Slipstream：単一バージョン決定的並行制御における Mammoth 併走の正しさ回復と Blind Write 最適化

上田海翔[1]　川島英之[1]

1. 背景と課題

グラフデータベースでは，短いトランザクション（短 tx）と，広範囲を処理する長時間トランザクション（mammoth [3]）が混在する．mammoth を排他的に実行すると短 tx が待たされる．TuskFlow [1] は，決定的並行制御 Aria [2] を拡張し，mammoth の処理を実行単位（epoch）ごとの上限（budget）で分割して短 tx と併走させる．本稿は，TuskFlow を C++ で実装し，正しさを回復する C1 と，それに伴う性能低下を抑える C2 を提案する．

各 epoch の Phase 1 では，全 tx が同じ状態を読み（単一バージョン），アクセスを予約表に登録し，書き込みは手元に保持する．Phase 2 で識別子（TID）の小さい tx を優先して検証し，通った tx が書き込みを反映する．abort した短 tx は次の epoch で再実行する．直列化可能であるには各短 tx を mammoth の前か後に配置する必要がある．TuskFlow は，反映前の値が残る未処理のキー（UNVISITED）へのアクセスを「前」，mammoth が書き終えたキー（VISITED）へのアクセスを「後」と判定する．両方に触れる短 tx は straddle として abort する．

1.1 unsafe commit

しかし TuskFlow [1] では，mammoth は Phase 1 の予約時にキーを VISITED にし，値は Phase 2 で反映するため，同 epoch 中は VISITED でも反映前の値が残る．**図 1** では $x = y = 0$ を両方 1 に更新する．短 tx は前 epoch で反映済みの $x = 1$ と現 epoch で mammoth が $y \leftarrow 1$ を予約したが未反映の $y = 0$ を読む．両方 VISITED で「後」と commit するが，これは mammoth の前 $(0, 0)$ にも後 $(1, 1)$ にも一致しない直列化不能な commit（unsafe commit）である．また，mammoth を書き手に登録しないため，短 tx との書き込み衝突（WAW）を見逃すことにもなる．

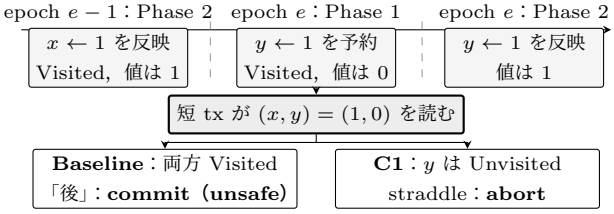


図 1　反映の前後にまたがる読みと Baseline・C1 の判定．
Fig. 1　Unsafe commit in Baseline and its prevention by C1.

1.2 blind write の不要な abort

WAW を検出すると，mammoth や TID の小さい短 tx と同じキーを書く短 tx は abort する．値を読まない上書き（blind write）でも同じである．しかし Phase 2 に読み取りはないため，読み取りの依存と矛盾しなければ，直列順序で最後の値だけを残して各 tx を commit でき，WAW abort は不要である．

2. 提案手法

2.1 C1: strict reservation

mammoth の書き込み予約時に，TuskFlow が想定する最小 TID [1] を書き手として記録し，WAW を検出する．同 epoch 中はキーを UNVISITED，反映後は VISITED と扱い，mammoth との前後は TID でなくこの状態で決める．図 1 の短 tx は x と y にまたがる straddle として abort する．一方，mammoth が予約中のキーと UNVISITED のキーだけに触れる短 tx は，Baseline では straddle で abort されたが，C1 では「前」として commit できる．

2.2 C2: blind write arbitration

C2 は，mammoth と短 tx による同一プロパティへの blind write を対象とする．C1 の前後判定と読み取りの衝突検査を維持し，WAW 単独での abort を省く．ただし，

[1] 慶應義塾大学
Keio University

© 2026 Information Processing Society of Japan

表 1　正しさの検査結果（5 回の中央値［最小–最大］）．
Table 1　Correctness check (median [min–max] of 5 runs).

構成	unsafe 率（%）	unsafe 件数	閉路あり
Baseline	25.75［24.89–33.48］	4,611	5/5
C1	0［0–0］	0	0/5
C1+C2	0［0–0］	0	0/5

TID の小さい tx に対して，その tx が書くキーを読む関係（RAW）と WAW を併せ持つ tx は保守的に abort する．書き込み前の値を読んだ tx をその tx より前に置く順序と，書き込みを残す順序が矛盾しうるためである．

検証を通った短 tx 間では，ノードのロック下で（epoch, TID）を比較し，キーごとに最大の書き手の値を残す（Thomas write rule）．同じ epoch に mammoth が書くキーでは，mammoth の値を残して短 tx の書き込みを省く．C1 が短 tx を「前」に置くため，短 tx の後に mammoth が上書きする直列実行と最終値が一致する．よって検証を通った短 tx は，書き込みの一部を省いて commit できる．

3.　評価

3.1　実装と条件

Baseline（C1・C2 未適用），C1，C1+C2 を比較する．3 構成とも，グラフタグ付け [1] で状態判定を mammoth の更新対象に限定し，短 tx は mammoth と同じプロパティに書き，再実行時は値を読み直す．Xeon Gold 5416S の物理 16 コア（SMT 有効，論理 32 コア）にプロセスを固定した．データは LDBC SNB SF3（頂点 1,064 万，辺 4,910 万）である．短 tx は Person を一様に選び，隣接辺を 9 回選択する．選んだ辺がメッセージの作成者を表すとき，そのメッセージを読むか blind write する．書き込み比率は，短 tx に占める更新 tx の割合である．mammoth は全メッセージ（1,033 万）を blind write する Q0 を 32 タスクに分割し，budget は epoch あたり合計 200 万更新とした．mammoth は開始 60 秒後に投入し，実験時間は 90 秒とした．worker は 32，epoch あたり最大 5 万 tx とし，各条件を 5 回実行して性能値の中央値を示す．

3.2　正しさと性能

投入負荷 1 万 tx/s，書き込み比率 50% で，commit した全 tx が読み書きした各ノードの版番号を記録し，依存グラフの閉路の有無で衝突直列化可能性を検査した．unsafe 率は，mammoth 併走中に mammoth が書くノードを読んで commit した短 tx のうち，反映前後の値を両方読んだ割合とする．**表 1** のとおり，Baseline は全 5 回で閉路が生じ，約 4 分の 1（中央値 25.75%）が unsafe commit であった．C1・C1+C2 は全実行で unsafe・閉路が 0 であった．

性能は，投入負荷 8 万 tx/s，書き込み比率 20・50・80% で，mammoth 併走中の最初と最後を除く epoch の平均スループットを比較した（**図 2**）．Baseline は直列化可能でないため参考値である．C1 は Baseline より 17–34% 低く，新たに検出する mammoth–短 tx 間の WAW abort（書き込み比率 80% で約 4.6 万件／実行）が一因と考えられる．C1+C2 では WAW 単独での abort が全実行で 0 となり，スループットは各書き込み比率で順に C1 の 1.11・1.24・1.46 倍，Baseline の 92・86・95% に達した．Baseline との差は epoch あたりの straddle abort の増加による．C1 は，Baseline が「後」として commit していた，反映済みと予約中のキーにまたがる短 tx を abort するためである．

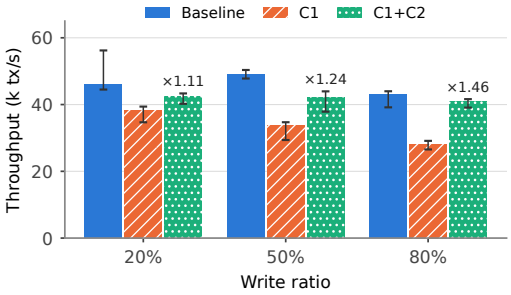


図 2　mammoth 併走中の短 tx スループット（投入負荷 8 万 tx/s，5 回の中央値と最小–最大）．数値は C1+C2 / C1.
Fig. 2　Short-tx throughput (80,000 tx/s offered; median and min–max of 5 runs). Labels: C1+C2 / C1.

4.　まとめ

C1 による unsafe commit の防止と，C2 による blind write の abort 削減を提案した．C++ 実装での LDBC SNB SF3 Q0 評価では，C1・C1+C2 で unsafe と閉路はなく，C1+C2 のスループットは C1 の 1.11–1.46 倍であった．

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] Theodorakis, G., Firth, H., Clarkson, J., Crooks, N. and Webber, J.: TuskFlow: An Efficient Graph Database for Long-Running Transactions, *Proc. VLDB Endow.*, Vol. 18, No. 12, pp. 4777–4790 (2025).
[2] Lu, Y., Yu, X., Cao, L. and Madden, S.: Aria: A Fast and Practical Deterministic OLTP Database, *Proc. VLDB Endow.*, Vol. 13, No. 12 (2020).
[3] Cheng, A., Waudby, J., Crooks, N., Firth, H. and Stoica, I.: Mammoths Are Slow: The Overlooked Transactions of Graph Data, *Proc. VLDB Endow.*, Vol. 17, No. 4, pp. 904–911 (2023).

© 2026 Information Processing Society of Japan