3H-02

SSDの耐久性を高めるキャッシュアーキテクチャの検討

小川 愛理† 吉瀬 謙二†

†東京工業大学 大学院情報理工学研究科

1 はじめに

近年，ストレージシステムとして従来の Hard Disk Drive (HDD) に替わって Solid State Drive (SSD) が多く利用されている．SSD は HDD に比べて消費電力が少なく，読み書きが高速であるなど多くのメリットを持つ一方で，内部フラッシュメモリの書き込み可能回数が非常に少ないという大きなデメリットを抱えている．

これを軽減するために多くの SSD では，書き込み要求のバッファキャッシュとして機能する大容量の random access memory (RAM) や，アドレスマッピング機構である Flash Translation Layer (FTL) が導入されている．FTL にとって都合の良いデータ系列をバッファキャッシュから出力することで，フラッシュメモリへの書き込み要求を大きく減らし，耐久性を高めることが可能である．我々はこれらの機構に着目し，さらなる SSD の耐久性向上を達成するための FTL を考慮したアーキテクチャを提案する．本稿では，バッファキャッシュアーキテクチャの検討を行う．

2 背景と関連研究

SSD は図 1 のような内部構成になっている．ホストシステムからの書き込み，読み出し要求はまず RAM のバッファキャッシュが受け取り，そこからデータが追い出されたり読み出しミスした場合にはその下の FTL へと要求が行く．FTL は，状況に応じて 3 つの命令 (read, write, erase) をフラッシュメモリに送る．フラッシュメモリは，4KB から 8KB 程度のページと，ページを 64 から 128 個程度まとめたブロックで構成される．フラッシュメモリは，読み書きをページ単位で行うことができるが，データの上書きはできないという特性を持つ．そのため，上書きする際には一度ブロック単位でデータを消去し再度データを書き込む必要があり，これには膨大な時間がかかる．また，ブロック消去は SSD の耐久性を下げる一番の要因となる．

SSD に搭載されるバッファキャッシュについては，幾つかの先行研究がなされている．BPLRU [1] は，2008 年に提唱された SSD 内バッファキャッシュのアルゴリズムで，主にランダム書き込み性能を高めることを目的としている．BPLRU は Block Associative Sector Translation (BAST) と呼ばれる Log-block FTL [2] の上に置かれることを前提とし，同じ SSD ブロックに所属するアドレスのデータを同じ LRU 順序で取り扱う．BAST は，同じ SSD ブロックに属する連続したデータの書き込みの際には，ブロック消去の回数を減らすことができるという特徴を持つ．BPLRU を用いることで，ランダムな書き込みをシーケンシャルな書き込みに変換して書き戻すことができるため，書き込み性能と耐久性を向上させることができる．

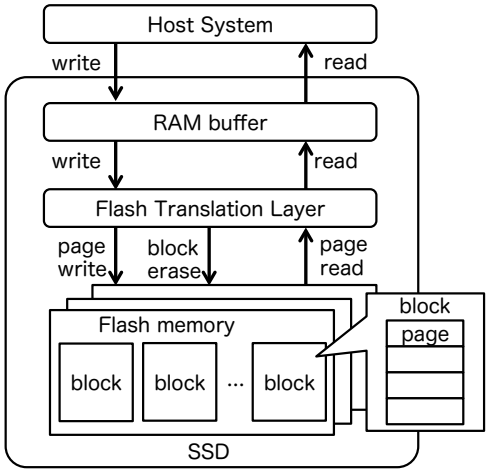

図 1: SSD の構成

PUD-LRU [3] は，2010 年に提唱されたアルゴリズムで，BPLRU を拡張し，どれだけ頻繁にアクセスされたかという指標を追加したものである．トランザクション処理など，幾つかの I/O トレースが強い時間的局所性を示していたことを考慮し，多くのアプリケーションで性能向上を達成した．

3 SSD の耐久性を高めるキャッシュアーキテクチャの検討

3.1 予備評価の環境

予備評価として，単純な LRU 方式のバッファキャッシュを実装し，2 つの I/O トレースにおける速度，ブロック消去の回数及びヒット率の評価を行った．評価には FlashSim [4] というイベント駆動の SSD シミュレータを用い，ページサイズ 4 KB, ブロックサイズ 512 KB の 1GB の SSD をシミュレートした．評価には，Financial1 [5] と MSNStorageCFS [6] という 2 つの I/O トレースを用いた．Financial1 は金融機関で用いられる OLTP アプリケーションの I/O トレースで，MSNStorageCFS は Microsoft の MSN サーバーの I/O トレースである．BAST FTL の手前に新たに LRU 方式の 2MB のバッファキャッシュを実装し，2 つの I/O トレースを入力として評価を行った．

3.2 評価結果

評価結果を表 1 に示す．表 1 における speedup はバッファキャッシュを入れていないときに比べた書き込

A Study of Cache Architecture to Enhance SSD Endurance

Eri OGAWA† and Kenji KISE†

†Graduate School of Information Science and Engineering Tokyo Institute of Technology

表 1: 2 つのトレースにおける LRU バッファキャッシュの挿入による性能向上

トレース	speedup	erase 削減率	ヒット率
Financial1	3.0	66.5 (%)	65.3 (%)
MSNStorageCFS	4.7	80.6 (%)	77.8 (%)

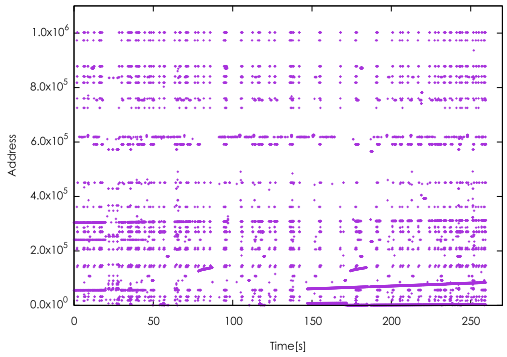

図 2: Financial1 トレースにおけるリクエストの系列

みの速度向上比を示し，erase 削減率はバッファキャッシュを入れていないときに比べたブロック消去回数の削減率を示す．

表より，速度向上，ブロック消去の削減率とバッファキャッシュのヒット率が比例していることと，Financial1 に比べ MSNStorageCFS の方が LRU による性能向上が大きいことがわかる．前者は，ヒット率が向上することでフラッシュメモリへの上書き回数が減り，ブロック消去の回数も減らせることと，コストの高いブロック消去を減らすことができると速度向上が得られることを示している．後者の理由を分析するため，Financial1 と MSNStorageCFS のアクセスパターンを調べた．

図 2 と図 3 は，Financial1 と MSNStorageCFS のリクエストの系列を，1 つのリクエストを 1 点としてプロットしたグラフである．グラフの横軸はリクエストのあった時間，縦軸はリクエストのアドレスを示し，それぞれ最初のリクエストから 1 万回分のトレースを抜き出してプロットしている．

図から，Financial1 は同じアドレスに頻繁にアクセスしており，そのアドレスは散発的であることがわかる．つまり，バッファキャッシュに入ってくるアドレスは間隔の開いた一定周期の系列である．バッファキャッシュには毎回違うアドレスが入ることになり，これは LRU には不利である．一方で MSNStorageCFS はランダムな系列であるため，Financial1 に比べれば LRU の恩恵が受けやすい．このことから Financial1 より MSNStorageCFS の方が LRU による性能向上が大きかったと考えられる．

3.3 考察

以上の結果から，LRU の効果はアプリケーションに依存することがわかった．既存のバッファキャッシュのアルゴリズムに当てはめると，LRU の効果が出にくい Financial1 のようなアプリケーションに関しては，PUD-LRU [3] にようにデータのアクセス頻度を考慮した置き換えアルゴリズムの方が性能向上が得られる

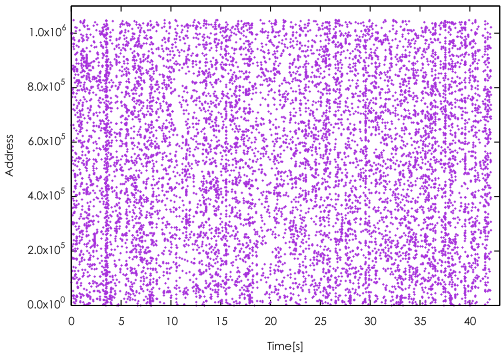

図 3: MSNStorageCFS トレースにおけるリクエストの系列

と考えられる．逆に，MSNStorageCFS のようなランダムな系列では BPLRU [1] のような LRU を用いたアルゴリズムの方が効果が出やすいと考えられる．つまり，既存の SSD 内バッファキャッシュのアルゴリズムはアプリケーションによる依存が大きいため，データ系列によって置き換えアルゴリズムを動的に変えるなどの工夫により，どちらの系列でも効果が出るのではないかと考察する．

4 まとめ

SSD の耐久性を高めるキャッシュアーキテクチャの検討のため，2 つのストレージの I/O トレースを分析した．その結果，既存の SSD 内バッファキャッシュのアルゴリズムはアプリケーションによる依存が大きいことがわかった．

今後の課題は，考察から得られた知見を生かしてアプリケーション依存をなくすための新たなバッファキャッシュのアーキテクチャを提案し，実装と評価を行うことである．

参考文献

[1] K.Hyojun and A.Seongjun. ”BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage”. In proceeding of the 6th USENIX Conference on File and Storage (FAST’08). 2008.

[2] J.Kim, J.Kim and Noh, S.H. and S.Min and Y.Cho. ”A space-efficient flash translation layer for CompactFlash systems”. IEEE Transactions on Consumer Electronics. vol.48. no.2. pp.366-375. 2002.

[3] J.Hu, H.Jiang, L.Tian and L.Xu. ”PUD-LRU: An Erase-Efficient Write Buffer Management Algorithm for Flash Memory SSD”. In proceeding of 2010 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS 2010). 2010.

[4] Y.Kim, B.Tauras, B.Urgaonkar. ”FlashSim: A Simulator For NAND Flash-based Solid-State-Drives”. In Proceeding of First International Conference on Advances in System Simulation (SIMUL’09). 2009.

[5] U.T.Repository. ”OLTP Application I/O”. http://traces.cs.umass.edu/index.php/Storage/Storage

[6] S.I.Repository. ”MSN Storage CFS”. http://iotta.snia.org/traces/