Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 3, Number 1, pages 81–82, January 2014

Designing a New Memory Management Unit for Hybrid Memory

Hirotaka Kawata
College of Information Science
School of Informatics
University of Tsukuba
Tsukuba, Ibaraki, JAPAN
s0911454@coins.tsukuba.ac.jp

Shuichi Oikawa
Division of Information Engineering
Faculty of Engineering, Information and Systems
University of Tsukuba
Tsukuba, Ibaraki, JAPAN

***Abstract*—With the spread of flash storage, flash as a replacement for part of main memory is hot topics for today's computer system. While having a good random access performance, flash storage is more power efficiency but also having larger memory than DRAM. In this paper, we propose a design of new hybrid memory management system by hardware to assists the hybrid DRAM and flash memory model. It provides LRU page buffer to reduce the page fault interrupt processing, and make efficient object cache on DRAM. To proof of the concept, we will implement new memory management unit to computer system.**

I. はじめに

近年, コンピューターに搭載される記憶装置は, SSD に代表される半導体を用いたフラッシュストレージの登場により, 大幅な高速化が図られた. 従来のハードディスクとは特性が異なるため, 記憶装置の利用方法や最適化手法が変化している.

現在の主なコンピューターシステムでは, 主に DRAM をメインメモリとして利用し, スワップ領域としてハードディスクの一部を利用している. そして, ハードディスクはランダムアクセス性能が芳しくないため, システム全体の性能を向上させるためには, DRAM の増設によるメインメモリの拡張が必要になる. しかしながら, ランダムアクセスが高速であるフラッシュストレージの登場により, それらをメインメモリの一部として利用する方法が考えられてきた. フラッシュストレージは, DRAM より安く低消費電力であるため, 効率的に性能を向上させることができるようになった.

また, スマートフォンやタブレットなど, 比較的作業用メモリを多く必要としながら, 低消費電力を要求するデバイスが登場してきた. これらのデバイスでは, 物理的制限と消費電力の問題から, DRAM の増設が難しく, かつ安価に提供する必要があるため, 新たなメモリシステムの形態が求められている. しかしながら, これらのデバイスは, フラッシュストレージを記憶装置として利用していることが多いので, DRAM の一部をそれらで置き換えることで, 問題を解決できると考えた.

さらに, サーバー用途においても Web においてよく使用される memcached[1] のような Key Value Store は, 大量のオブジェクトをメモリ上に常駐させ, 高速なアクセスを実現している. しかし, Key Value Store で利用されるオブジェクトは, すべてが頻繁にアクセスされるわけではないので, すべてを DRAM に常時配置しておく必要はない. このような要求があるアプリケーションでは, ハードディスクを利用したスワップ領域ではパフォーマンスの要求を満たさないが, ランダムアクセスが高速なフラッシュストレージをメモリとして利用することで, 少ない DRAM で同程度以上のパフォーマンスを実現することができる.

以上の背景を踏まえて, 本研究では DRAM とフラッシュストレージが共存したコンピューター上で, 効率的にフラッシュストレージをメインメモリの一部として利用することを前提とした, メモリマネジメントユニット (MMU) を提案する.

II. フラッシュストレージのメインメモリとしての利用

メインメモリの一部としてフラッシュストレージを利用するには, スワップ領域として利用する方法と, オブジェクトストアとして利用する方法などが考えられる. 従来は, スワップ領域として利用する方法が一般的であったが, その特性からオブジェクトストアとして利用する方法が研究されている. また, メインメモリを, フラッシュストレージとして利用するには, いくつかの問題があり、同時にそれらを解決する必要がある. これらは, 書き込み回数に制限があり, 適切に管理しないと寿命が縮んでしまうといった問題, 読み込みに比べて, 書き込みのレイテンシが遅いという問題, そして, ブロックデバイスとしてセクタ単位でアクセスのアクセスが必要, といった問題である.

Saxena らの研究 [3] では, スワップ領域としてフラッシュストレージを利用する, 仮想メモリ管理システム FlashVM を提案している. この研究では, フラッシュストレージのランダムアクセス性能が良い事を利用して, なるべく書き込み回数が少なくなるような Page Write-back アルゴリズムの提案, discard コマンドを利用したガベージコレクションの提案などを行い, 従来の仮想メモリ管理システムに比べて, 性能と寿命を向上させている.

また, Badam らの研究 [4] では, ランタイムライブラリによる DRAM とフラッシュのハイブリッドメモリ管理システム SSDAlloc を提案している. この研究では, Object Per Page (OPP) という概念を採用し, 1 オブジェクトは 1 ページとして永続的に同じアドレスのメモリ上に Page Buffer として配置される. さらに, SSD と Page Buffer の間に RAM Object Cache を導入し, その中では実際のオブジェクトサイズのメモリ領域しか割り当てられない. このような仕組みにより, SSD が提供する大容量のメモリ空間と, 効率的な DRAM の利用を両立している. また, ランタイムライブラリとして提供しているので, 既存のシステムやソフトウェアに対する変更を最小限に抑えることができる.

しかしながら, SSDAlloc はソフトウェアのソースコードの変更なしでは, 最大限に恩恵を預かることができない. だが, FlashVM よりは性能が良く理想的な方法である.

III. 提案するシステムの概要

II で述べたような問題を解決し, 既存研究で行われてきたフラッシュのためのメモリシステムの技術を参考に, いくつかの問題点についての解決を行うため, 本研究では DRAM とフラッシュのハイブリッドメモリを実現する MMU を搭載するシステムを提案する. Bodam らの SSDAlloc の実装 [4] を参考に, 更に改良点を加え, ハードウェア MMU を成す. 提案するシステムの特徴は以下のようなものである.

- 1 Object per Page: Page Buffer 1 ページに, 1 オブジェクトを割り当てる方式である. この方式にすることで, メモリの利用効率は下がるが, オブジェクトごとの利用状況を詳細に取得することができる. よって, オブジェクト移動を少なくするアルゴリズムを適用することができる.
- Page Buffer の LRU 化: SSDAlloc では, Page Buffer と RAM Object Cache の間のオブジェクトの移動が, FIFO によって行われている. ランタイムライブラリによる実装では, これらの移動を LRU で実装することは困難である. しかしながら, FIFO により頻繁に利用するオブジェクトが Page Buffer から押し出されると, ページフォールトにより割り込み処理が発生することになる. ハードウェアの支援を借りた実装であれば, Read フラグや参照カウントを持つテーブルを用意することで, この部分を LRU で扱うことができる. 頻繁に利用するオブジェクトは, Page Buffer から押し出されることはなくなる.
- Page Buffer と RAM Object Cache 間の移動のコスト削減: SSDAlloc では, Page Buffer と RAM Object Cache 間の移動は, ページフォールトによる割り込み処理で実現されている. しかしながら, 頻繁に処理が起こるこの部分で, 割り込みを起こすとパフォーマンスが低下する. よって, 本来の DRAM のパフォーマンスに近づけるには, このオーバーヘッドをなくすことが重要だと考えた. MMU のハードウェア支援により, ページフォールトを起こさずにこの部分のやりとりを実現, またはその移動コストを最適化する仕組みを用意する.
- 従来型 MMU とフラッシュハイブリッド MMU との共存: 本研究で提案するハイブリッドメモリに最適化された MMU は, 従来型の MMU と互換性がない. システムやソフトウェアの変更を最小限に抑えるためには, 従来型の MMU と親和性が高いシステムにする必要がある. そのため, 従来型の MMU と併用して利用できる仕組みを用意する. 特に, 比較的小さな領域を大量に確保するアプリケーションに関しては, 本研究で提案する MMU が最適であるが, 大きなメモリ領域を確保する場合に関してはその限りではない. よって, 併用する際には, アプリケーションや割り当てるメモリを利用するオブジェクトの性質により, 従来型 MMU との使い分けが重要になる.

IV. ロードマップ

III に於いて提案した MMU は, MMU を従来型の MMU と併用する必要があり, MMU はプロセッサーに密に結合されたユニットであるため, 既存の MMU の流用では実装が難しい. そのため, 本研究では新たに MMU の設計を行い, 実装する.

新規 MMU の実装を進めていくにあたって, 現在私が共同開発中の MIST32 プロセッサ [5] を搭載したコンピューターシステムをターゲットシステムとする. このプロセッサは, 標準的な RISC プロセッサの命令体型を採用した, 32bit プロセッサである. 近年の高性能モバイル機器をターゲットとしたプロセッサで, 標準的な従来型の MMU が搭載されたコンピューターシステムを構築できる.

手順としては, まずソフトウェアシミュレーター上に上記で提案した MMU の実装を行い, その後シミュレーター上で本研究で提案した MMU に関するベンチマークを行う. 従来型 MMU とスワップとの併用や, DRAM のみを利用した場合のベンチマーク比べて, 設計の妥当性を検討する. また, 意図的に DRAM 領域 (Page Buffer, RAM Object Cache) をのサイズを可変させベンチマークを測ることで, ハイブリッドメモリをどの程度効率よく利用できるか, DRAM のみに比べてどの程度のパフォーマンス低下で完結するかと言った情報が取れると考えている.

その後, シミュレーター上に実装したシステムを元に, FPGA を利用した MMU 回路の実装を落とし込み, 開発中のプロセッサと共に FPGA 上でハードウェアとして動作させる. その上で, ベンチマークを行い結果を評価し, ハードウェアとしての回路規模とパフォーマンスとのバランスを測定する.

V. まとめと今後の課題

本論文では, フラッシュストレージと DRAM のハイブリッドなメモリモデルを利用した新しい MMU の設計を提案した. 示したロードマップの中で, 現段階では MMU の設計と問題の解決方法について検討している段階である.

特に, RAM Object Cache において Pool allocator を利用しているため, この部分をどのようにハードウェアに最適な形で落とし込むかが, 重要な課題となる. また, 従来型の MMU との併用で, どのようにメモリ割り当てを振り分けるか, オペレーティングシステムの側から, これらの MMU をどのように活用するかが課題となる.

この先実装を進め, 提案したシステムについてベンチマークを取り, 利点や課題についてさらに評価をしていく予定である.

REFERENCES

[1] Memcached ,. http://www.danga.com/memcached/

[2] Michael Stonebraker. 2010. SQL databases v. NoSQL databases. Commun. ACM 53, 4 (April 2010), 10-11. DOI=10.1145/1721654.1721659 http://doi.acm.org/10.1145/1721654.1721659

[3] Mohit Saxena and Michael M. Swift. 2010. FlashVM: virtual memory management on flash. In Proceedings of the 2010 USENIX conference on USENIX annual technical conference (USENIXATC'10). USENIX Association, Berkeley, CA, USA, 14-14.

[4] Anirudh Badam and Vivek S. Pai. 2011. SSDAlloc: hybrid SSD/RAM memory management made easy. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (NSDI'11). USENIX Association, Berkeley, CA, USA, 16-16.

[5] Ito, Takahiro, and Hirotaka Kawata. "MIST32 Processor Architecture." Open Design Computer Project. N.p., n.d. Web. 22 Oct. 2013.