高性能で耐故障な MySQL の開発
- Kamo： メニーコア時代の次世代データベース -

1　背景

データベースは，メッセージアプリ，E-commerce サイト，銀行など，社会を支える多様なサービスの基盤として広く利用されている。中でも MySQL は，高いシェアを持つリレーショナルデータベースであり，小規模なサービスから大企業のシステムまで幅広く用いられている。

近年，サービスの発展に伴って，データベースに求められる処理性能は増大している。加えて，災害時やアクセス集中時にも安定して動作し続けることが重要であり，高負荷への対応と継続的なサービス提供の両立が強く求められている。特に現代の計算機環境では CPU のマルチコア化が進んでおり，その性能を活かせるデータベースの実現が重要である。

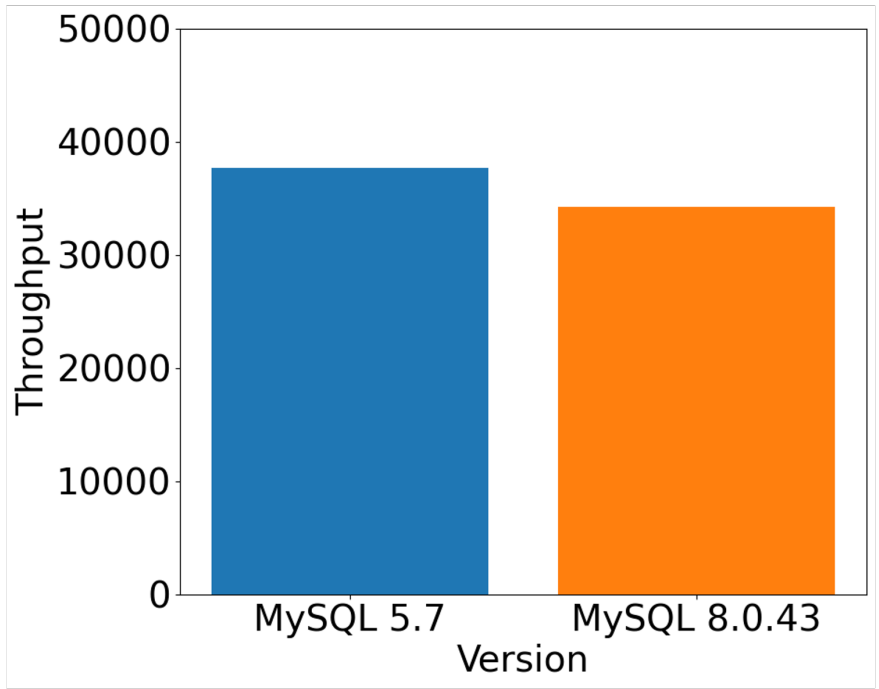

図 1: MySQL の性能の歴史

一方で，MySQL は 1990 年代に設計された基盤を引き継いでおり、マルチコア環境への適応が十分とは言えない。本プロジェクトで実施した調査では 2025 年リリースの MySQL8.0.43 は 2015 年リリースの MySQL5.7 と比較して性能がわずかに低下しており、約 10 年間のバージョン更新を経てもマルチコア環境における性能は改善されていないことが確認された（図 1)。したがって，既存アプリケーションとの互換性を維持しながら，高性能化と耐障害性を両立する MySQL 互換 DBMS を実現することには大きな意義がある。

2　目的

本プロジェクトの目的は，MySQL の互換性を維持したまま，既存の MySQL より高性能で，かつ耐故障性を備えた DBMS を実現することである。そのために，MySQL で標準的に用いられる InnoDB の代わりに，近代的な並行性制御によりメニーコア環境で高いスケーラビリティを持つ LineairDB を採用し，これを MySQL のストレージエンジンとして統合した Kamo を開発する。

Kamo においては，MySQL 標準の準同期レプリケーションをサポートし，Failover による可用性の付与も行うことで，性能向上に加えて障害時にもサービス継続が可能な MySQL 互換 DBMS の実現を目指した。

3　開発の内容

本プロジェクトでは，LineairDB を MySQL のストレージエンジンとして組み込んだ Kamo を実装した。しかし，LineairDB はもともと高性能な Key-Value ストアとして設計されており，そのままでは RDBMS である MySQL のストレージエンジンとして利用できない。そこで本プロジェクトでは，大きく分けて三つの開発に取り組んだ。

第一に，LineairDB 自体の機能拡張を行った。プロジェクト開始時点の LineairDB は，`Read`，`Write`，`Scan` といった基本 API を備えていた一方で，RDBMS に必要なテーブル管理機構や `Secondary Index` を持っていなかった。そこで，複数テーブルを明示的に扱うための `Table` 機構を実装するとともに，`Insert`，`Update`，`Delete` などの操作を追加した。さらに，主キー以外の条件による検索を可能にする `Secondary Index` を実装し，その更新・削除・走査において整合性が保たれるよう拡張した。これにより，LineairDB を単なる Key-Value ストアから，RDBMS のストレージエンジンとして利用可能な基盤へ発展させた。

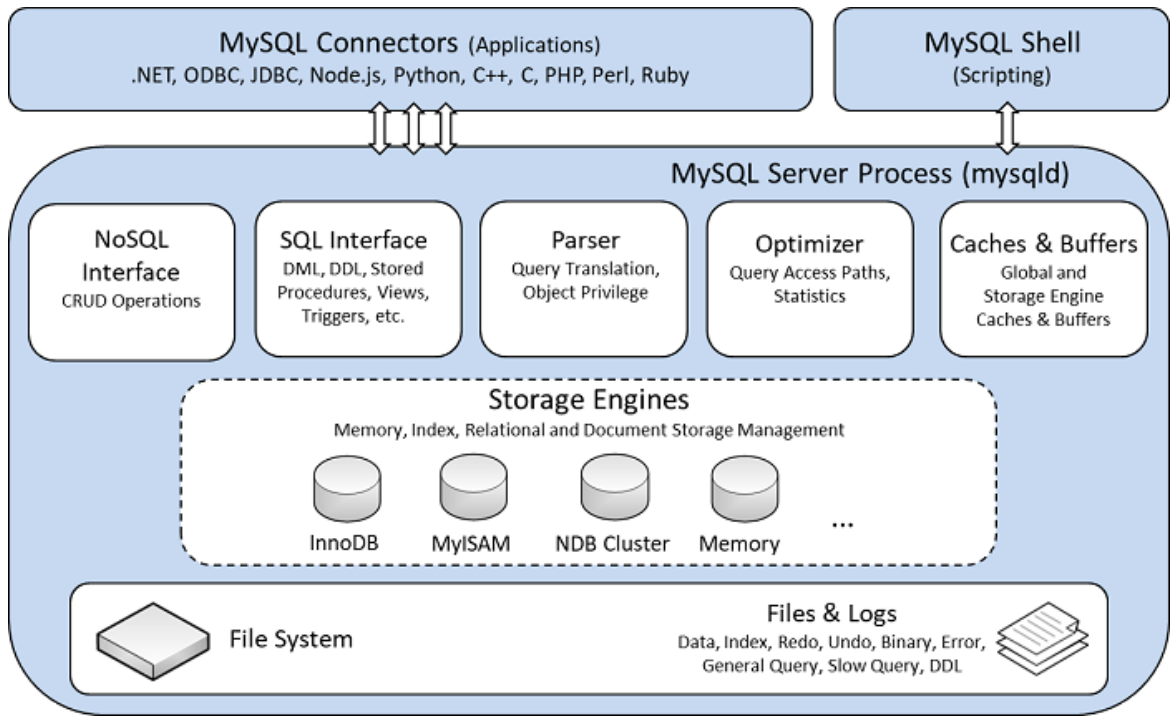

図 2: MySQL のプラガブルストレージエンジンアーキテクチャ（出典：MySQL Reference Manual `https://dev.mysql.com/doc/refman/8.0/ja/pluggable-storage-overview.html` ）

第二に，MySQL と LineairDB を接続する handler を実装した。MySQL は，サーバ層とストレージエンジン層が分離された構造を持ち，SQL の実行結果は handler インターフェースを通じてストレージエンジンに伝達される（図 2)。そのため，Kamo を実現するには，MySQL から受け取ったテーブル操作やレコード操作を LineairDB の API に変換する handler が必要であった。本プロジェクトではこの handler を実装することで，MySQL 互換の SQL インターフェースを維持したまま，内部では LineairDB による高速なトランザクション処理を利用できるようにした。

第三に，耐障害性と可用性を付与するための構成を実装した。当初は Raft の導入も検討したが，MySQL 本体への大規模な改変が必要であり，開発期間内での実現可能性を踏まえて方針を変更した。最終的には，MySQL 標準の Semi-Synchronous Replication を採用し，さらに Orchestrator による Primary 障害の自動検知と Replica の昇格，ProxySQL による接続先制御を組み合わせることで，自動 Failover を実現した。これにより，障害発生時にもサービスを継続可能な構成を整備した。

以上の開発により，Kamo は MySQL 互換のインターフェースを維持したまま，業界標準の OLTP ベンチマークである TPC-C ワークロードを実行可能なシステムとなった。TPC-C を実行できることは，単に基本的な SQL 処理が可能であることにとどまらず，複数テーブル，更新処理，索引アクセスを含む実用的なトランザクション処理を支えられることを示すうえで重要である。しかしながら、機能が備わっていても性能が高くなければ意味がない。そのため、開発した Kamo について，TPC-C ワークロードを用いて既存の MySQL と性能比較を行った。

実験環境は表 1 に，また可用性を付与するための構成は表 2 に示す。これらの構成のもとで評価を行った結果を図 3 に示す。結果から，Kamo は分散環境下において既存の MySQL に対して最大 18.7 倍の性能を示すことがわかった。

表 1: 実験環境

項目	内容
構成	DB ノード 3 台（Primary1 台 +Replica2 台，同一スペック）
CPU	AMDEPYC9654P96-CoreProcessor（192vCPU）
メモリ	1.0TiB
ストレージ	1TB（実効容量 993GB）
OS	Ubuntu22.04.5LTS
カーネル	Linux5.15.0-119-generic
仮想化基盤	KVM（仮想マシン）
レプリケーション方式	MySQL Semi-Synchronous Replication

表 2: ProxySQL ノードの構成

項目	仕様
CPU	IntelXeonGold6212U@2.40GHz（20vCPU）
メモリ	31GiB
ストレージ	約 500GB（実効容量約 405GB）
OS	Ubuntu22.04.5LTS
カーネル	Linux5.15.0-121-generic

4 従来の技術との相違

従来の MySQL は，成熟した互換性と豊富な利用実績を持つ一方で，マルチコア環境におけるスケーラビリティには改善の余地がある。本プロジェクトは，その MySQL を置き換えるのではなく，プラガブルストレージエンジン機構を活用することで，MySQL 互換のまま高性能化を図った点に特徴がある。

また，単に LineairDB を接続しただけではなく，RDBMS に必要な `Table`，`Insert/Update/Delete`，`Secondary Index`，走査機構などを新たに実装し，MySQL の handler と接続することで，SQL から透過的に利用できるようにした。すなわち，本プロジェクトの新規性は，高性能な Key-Value ストアを MySQL 互換 DBMS のストレージエンジンとして成立させるために必要な機能群を実装した点にある。

さらに，耐障害性についても，既存の MySQL レプリケーション機構に ProxySQL と Orchestrator を組み合わせることで，自動 Failover を含む実運用可能な構成を実現した。これは単なるベンチマーク用試作にとどまらず，高性能性と可用性の両立を目指した実践的なシステム開発である。

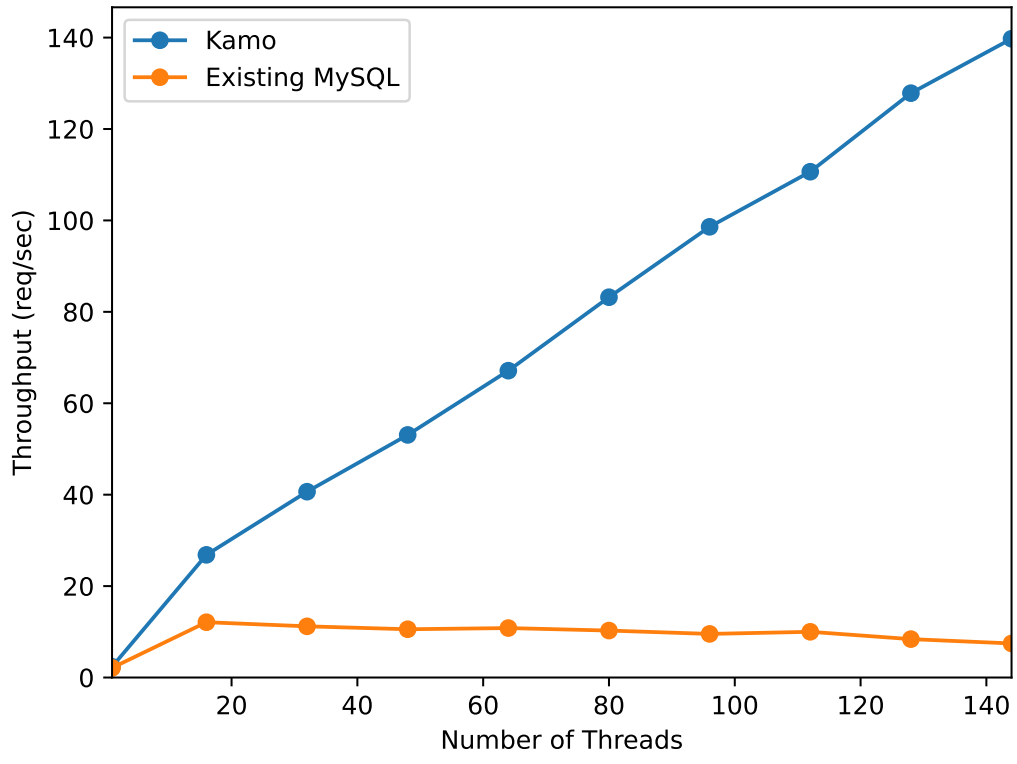

図 3: ProxySQL 構成における TPC-C ワークロードでの Kamo と既存の MySQL の比較

5 期待される効果

本プロジェクトの成果は，MySQL 互換の SQL インターフェースを維持したまま，高性能化と耐障害性の向上を両立できる点に意義がある。MySQL は広く利用されている主要な DBMS であり，その互換性を保ったまま性能向上が可能になれば，既存アプリケーション資産を活かしつつ導入できるため，Web サービス，E-commerce，金融系システム，企業情報システムなど，MySQL を利用する幅広い分野への波及が期待される。

技術的効果としては，既存の MySQL では十分に活かしきれなかったマルチコア環境での高並列処理性能を，引き出せる可能性がある。実際に，本プロジェクトで開発した Kamo は，192 vCPU・3 台構成の評価環境において，TPC-C ワークロードで既存の MySQL に対して最大 18.7 倍の性能を示しており，高負荷な OLTP 処理を扱うシステムにおいて，処理能力の向上やサーバ資源の有効活用に寄与することが期待される。

また，可用性の面では，MySQL 標準の Semi-Synchronous Replication に加え，ProxySQL および orchestrator を組み合わせることで，障害検知，接続先切替，Primary 昇格までを含む自動 Failover 構成を実現している。これにより，障害時にもサービス継続が求められる実運用環境への適用可能性を高める効果がある。

さらに，本プロジェクトは，高性能な Key-Value ストアを MySQL 互換 DBMS のストレージエンジンとして成立させるための技術蓄積という点でも意義がある。今後の高性能 DBMS，MySQL 拡張技術，高可用データベース基盤の研究開発を活性化することが期待される。

6 普及の見通し

本プロジェクトで開発した Kamo は，MySQL 互換の SQL インターフェースを維持しつつ，高性能化と耐障害性の向上を目指した DBMS である。そのため，既存の MySQL を利用している開発者や運用者にとって，比較的導入を検討しやすい構成になっている。特に，高負荷な OLTP 処理を扱う Web サービス，E-commerce，

業務システムなどにおいて，既存アプリケーション資産を活かしたまま性能改善を図る手段として普及することが期待される。

また，本プロジェクトで得られた成果は，単なる試作にとどまらず，高性能な Key-Value ストアを MySQL 互換 DBMS のストレージエンジンとして成立させるための実装知見を含んでいる。そのため，データベースシステムや高性能トランザクション処理の研究分野においても，今後の研究開発の基盤として活用される可能性がある。特に，MySQL の互換性を保ちながら内部実装を刷新するという方向性は，既存 DBMS の性能改善や高可用化を検討する上で有用な事例になると考えられる。

今後は，OSS としての公開に向けて，導入方法や制約事項を含むドキュメント整備を進めるとともに，性能評価結果や運用手順を整理し，利用しやすい形で提供していく予定である。さらに，学会や技術コミュニティでの発表を通じて認知を広げることで，研究用途に加えて，実サービス基盤への適用可能性も高めていきたい。将来的には，性能と可用性の両立が求められるシステムにおいて，本成果を発展させた技術が広く利用されることを目指している。

7　クリエータ名（所属）

- 宮﨑 祐介（慶應義塾大学 総合政策学部 総合政策学科）
- 中森 辰洋（慶應義塾大学 大学院政策・メディア研究科）
- 李 浩文（慶應義塾大学 大学院政策・メディア研究科）

（参考）関連 URL

- ソースコード：https://github.com/mitou-Kamo/LineairDB-storage-engine