム全体のページ例外処理時間（プロセス数５）

また、文献[1]の評価と同様な条件（$t2$ = 6ms（DK），0.1ms（SSD）、P = 100 ページ）で、$t5$ = 1ms で他は ti = 0.001ms、さらにプロセス数５（n = 5）の場合について、プログラム全体の PF 処理時間を図２に示す。なお、テキスト部サイズはデータ部サイズの約 20 倍[1]であるから、プログラムに占めるテキスト部の割合を 95%前後とした。図２は、上記（２）と（３）の内容を示している。

４．おわりに

OFF2F における CoW 機能を考慮した PF 処理時間を定式化した。OFF2F による PF 処理時間短縮の効果は大きいものの、CoW 機能の処理を考慮すると、効果の割合は小さくなる。残された課題として、提案手法の実装評価がある。

＜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，“揮発/不揮発性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).