Flash メモリ向け DBMS 実装の提案

上野康平† 笹田耕一†

1. はじめに

リアルタイムウェブサービスの普及により、超高速 OLTP の需要は増加の一途を辿っている。この需要に対し、より高速なデータベースシステム (DBMS) のバックエンドストレージとして Flash メモリが注目されている。Flash メモリは、現在主流の HDD に比べて 1 万倍以上の IOPS 性能を有しており、これを活用することで、トランザクション処理を高速に行える DBMS の構築が期待できる。

しかし、既存の DBMS 実装を Flash メモリ上で運用した場合、その性能を活かしきれないという問題がある。一つは、Flash メモリの苦手とするランダム書き込みを多発する点である。Flash メモリでは、データの書き換えはブロックの消去、変更済みデータの再書き込みというプロセスを経て行われ、コストが高い。もう一つは、既存の実装を Flash メモリ上で動かした場合 CPU-bound になってしまう点である。既存の実装では低速な HDD を前提としているため、CPU を活用して書き込み数を削減する方策がとられているが、書き込みが高速な Flash メモリでは逆にボトルネックになってしまう。

我々は、Flash メモリの特性を活かし、その性能を最大限に発揮させるような DBM 実装を開発した。具体的には、固定サイズのシーケンシャル書き込み、及び CPU 負荷の軽いトランザクション処理を徹底するために、「後載せページ」「ログの多重化」「楽観的トランザクション制御」の 3 つの手法を用いた。

今回、我々は以上の手法を実際に DBMS として実装し、Flash メモリ上で予備評価を行った。その結果、既存の DBMS 実装と比較して、5 倍以上高速な結果を得ることができた。

2. 手　　法

2.1 後載せページ

後乗せページは、以前筆者らが提案した木構造に対してログ構造化手法を効率的に適応する手法である (図 1)[3]。

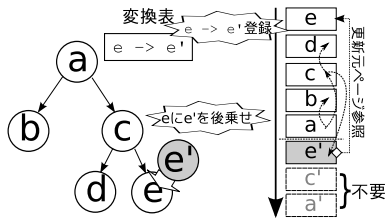

図 1 後乗せページを用いた木構造の更新．それぞれの円は節ページを示す．

シーケンシャル書き込みのみでデータ管理を行う手法として、Rosenblum ら[2] によるログ構造化手法がよく知られている。しかし、データベースで多用される木構造に用いた場合、木のルート付近のノードは参照先の葉ノードの更新により毎回書き直されることになってしまう。

後乗せページには、それがどのページに対する更新なのかという情報が埋め込まれており、これを変換表を用いて管理することにより、他のページからの参照関係を壊さずに更新することができる。これにより、木のルート付近のページ書き込みを抑制することができる。

2.2 ログ多重化

ログの多重化は、ページ更新情報以外のログを、ページ更新ログに織り込む手法である。データベース実装では、レコードの更新情報以外に、トランザクション状態やテーブルの統計情報といった情報も管理している。これらは、永続化のためにストレージ上に保存される。

しかし、これらの情報を別々のログで管理していると、状態の更新に伴う書き込みが分散してしまう。今回、ページ更新ログにこれらのログも埋め込むことで、これらのログへの書き込みを一括して行えるようにし、書き込みの分散を解決した。ページ更新ログの 1 エントリは 4K バイトで構成されるが、ここから 40 バイトの領域を設け、ページ更新以外の情報を管理する。

† 東京大学大学院情報理工学系研究科

表 1 ベンチマーク環境

CPU	Intel Xeon E5345 CPU x2 (8-way)
メモリ	8GB DDR2 FB-DIMM
OS	Linux 3.0.0
ストレージ 1	メインメモリ上の `tmpfs`
ストレージ 2	Intel X25-E 160GB SSD 上の `ext2`
ストレージ 3	HGST Deskstar 7K2000 x3(RAID0) 上の `ext2`

2.3 楽観的トランザクション制御

楽観的トランザクションは、書き込みを含むトランザクションを並行して行う手法である。

まず、並行性の高いトランザクション制御方法として、楽観的トランザクション制御[1] が知られている。ブロッキングを伴うロックを用いずに、並行にトランザクション処理を行い、最後のコミット時に他のトランザクションと衝突していないことを確認する。

楽観的トランザクション制御は、ロックを用いる悲観的トランザクション制御に比べて、競合が少ない場合のスループットは高いが、競合が起きた場合に無駄な IO が発生してしまうという欠点がある。しかし、Flash メモリは、非常に高い IO 性能を発揮するため、ロックを用いた悲観的トランザクション制御ではそのスループットを使い切れない。よって、Flash メモリを前提にしたシステムでは、競合トランザクションのアボートによる無駄な IO を考慮しても、楽観的トランザクションの方が高い性能を発揮する。

さらに、トランザクションの競合解決をロックフリーアルゴリズムで行うことで、ブロックすることなくトランザクションを実行することができる。これにより、CPU の利用効率が上がり、IO 性能の高い Flash メモリをストレージとして用いた場合でも処理性能が CPU により律速されることを防ぐことができる。

3. 評　　価

上で述べた手法を適用したデータベース実装を作製し、評価を行った。作製したデータベースは、Unix DBM 互換の Key-value Store 型のインターフェースを持ち、独自拡張としてトランザクションをサポートしている。実験に用いた環境を表 1 に示す。

本実装及び他の DBM 実装に対し、100 万レコードを昇順に挿入した場合のトランザクション (tx) 処理性能を計測した。それぞれのレコードは、4byte の key、8 バイトの value から構成されている。トランザクションの粒度は、1 レコードごと、100 レコードごとの 2 パターンを試した。また、ストレージは表 1 の 3 種を対象とした。ストレージ 2, 3 の計測では、それぞれトランザクション終了時にディスク同期を行うものとした。

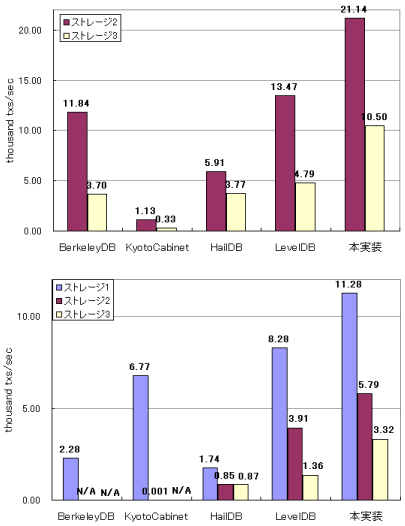

図 2 逐次書き込みを行うトランザクションの処理速度の比較
グラフ上: 1 レコード/tx グラフ下：100 レコード/tx
N/A は時間がかかりすぎて計測不可

計測結果を図 2 に示す。何れのベンチマークでも、本実装が最も高い性能を示していることがわかる。他実装の中で最も高い処理性能を持つ LevelDB と比較しても、ストレージ 2 の SSD 上で 1 レコード/tx のケースで 56%、100 レコード/tx のケースで 36%の速度向上が得られる。

4. おわりに

我々は、Flash メモリ向けの DBMS 実装を開発し、予備評価を行った。「後載せページ」「ログの多重化」「楽観的トランザクション制御」の手法を適用することで、既存の DBMS 実装に比べて高い性能を発揮することを示した。

今後は、より性能の高い SSD での評価や、より複雑なワークロードでの評価を行いたいと考えている。

参 考 文 献

1) H. T. Kung and John T. Robinson. On optimistic methods for concurrency control. *ACM Trans. Database Syst.*, Vol.6, pp. 213–226, June 1981.
2) Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Trans. Comput. Syst.*, Vol. 10, pp. 26–52, February 1992.
3) 上野康平, 笹田耕一. 後載せページによる効率的なログ構造化インデックス. 先進的計算基盤システムシンポジウム (SACSIS2011), 2011.