1L-05

並列ログ先行書き込み法における チェックポインティングの効率化

小見山 雄希[1,a)]　川島 英之[1,b)]

概要：本論文では並列ログ書込み先行方式において効率的なチェックポインティング方式として，分割統治方式を提案する．実験により，提案手法は従来手法より 52 倍程度優れた性能を示したことを報告する．

1. 背景

データ量の増加に伴いデータ集約型科学が発展しつつある．例えばすばる望遠鏡から得られる一晩 300GB 程度の観測画像を専用パイプラインシステム [1] で処理することで超新星爆発などの科学的発見が得られる．このような大規模データ処理ではトランザクション処理が使われることが多い．例えば天文データ処理ではファイルシステムのメタデータ管理で使われる．

トランザクション処理システムは並行性制御機構とログ永続化機構から構成される [2]．本研究では後者のログ永続化機構に注目し，ログ先行書込み法を研究対象とする．これは障害回復に必要な機構である．

トランザクション処理を高性能化するには，それが立脚するハードウェアの特性を引き出す必要がある．ログ永続化機構はログをストレージデバイスに書き込むため，ストレージデバイスの特性を引き出す必要がある．近年は不揮発性を有するメモリである NVRAM が脚光を浴びており，本研究ではこの NVRAM を前提とする．

2. 課題

NVRAM を前提としたログ先行書込み法においては，それを並列化する手法が提案されている [3], [4], [5]．これらの手法は NVRAM が並列 I/O による高性能化をもたらす点を活用するものである．文献 [3] では並列方式は既存方式である逐次方式に比べて 10 倍程度の高性能化をもたらすことが報告されている．

これらの研究で未解決課題として残されている点がチェックポイント機構である．これはリカバリ処理の高速化にために使われる機構である．ログ先行書込みプロトコルを用いる場合，システムクラッシュからの再起動後，ログを適用することによりデータベースの状態を復旧させる．もしもログが長大ならば，その適用時間は長大になる．チェックポイントは周期的に更新オブジェクトをストレージに書き出すことにより，この開始時点を短縮化する．この更新情報は dirty page table と transaction page table に記録される．

従来の逐次方式ならば，dirty table table へのアクセスに排他制御は不要だが，並列方式では複数のワーカースレッドが同時にエントリを更新するために排他制御が必要になり，これが性能劣化をもたらしうる．この排他制御問題の解決手法を 3 章で提案する．

3. 提案

dirty page table ならびに transaction page table に対する排他制御問題を解決するために，我々は分割統治法を提案する．排他制御問題が生じる理由は，それが集中的に管理されているからである．我々のアプローチは，上述の table をワーカースレッド毎に分散して管理させることで，問題解決を試みる．我々が知る限り，この問題にアプローチした既存研究は存在しない．

このアプローチを用いる場合，各スレッドの管理するデータを集約する必要がある．この集約に際して，write-writeconflict が生じる可能性がある．同一オブジェクトに対する複数の更新処理，即ち write-write が実行されたことが集約時に始めて観測される可能性がある．

ところが，これは我々の問題設定においては WW は問題にならない．なぜなら更新結果を read 操作がないからである．すなわち我々の扱う問題は read-modify-write ではなく blind-write となる．この場合には Thomas' write rule [2] を時間反転して適用することで，複数の write 操作

[1] 慶應義塾大学環境情報学部
Department of Environment and Information Studies, Keio University
[a)] t14349yk@sfc.keio.ac.jp
[b)] river@sfc.keio.ac.jp

Algorithm 1 集約スレッド

```
1: for wt ∈ WorkerThread do
2:     Lock dirty page table
3:     Aggregate dirty page table with Scan
4:     Unock dirty page table
5:     Lock transction page table with Scan
6:     Aggregate transction page table
7:     Unlock transction page table
8: end for
```

Algorithm 2 ワーカースレッド

```
1: Lock database objects
2: Insert a log to local buffer
3: Synchronize
4: Lock dirty page table
5: Update or Append entry in dirty page table
6: Unlock dirty page table
7: Lock transaction table
8: Update or Append entry in transaction table
9: Unlock transaction table
10: Unlock database objects
```

を最初の 1 つに縮退できる．この集約処理に整列を使えば $O(Nlog(N))$ 程度のコストを要するが，我々は single access を用いることで $O(N)$ での実現を行う．この集約スレッドの動作をアルゴリズム 1 に示す．

一方，ワーカースレッドはログ書込み，データベースアクセス，そして dirty/transaction page table の操作を行う．データベースアクセスとこれらのデータ構造へのアクセスにはロックを用いる．ログバッファと table へのアクセスにはロックを用いない．なぜならログバッファは占有されているからである．この処理をアルゴリズム 2 に示す．このプロトコルはログを永続化してからロックを解放する，いわゆる conservative lock release である [6].

4. 評価

提案手法を評価するために，並列ログ先行書込み法に基づくトランザクション処理システムを実装した．この方式は P-WAL [3], [4] に基づく．プログラム行数は 400 行程度であり，開発には C++言語を用いた．データオブジェクトに用いるロック機構は pthread_mutex_lock を用いた．並行性制御法は S2PL を用いた．実験はシングルノードにおいて実行した．CPU は 24 コアである．1 スレッドが実行するトランザクション数は 100，各トランザクションが実行するオペレーションは 100(update 99, commit 1) とした．データベースオブジェクトの数は 100 万である．NVRAM へのアクセスは遅延挿入により模擬した．遅延は 1μ 秒であり，これは NVDIMM を想定している．

実験結果を 1 に示す．この結果より，提案手法 (Propoesd) は従来手法である集中管理方式 (Single) に対して大幅な性能改善を示すことがわかる．性能差が最大となるのは 24 スレッドの時であり，その差は 52.65 倍である．この理由は table に対して集中管理方式では複数のワーカースレッドがひとつのオブジェクトに対するロックを取り合う一方，提案方式ではそのロック競合問題を解決しているからだと考えられる．一方，Proposed はスレッド数に対してスケールしない．この理由は書込み処理が多いためトランザクションが衝突するからだと考えられる．その根拠としてチェックポイント処理を一切行わない結果である ideal でも同様な現象が発現している．

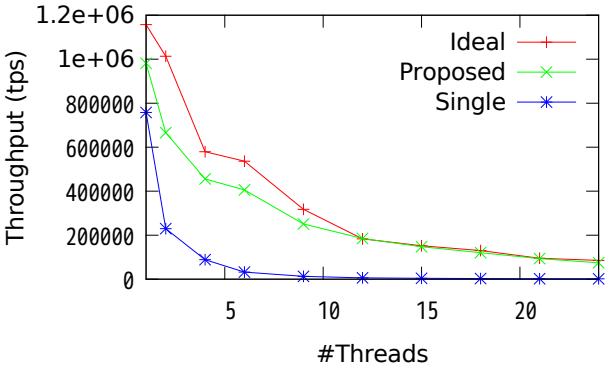

図 1 実験結果

5. 結論

トランザクション処理において必須機能であるログ永続化法は並列化により性能が大幅に向上される．一方，この並列化はチェックポイント処理の性能劣化という問題を引き起こす．本論文では並列ログ書込み先行方式において効率的なチェックポインティング方式として，分割統治方式を提案した．実験の結果，提案手法は従来手法より 52 倍程度優れた性能を示すことがわかった．

謝辞 本研究の一部は，JST CREST JPMJCR1303 and JPMJCR1414, KAKENHI JP17H01748, and project commissioned by NEDO による．

参考文献

[1] Tanaka, M., Tatebe, O. and Kawashima, H.: Applying Pwrake Workflow System and Gfarm File System to Telescope Data Processing, *IEEE CLUSTER*, pp. 113–122 (2018).
[2] Weikum, G. and Vossen, G.: *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*, Elsevier (2001).
[3] Kamiya, K., Kawashima, H., Hoshino, T. and Tatebe, O.: Parallel write-ahead logging method P-WAL, *IPSJ TOD*, Vol. 10, No. 1, pp. 24–39 (2017).
[4] Tu, S., Zheng, W., Kohler, E., Liskov, B. and Madden, S.: Speedy Transactions in Multicore In-memory Databases, *SOSP*, pp. 18–32 (2013).
[5] Wang, T. and Johnson, R.: Scalable Logging Through Emerging Non-volatile Memory, *PVLDB*, Vol. 7, No. 10, pp. 865–876 (2014).
[6] Nakamura, Y., Kawashima, H. and Tatebe, O.: Integrating Tic-Toc with Parallel Logging, *CANDAR CSA*, pp. 1 – 7 (2018).