高性能かつ耐故障なデータベースシステムの設計

宮崎 祐介[1,a)]　中園 翔[2,b)]　中森 辰洋[1,c)]　李 浩文[1,d)]　川島 英之[1,e)]

概要：本研究では，高スループットと高可用性を同時に満たす MySQL 互換データベースマネージメントシステムの実現を目的とし，従来の InnoDB ストレージエンジンが抱える並列実行性能および対故障性付与時の性能劣化という課題に取り組む．InnoDB は 2 相ロック (2PL) に基づく並行性制御を採用しており，多コア環境におけるスケーラビリティが限定的であり，最新のトランザクション処理手法である Silo 系手法と比較して性能面で劣ることが報告されている．また MySQL の高可用構成である InnoDB Cluster (Group Replication) では，障害耐性を得る代償として書き込み性能がおよそ半減し，高性能と耐故障性の両立が困難であるという問題がある．これに対し本研究では，MySQL 互換インタフェースを維持したまま，InnoDB を新規ストレージエンジン LineairDB へ置き換えるとともに，複製機構としてバルク転送法を取り入れた Raft レプリケーションを統合したシステム設計を提案する．LineairDB はエポック同期型グループコミット（Epoch-based Group Commit; EBGC）に基づく並行性制御を採用し，ロック競合を回避しつつトランザクションの並列実行を最大化する．初期評価では，16 スレッド環境において LineairDB は InnoDB に対し約 5 倍のスループットを示した．

1. はじめに

オンラインサービスの大規模化に伴い，DBMS のトランザクション処理には高い並列性能が求められる．2020 年の独身の日には Alibaba 社の注文件数が毎秒 583,000 件に達した [1]．広く用いられる MySQL はプラガブルなストレージエンジンを採用する一方，デフォルトの InnoDB では，競合する書き込みがロック獲得・解放により処理され [2]，多コア環境でロックやキャッシュラインの競合が顕在化するためにスループットが頭打ちになりやすい．さらに高可用性構成（MySQL InnoDB Cluster / Group Replication）では，同期複製により可用性は確保できるが，書き込み性能は対故障性を付与しない場合と比べ大きく低下する． [3].

本研究は「高い並列性」と「高可用性」を両立する MySQL 互換 DBMS の設計を目的とし，(i) InnoDB を最新のストレージエンジン **LineairDB**[*1] [4] に置き換え，(ii) LineairDB 上で Raft に基づく合意とバルク転送法を用いたレプリケーション [5] を統合する設計を提案する．

[1] 慶應義塾大学
[2] LINE ヤフー株式会社
[a)] miyayu@keio.jp
[b)] shnakazo@lycorp.co.jp
[c)] tatsuhironm@keio.jp
[d)] tony_li.haowen@keio.jp
[e)] river@sfc.keio.ac.jp
[*1] https://github.com/LineairDB/LineairDB

2. 設計アプローチ

2.1 LineairDB ストレージエンジン統合

LineairDB はエポック同期型グループコミット（Epoch-Based Group Commit; EBGC） [6], [7] を中核とする並列性を持つストレージエンジンである．EBGC は，実時間を一定の時間ごとに区切ったエポックに分割し，同一エポック内の複数トランザクションのコミットを終端で一括実行する．これにより，(i) トランザクション群に安定した半順序を与え，(ii) システムが安全な一貫状態を周期的に獲得できる同期点として機能する．

図 1 に示すように我々が提案する設計では，ストレージエンジンがプラガブルであるという MySQL の特性を利用し，MySQL のインターフェース，SQL パーサやオプティマイザ等の上位レイヤを維持したまま，InnoDB の代替として LineairDB を組み込む．

2.2 並行性制御法

LineairDB は，Silo [6] に基づく楽観的並行性制御法（Optimistic Concurrency Control; OCC）を採用している．各トランザクションは実行中にロックを取得せずに読み取り・更新を行い，コミット時の検証フェーズで一貫性と競合を確認することで直列化可能性を保証する．この方式では，ロック取得の競合が生じにくく，多数スレッド環境において高いスループットが期待できる．

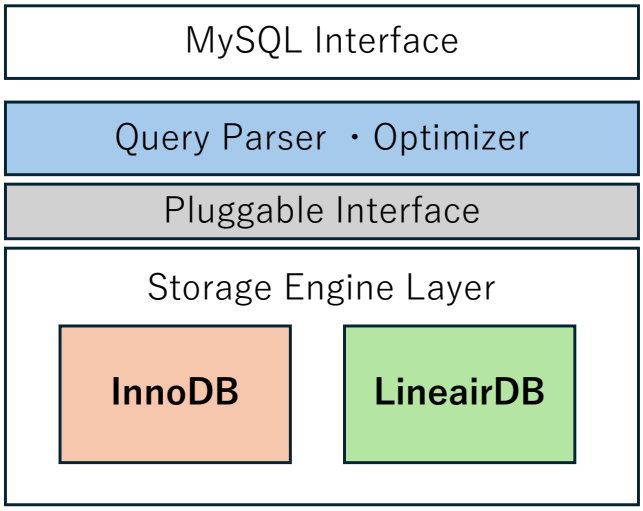

図 1: 設計の概要

2.3 高可用化のための Raft レプリケーション統合

高可用性のために，更新ログを Raft 合意プロトコル [8] で複製する．Raft ではリーダが更新ログを順序付けてフォロワに配布し，多数派の合意を得た更新のみをコミット済みとすることで，リーダ障害時にも再選出によりサービス継続を実現する．本設計では，複数の更新をまとめて一括転送するバルク転送法 [5] を採用し，往復遅延や I/O 同期の回数を削減する．

2.4 実装例

LineairDB を MySQL に統合するにあたり，ストレージエンジン用ハンドラ (handler) 層を実装し，LineairDB の楽観的トランザクション処理基盤と MySQL サーバ機能を接続した．本稿では特に，複合キーのシリアライズ方式とレンジ検索対応を例に開発内容を述べる．LineairDB の handler では，MySQL が渡す複合キーを自己記述的なバイト列へ変換し，LineairDB トランザクションでそのまま辞書順比較できるようにしている．各カラムは [NULL マーカ][型タグ][長さ 2byte][payload] にエンコードされ，serialize_key_from_field() が型ごとの正規化を担う．整数はリトルエンディアンからビッグエンディアンに変換したうえで符号ビットを反転し，DATETIME は MySQL 標準の sortable 形式をそのまま利用，文字列は実データのみを payload として連結する．これによりキーパートを順番に連結したバイト列の辞書順が SQL の複合キー順序と一致し，範囲検索は start_key と end_key を生成して Scan するだけで実現できる．プレフィックス検索時には，build_prefix_range_end がプレフィックスに 0xFF を付加した上限キーを合成し，同じルールで Scan に渡す．

3. まとめ

本研究では，MySQL 互換を維持しつつ，エポック同期型グループコミットを用いた LineairDB と，バルク転送を備えた Raft の統合により，高性能かつ高可用性を備えたデータベースシステム設計を提案した．現在，提案システムのプロトタイプ実装を進めており，MySQL サーバ上での LineairDB ストレージエンジン統合を段階的に行っている．特に，複合キーのシリアライズ方式や範囲検索処理など，MySQL 互換性を保ちながら主要なクエリを実行可能にするための機能を実装中である．今後は，実装の完成とベンチマーク評価を通じて，提案手法の有効性を定量的に示す予定である．

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) および JPNP16007，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 の支援を受けたものです．

参考文献

[1] Cloud, A.: Alibaba Cloud Supported 583,000 Orders/Second for 2020 Double 11 - The Highest Traffic Peak in the World!, https://www.alibabacloud.com/blog/alibaba-cloud-supported-583000-orderssecond-for-2020-double-11---the-highest-traffic-peak-in-the-world_596884,

[2] MySQL: MySQL 8.0 Reference Manual: Alternative Storage Engines, https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html.

[3] GMO インターネットグループ グループ研究開発本部: MySQL InnoDB Cluster（Group Replication）と非同期・準同期レプリケーションの書き込み性能の比較, https://recruit.gmo.jp/engineer/jisedai/blog/mysql-innodb-cluste-performance/ (2022).

[4] 中園 翔, 別所祐太朗, 中森辰洋: LineairDB: エポック同期を活用したトランザクショナルストレージエンジン，コンピュータ ソフトウェア， Vol. 41, No. 1, pp. 1_15–1_35（オンライン），DOI: 10.11309/jssst.41.1_15 (2024).

[5] Yamashita, A., Tanaka, M., Bessho, Y., Fujiwara, Y. and Kawashima, H.: Improving Raft Performance with Bulk Transfers, *11th International Symposium on Computing and Networking Workshops (CANDARW 2023)*, Institute of Electrical and Electronics Engineers (IEEE), pp. 38–44 (online), DOI: 10.1109/CANDARW60564.2023.00015 (2023).

[6] Tu, S., Zheng, W., Kohler, E., Liskov, B. and Madden, S.: Speedy transactions in multicore in-memory databases, *SOSP*, pp. 18–32 (2013).

[7] Chandramouli, B., Prasaad, G., Kossmann, D., Levandoski, J. J., Hunter, J. and Barnett, M.: FASTER: A Concurrent Key-Value Store with In-Place Updates, *SIGMOD Conf.*, pp. 275–290 (2018).

[8] Ongaro, D. and Ousterhout, J. K.: In Search of an Understandable Consensus Algorithm, 2014 USENIX Annual Technical Conference (USENIX ATC ' 14), Philadelphia, PA, USENIX Association, pp. 305–319 (2014).