非同期ロックマネージャによる決定論的並行性制御法の高速化

内田 克[1,a)]　川島 英之[1,b)]

概要：本研究では，非同期ロックマネージャを用いることで決定論的並行性制御法の高速化を実現するシステムを提案する．このシステムは，トランザクションの決定論的並行性制御手法 Determinism における逐次的ロッキングというボトルネックを排除した．本研究では，提案手法を評価するため，広く普及する SS2PL と従来のロック機構を持つ Determinism との比較を行う実験を行った．実験の結果，競合の多少に関わらず，提案手法が高い性能を示すことがわかった．

1. 動機

扱うデータ量が膨大になっている現代では，トランザクション処理は幅広く利用されており，その効率的な実現方式は多く探求されている．そのような方式の 1 つに Determinism [1] という方式がある．この方式は begin から end までの全てのデータベース操作が明らかであるという前提において，複数のトランザクション実行結果が必ず，特定の一つの serialization order に帰着するよう．決定論的に並行性制御を行う方式である．この方式は，例えば ATM での振り込みや引き出し，ブロックチェーンシステムにおけるデータアクセスで利用可能であり，アプリケーションは増加しつつある．本研究では，この Determinism に着目し，高速化の可能性を探求することを目標とした．

2. 課題

Determinism の性能ボトルネックは，トランザクションスケジューリングとロックマネージャにある．トランザクションスケジューリングの計算コストは小さく，無視可能な程度である．それに対して，ロックマネージャがもたらす性能劣化の影響は大きい．Determinism においては，2 つのトランザクションが $A \rightarrow B$ という順序でキューにあるとき，その順序に従って，それぞれのトランザクションはアクセスするレコードに対するロックを一括で排他的に獲得する．そのため，A と B が競合していなくても，B は A が全てのロックを獲得するまで実行を開始できない．このような問題は，pthread_mutex_lock などの，ロックマネージャがロック獲得をリアルタイムに実行する方式を採用すると生じる事態であり、深刻な性能劣化をもたらす．

3. 提案

Determinism におけるリアルタイム方式のロック獲得の問題を解決するために，本研究ではロック獲得のタイミングを遅延させ，非同期で行う．ロック獲得要求をロックマネージャに登録したあと，トランザクションはその要求が得られるまで待機を行う．ロック獲得が可能になったタイミングでトランザクションは再起動し，処理を遂行する．この方式により，競合しないトランザクションは同時に実行可能になる．この設計を実現するために，本研究ではロックマネージャを自作した．このロックマネージャにおいては，各レコードがトランザクション ID をキューの形式で保持する．トランザクションは開始時に，アクセスする全てのレコードのキューにトランザクション ID を排他的に追加する．キューの先頭にトランザクション ID があるトランザクションはレコードに対して，ロックの獲得を試みることができる．ロックの獲得に失敗すると成功するまでロックの獲得を試み，成功するとレコードのキューからトランザクション ID を削除する．全てのレコードに対してロックの獲得ができると read/write オペレーションに移行する．この方式により，トランザクションのロック獲得は並行して行いつつ，実行結果は特定の serialization order に帰着することが可能である．

4. 評価

提案を評価するために，非同期ロックマネージャを導入

[1] 慶應義塾大学環境情報学部
Department of Environment and Information Studies, Keio University
[a)] t20102mu@sfc.keio.ac.jp
[b)] river@sfc.keio.ac.jp

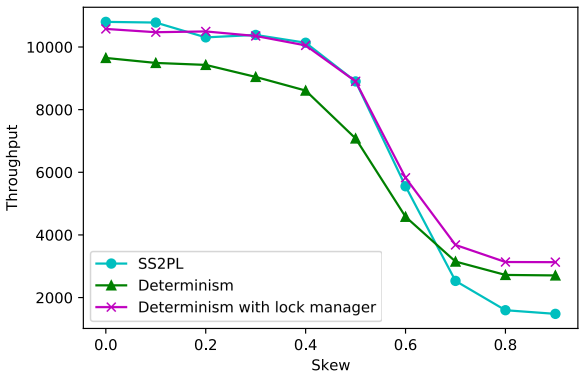

図 1 スループット

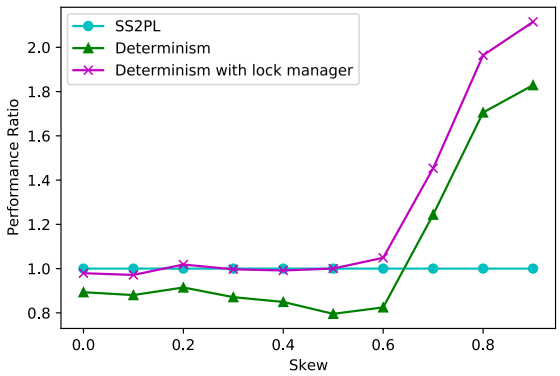

図 2 性能比

した Determinism に基づくトランザクション処理システムを実装した．開発には C++言語を用いて，近代的な並行制御手法を網羅的に分析可能なプラットフォーム ccbench [2] 上で実装し，世の中で幅広く使われている SS2PL と従来のロック機構を持った Determinism と比較し，評価した．CPU は 4 コアであり，スレッド数も 4 である．各トランザクションが実行するオペレーションは，それぞれ 50%の確率で read あるいは write の 99 個と，全体の 90 番目のオペレーションとして存在する 100 μ s の sleep の 1 個とした．データベースのオブジェクト数は 10 万である．

実験結果として得たスループットを図 1 に示す．この結果により，提案手法 (Determinism with lock manager) は従来のロック手法 (Determinism) に比べて，競合の発生頻度に関わらず，常に優れた性能を示すことが分かる．提案手法においては，競合しないトランザクション同士は並行して，ロックの獲得が可能であることが要因として考えられる．そのため，競合が発生しにくいときの方が，競合が頻発するときに比べ，提案手法と従来手法の性能差が大きい．また，SS2PL と比較すると，競合は発生しにくい場合は同等の性能を示し，競合は発生しやすい場合は提案手法の方が優れた性能を示す．SS2PL の性能を 1 として性能比を示す図 2 を見ると，競合が最も発生する場合には，提案手法の性能が SS2PL と比べ，約 2.1 倍優れている．SS2PL はデッドロックを避けるために，ロックの獲得に失敗するとトランザクションを abort し，リスタートする 2PL-NoWait 方式を採用している．そのため，競合が頻発する場合，トランザクションは abort を繰り返し，abort が発生しない Determinism と比較すると低い性能を示している．

5. 関連研究

Determinism は，分散データベースでも用いられている．Calvin [3] では，複数のトランザクションの実行結果が特定の serialization order に帰着するという Determinism の特性を活かし，分散データベースの実現において必要なレプリカ間の同期に一般的に用いられる 2 相コミットというボトルネックを排除し，高性能を実現している．

Aria [4] も Determinism の方式を採用した分散トランザクションプロトコルである．Aria では，データベーススナップショットに対してトランザクションを実行し，トランザクションコミット時にコミットすべきかどうかを決定論的に選択する．このことにより，全てのデータベース操作が事前に明らかである必要があるという Determinism の制約を取り除き，現実のワークロードに適応したデータベースシステムを実現している．

6. 結論

Determinism は特定の順番に従ってロックを獲得するため abort せず，競合が頻発する場合においては SS2PL のような Determinism ではないシステムより高い性能を示す．しかしながら，ロックの取得が逐次的に行われるというボトルネックを有している．本研究では非同期ロックマネージャの導入により，このボトルネックの解消を提案した．実験の結果により，提案手法は，競合の多少に関わらず，他の手法より高い性能を示すことがわかった．

謝辞 本研究の成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP16007）と科研費 (22H03596) の結果得られたものである．

参考文献

[1] A. Thomson and D. J. Abadi. The case for determinism in database systems. VLDB, 2010.
[2] Takayuki Tanabe, Takashi Hoshino, Hideyuki Kawashima, Osamu Tatebe. An Analysis of Concurrency Control Protocols for In-Memory Databases with CCBench. PVLDB, 13(13): 3531- 3544, 2020.
[3] A. Thomson, T. Diamond, S. C. Weng, K. Ren, P. Shao, and D. J. Abadi, “Calvin: Fast Distributed Transactions for Partitioned Database Systems,” in Proc. SIGMOD, ACM, 2012.
[4] Yi Lu, Xiangyao Yu, Lei Cao and Samuel Madden. Aria: A Fast and Practical Deterministic OLTP Database. PVLDB, 13(11): 2047-2060, 2020.