不揮発性メモリを用いた障害後のプロセス復元手法

2012SE170 村林工
指導教員：宮澤元

1 はじめに

近年，永続的にデータを保存できる不揮発性メモリの開発が進んでいる．バイトアクセス可能な不揮発性メモリの大容量化が進めば，将来的には磁気ディスクのような低速デバイスを持たず永続データを含めた全ての情報を主記憶装置に保存するようなアーキテクチャが登場すると予想される．そのようなアーキテクチャでは，障害発生直前の主記憶装置の状態を再起動後に読み取ることも可能となる．

このような特性を用いてプロセスのチェックポイントを不揮発性メモリ上に取り，プロセスの信頼性を向上する研究が進んでいる．しかしこれらの手法の多くはチェックポイントを取るデータやタイミングの選択をアプリケーションに委任しており，アプリケーション開発者は適切な選択をする手間が増える．OS(Operating System) がタイミングやデータを決定する手法も存在するが，障害の発生したタイミングによっては，アプリケーション再開時に余計な手戻りが増加する可能性がある．またこれらの手法では主に単一のプロセスに対しての信頼性向上を目的としているので，複数プロセスを用いる処理への対応は十分でない．

本研究では，不揮発性メモリを主記憶装置とする計算機環境において，電源喪失等の主記憶装置の故障以外のハードウェア障害発生後に，ハードウェア障害の原因を取り除きさえすれば，プロセスを安全に復元できる手法を提案する．OS がチェックポイントのタイミングを決定するので，アプリケーション開発者は障害発生時の処理をほとんど考えることなくアプリケーションを開発できる．また，再起動後にカーネル空間内に残ったデータを利用し復元することで，複数プロセスに紐づくカーネル空間内のデータが復元でき，プロセス間通信等の複数のプロセスを用いた処理も安全に再開できる．

2 研究の背景

本節では研究の背景である次世代不揮発性メモリについて示し，それらの特徴を活かした信頼性の向上手法の研究を示した後に，これらのシステムの問題点を示す．

2.1 次世代不揮発性メモリ

近年，バイトアクセス可能な不揮発性メモリの研究が進んでおり，更なる研究開発によって不揮発性メモリが主記憶装置に採用された計算機環境が普及すると考えられる．バイトアクセスが可能な不揮発性メモリには Phase-Change Memory(PCM)，Resistance RAM(ReRAM)，Magneto-resistive RAM(MRAM) などがある．これらの不揮発性メモリには速度・耐久性・書き込み回数制限等の問題点も多く残っているが，これらの欠点を克服する研究も多く提案されており，将来的に主記憶装置に採用することが可能になると考える．

2.2 プロセスの復元

次世代不揮発性メモリを主記憶装置に採用することで，高速に主記憶装置に永続データを残すことが可能になる．その性質を利用して，プロセスのチェックポイントを不揮発性メモリ上に取るなどし，プロセスの信頼性を向上する研究が進んでいる．Mnemosync[1] や，NV-Heaps[2] では不揮発性メモリと従来の DRAM の両方を用い，主記憶装置内の重要データを永続データとすることで信頼性の向上を図っている．また，BLCR[3] や追川らの研究 [5]，小林らの研究 [4] のように，プロセスのメモリ空間やその時のレジスタ，カーネル空間内のプロセスに関するデータの全てをチェックポイントとして記録する手法も存在する．

2.3 問題点

前節で述べたプロセスの復元手法では，プロセス単体やプロセスグループでの復元を前提としており，プロセス間通信など複数プロセスを利用した処理を考慮していない．

また，障害発生後に主記憶装置に残ったデータを用いてプロセスの復元を行う場合，障害発生直前に実行中であったプロセスのレジスタの値を取得できないなど単純な方法では障害発生直前の状態に戻すための実行コンテキストを完全には取得できない．

3 カーネル空間内に残ったデータを利用したプロセス復元

前節で述べた問題点を解決するため，障害発生による再起動後にカーネル空間内に残ったデータを利用し，プロセスを復元する．そのための手法を本節で提案する．

3.1 概要

従来のチェックポイントシステムでは，プロセスのメモリ空間やレジスタのみでなく，プロセスがどのファイルを開いているかなどの情報もチェックポイントに含めている．プロセス再開後にそのような情報がなければ，ファイルの書き込み処理などをすることが出来ないからである．

本研究では不揮発性メモリを主記憶装置に採用した計算機環境において，チェックポイントとしてチェックポイントを取った時点のメモリ空間，レジスタの値を主記憶装置の別の場所にコピーする．それ以外にプロセス再開時に必要となるカーネル空間内のプロセスに関連するデータは，再起動後にカーネル空間内に残っているデータを利用する．その為に，再起動後にカーネル空間内のデータの整合性を確保しておく必要がある．

3.2 コンピュータの起動

コンピュータの起動時に，前回の終了が障害による終了であったかどうかを取得する．正常な終了時にフラグを立てるようにすることで，前回の終了が異常終了かどうかを見分けることが可能である．異常終了直後の再起動であれば，OS 起動時に実行するプロセス一覧のリストの初期化処理，プロセスが利用しているファイルの一覧のリストの初期化処理などを行わない．なぜならそれらのデータは主記憶装置が不揮発性メモリであるのでカーネル空間上に残っているからである．

3.3 プロセスの復元

提案手法でプロセスの復元を行う場合，チェックポイントを用いて復元を行うプロセスは，障害発生直前の時点で実行中であったプロセスのみである．実行中でなかったプロセスは，主記憶装置内に待避されているレジスタの値のみで復元することが可能である．

3.4 特権処理中に変更できるデータ

特権処理中に変更できるデータを重要データとそうでないデータにわけ，カーネル空間内の処理が始まる前に重要データのコピーを取る．重要でないデータとは画像バッファなどの多少の変化では動作に問題のないデータである．特権処理中に障害が発生し再起動が行われた場合，データをコピー先からコピー元へ戻しデータの不整合を防ぐ．これにより再起動後でも，障害発生前のカーネル空間内の整合性の取れたデータを参照することが可能となる．

3.5 プロセスへの通知

外部装置を扱うシステムコールは，障害発生後の再起動時にシステムコールの実行を再開せずに中断し，そのことをプロセスに通知する．障害発生と同時に外部装置の電源も止まり，再起動時には外部装置の状態が障害発生前と変化する可能性がある．それによって，システムコールの実行が思わぬ不具合に発展する可能性があるからである．また，ハードウェアの割り込み待ち中のプロセスは wake up し，再起動が行われたことを通知する．再起動後にハードウェアの状態が変化し，割り込みを起こさなくなりプロセスがフリーズを起こす可能性があるからである．

4 実装

3 節で述べた障害対策機能を持つ Linux のような汎用型 OS を独自に Intel x86(32bit) で実装した．

4.1 特権処理で利用するデータのコピー

特権処理で利用する重要なデータが存在するページには ReadOnly とし，そこにデータが書き込まれた際に起こる例外が起こるタイミングで，データのコピーを行う．特権処理中に障害が発生し再起動が行われた場合，ページが ReadOnly かどうかを確認し，ReadOnly でなかったデータはコピー先からコピー元へと戻す．

4.2 システムコールの中断

再起動後にシステムコールを中断する際は，システムコールの引数に中断フラグをセットする．中断フラグが立っている場合，システムコールの途中で中断する．

5 実験

次世代不揮発性メモリを用いた計算機環境をエミュレートするために，Virtual Box の改造を行った．特権モード中に無限ループと割り込み禁止命令が実行されるようにし，アプリケーションが特権モードで止まった際に電源を落とすことで特権モード中の障害を再現し，適当なタイミングで何回か落とし非特権モード中の障害を再現した．その際，VirtualBox のログを確認し非特権モードで落ちていることを確認した．その結果，いずれの場合に置いても正常な再起動が確認でき，本研究の有用性が確認された．

6 まとめ

本論文では，次世代不揮発性メモリを主記憶装置に使用した計算機環境において，再起動後に残ったデータを利用することによって，信頼性の向上を実現する手法を提案した．本論文で提案した機能を実装した Linux のような汎用 OS を独自に作成した．独自に作成した OS を動作させる実験により，提案した手法が有用であることが確認できた．

参考文献

[1] Volos, H., Tack, A. J. and Swift, M. M.: “Mnemosyne: Lightweight Persistent Memory”, Proc. of the 16th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 1:13-13:13(2011).

[2] Coburn, J., Caulfield, A. M., Akel, A., Gupta, R. K., Jhala, R. and Swanson, S.: “NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories”, Proc. of the 16th ACM International Conference on Architectural Support for Programming Languages and Operating System, pp. 1:13-13:13(2011).

[3] Hargrove, P. and Duell, J.: “Berkeley Lab Checkpoint/ Restart (BLCR) for Linux Clusters”, Proc. of Scientific Discovery through Advanced Computing (SciDAC), pp.494-499 (2006).

[4] 追川 修一, 三木 聡 : “Non-Volatile メインメモリを用いたチェックポイント・リスタートシステム”, 情報処理学会論文誌コンピュータシステム, Vol. 6, No.4, pp.49-57(2013).

[5] 小林 直登, 山田 浩史 : “ストレージクラスメモリを用いたアプリケーション透過なチェックポイント取得手法”, 情報処理学会研究報告 Vol.2015-ARC-215 No.10 Vol.2015-OS-133 No.10 (2015).