Wound精密化による2PLスケジューリング空間の拡大

角井 俊亮[1,a)]　星野 喬[2,b)]　川島 英之[1,c)]

1. 背景

データベースにおける並行性制御は，トランザクションを並行に実行した場合であっても直列化可能性を保証することができる．代表的な並行性制御プロトコルの一つとして，Two-Phase Locking（2PL）[1] がある．2PL の下では，トランザクションはアクセスするデータに対してロックを取得し，必要なロックをすべて取得し終えるまで取得したロックを解放しない．データがすでに他のトランザクションによってロックされていた場合には，そのロックが解放されるのを待つ．その性質上，複数のトランザクションが互いのロックの解放を待ち合い，実行を継続できなくなるデッドロックが発生しうる．デッドロックを事前に防ぐ代表的な手法の一つとして，wound-wait[1] がある．wound-wait では，トランザクションは開始時に一意なタイムスタンプ (ts) を取得しており，それに基づいてデッドロックを防ぐ．具体的には，トランザクション T_i が，トランザクション T_j がロックを保持するデータに対してロックを要求して競合した場合，T_i はタイムスタンプに基づき次のように振る舞う．

- $\mathrm{ts}(T_i) < \mathrm{ts}(T_j)$ ならば，T_i は T_j をアボートさせる．
- $\mathrm{ts}(T_i) > \mathrm{ts}(T_j)$ ならば，T_i は T_j がロックを解放するまで待つ．

デッドロックは，トランザクション間の待ちの関係がサイクルを形成することで発生する．サイクルが形成されるためには，あるトランザクションが自身よりタイムスタンプの大きいトランザクションを待つ関係が必要である．wound-wait では，$\mathrm{ts}(T_i) < \mathrm{ts}(T_j)$ のとき，T_i は T_j を待たずにアボートさせる．これにより，この関係が生じないため待ちのサイクルが形成されず，デッドロックが防止される．

[1] 慶應義塾大学
[2] サイボウズ・ラボ株式会社
[a)] 253984kadoi.shunsuke@keio.jp
[b)] hoshino@labs.cybozu.co.jp
[c)] river@sfc.keio.ac.jp

2. 問題

wound-wait は，あるトランザクションが自身よりタイムスタンプの大きいトランザクションを待つという，デッドロックの発生時には必ず成り立つ関係を排除することで，デッドロックを事前に防いでいる．しかし，この関係は 2 つのトランザクションのタイムスタンプのみで判定されるため，実際にデッドロックが発生している状況に比べて成り立つ範囲が広い．そのため，この関係が成り立つたびにアボートを行う wound-wait では，デッドロックにつながらない不要なアボートが発生する．アボートされたトランザクションは，それまでに実行した処理を破棄してトランザクションを再実行するというコストを払う必要がある．

3. 提案

3.1 概要

本研究では，wound-wait よりも限られた条件でトランザクションをアボートさせる，デッドロック防止プロトコルを提案する．これにより，デッドロックの発生を防ぎつつ，トランザクションのアボートを削減して性能を改善する．提案手法は wound-wait を基に拡張しているため，それぞれのトランザクションは開始時に他のトランザクションと重複しないタイムスタンプを取得することを前提とする．

3.2 デッドロック発生時に成り立つ条件

N 個（$N \geq 2$）のトランザクションの間でデッドロックが発生している状況を考え，その中で $\mathrm{ts}(T_k)$ が最小であるとする．このとき，T_k は T_k よりタイムスタンプの大きいトランザクションを待っており，かつ T_k よりタイムスタンプの大きいトランザクションが T_k を待っている．この構造はデッドロックが発生するために必要な条件であり，wound-wait がアボートを行う条件よりも成り立つ範囲が狭い．

© 1959 Information Processing Society of Japan

3.3 プロトコル

トランザクション T_i は，次の 2 つの条件がともに成り立つ場合にのみ，T_i が待っているトランザクションのうち，T_i よりタイムスタンプの大きいものをアボートさせる．

(1) T_i が，$\mathrm{ts}(T_i) < \mathrm{ts}(T_j)$ となるトランザクション T_j を待っている．

(2) $\mathrm{ts}(T_i) < \mathrm{ts}(T_k)$ となるトランザクション T_k が，T_i を待っている．

wound-wait は，条件 1 が成り立つ状況を生じさせないことで，デッドロックを防止している．これに対し，提案手法では条件 1 と条件 2 がともに成り立たない限り，トランザクションをアボートさせることはない．具体的には，T_i よりタイムスタンプの大きいトランザクションが T_i を待っていない限り，T_i は wound-wait であればアボートさせるはずのトランザクションをアボートさせずに，そのロックの解放を待つことができる．したがって，提案手法はデッドロックの発生を防ぎつつ，wound-wait に比べてデッドロックの発生に関与しないトランザクションのアボートを削減することができる．

4. 評価

評価にあたり，wound-wait 及び提案手法を並行性制御プロトコルを評価するためのプラットフォームである CCBench[2] を拡張して実装した．実験には，24 の物理コアを持つ Intel Xeon Gold 5418N を 2 基（計 96 論理コア）と，256 GB のメモリを搭載したサーバを用いた．評価には，Yahoo! Cloud Serving Benchmark[3] の Workload A（YCSB-A）を用いた．レコード数は 100 万，1 トランザクションあたりの操作数は 10，アクセスの偏りを表す Zipf 分布のパラメータ θ は 0.8 とした．

図 1 にスループットを，**図 2** にアボート率を示す．提案手法は wound-wait に対して，最大 65%のスループットの向上を達成した．また，提案手法は wound-wait に対して一貫してアボート率が低いことが確認できる．

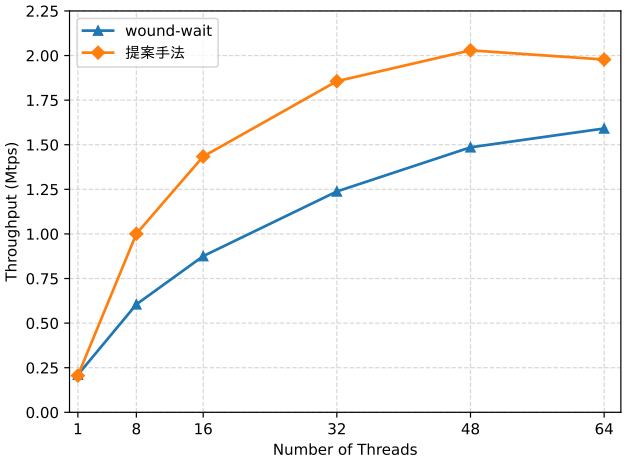


図 1 YCSB-A（$\theta = 0.8$）におけるスループット
Fig. 1 Throughput on YCSB-A ($\theta = 0.8$)

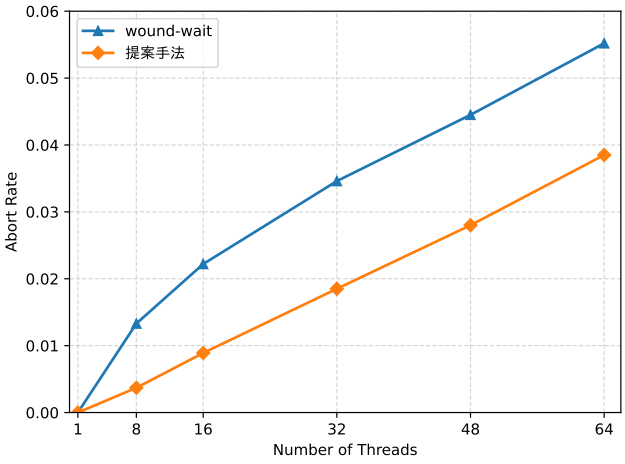


図 2 YCSB-A（$\theta = 0.8$）におけるアボート率
Fig. 2 Abort rate on YCSB-A ($\theta = 0.8$)

5. 結論

本研究では，wound-wait よりも限られた条件でトランザクションをアボートさせる，デッドロック防止プロトコルを提案した．これにより，デッドロックの発生を防ぎつつ，デッドロックの発生に関与しないトランザクションのアボートを削減し，性能を改善した．YCSB-A を用いた評価では，wound-wait に対して最大 65%のスループットの向上を達成した．

謝辞 本論文は，NEDO（国立研究開発法人新エネルギー・産業技術総合開発機構）の委託事業「ポスト 5G 情報通信システム基盤強化研究開発事業」（JPNP20017）ならびに JPNP25014，JSPS 科研費 25H00446，JST CREST JPMJCR24R4，セコム科学技術財団，JST COI-NEXT SQAI（JPMJPF2221），JST ムーンショット型研究開発事業 JPMJMS2215 から得られた成果に基づいています．

参考文献

[1] Silberschatz, A., Korth, H. F. and Sudarshan, S.: *Database System Concepts*, McGraw-Hill Education, New York, seventh edition (2020).

[2] Tanabe, T., Hoshino, T., Kawashima, H. and Tatebe, O.: An Analysis of Concurrency Control Protocols for In-Memory Databases with CCBench, *Proceedings of the VLDB Endowment*, Vol. 13, No. 13, pp. 3531–3544 (2020).

[3] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R. and Sears, R.: Benchmarking Cloud Serving Systems with YCSB, *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*, pp. 143–154 (2010).

© 1959 Information Processing Society of Japan