PostgreSQL インサイド

# 外部テーブルに対する一括挿入の高速化

## PostgreSQL 14 でコミットされた機能の先行紹介

## 技術者 Blog

**綱川 貴之**

### はじめに

富士通のシニアディレクターである Amit Kapila が率いる"グローバル PostgreSQL 開発チーム"は、当社の PostgreSQL に対する取り組みの一環として、コミュニティーへ継続的な貢献を行っています。私はこのチームの一員として、世界中の才能ある情熱的なコミュニティーメンバーとともに、PostgreSQL の発展に積極的に取り組んでいます。
私たちのチームは、メンバーが取り組んだ機能やパッチに焦点を当て、ブログ記事を発信していきます。この記事では、PostgreSQL 14 にコミットされた「外部テーブルに対する一括挿入の高速化」について、それに取り組んだ理由と、エキサイティングな性能向上の成果をご紹介します。

### なぜこの開発に取り組んだか？

私たちのチームは、PostgreSQL のスケーラビリティーや性能の向上に注力しています。大きな目標の 1 つは、書き込みスケールアウトを実現することです。これにより、PostgreSQL を適用できるワークロードをさらに広げたいと考えています。
スケールアウトのための複数サーバー間のやりとりには、外部データラッパー（以降、FDW と略します）機構に基づく postgres_fdw を用いるアイデアが、コミュニティーでは有望であり現実的だと考えられています。なぜなら、postgres_fdw は長年にわたり、数多くの開発者による改善を積み重ねてきているからです。
そのようなときに私たちは、PostgreSQL コミッターである Tomas Vondra 氏が postgres_fdw による一括挿入を速くしようとしているのを見ました。彼が言うには、FDW を使ってシャード化したデータベースへの挿入が遅いという問題を、顧客からよく報告されるそうです。
あるユーザーも PostgreSQL のバグ報告用メーリングリストに、同様の問題を報告しました。その報告では、Google Cloud 上で postgres_fdw を使ってシャード化したデータベースに対し、INSERT SELECT 文で 2,000 万行を挿入するのに、1 時間 50 分もかかったといいます。シャード化しない場合は 8 分で完了したのに比べると、14 倍も遅いということです。
遅い原因は、ネットワークのレイテンシーと通信回数です。現在の FDW 機構では、FDW が 1 行ずつ外部テーブルに行を挿入します。そのたびに、遠隔のサーバーとの往復通信が生じるのです。
Tomas Vondra 氏たちは postgres_fdw のみを変更し、まとめて複数の行を外部サーバーに送る方法を試行していました。しかし、エグゼキュータを変更せずに済ませる方法を模索している間に、開発が 3 か月ほど止まっていたのです。

### どのような機能を開発したか？

そこで私は、次のような提案をしました。外部サーバーに複数行を一括して送るというアイデアは、Tomas Vondra 氏のものと同じです。

- FDW インターフェイスに、一括行挿入のための以下の関数を新たに追加する。
    - BeginForeignBatchInsert
    - ExecForeignBatchInsert
    - EndForeignBatchInsert
    - GetForeignBatchSize
- ExecForeignBatchInsert()は行の配列と行数を受け取り、それらの行を一括して外部サーバーに送る。
- executor は複数の行を蓄積し、まとめて FDW に渡す。蓄積する行数は、GetForeignBatchSize()で FDW から取得する。

 

- postgres_fdw は、渡された行配列を使って「INSERT ... VALUES (row 1), (row 2), ..., (row n)」文を組み立てて、外部サーバーでそれを実行する。
- postgres_fdw の外部サーバーと外部テーブルの設定に、batch_size オプションを新たに追加し、利用者が外部テーブルごとに一括して挿入する行数を指定できるようにする。

これは、実に素直でシンプルなつくりです。
これにより、INSERT SELECT 文だけでなく、複数行を指定した INSERT VALUES 文も速くなります。アプリケーションを変更する必要はありません。
上記のインターフェイスは、oracle_fdw や mysql_fdw といった、他のデータベース用の FDW でも素直に実装できるはずです。

## どれだけ速くなったか？

まず、単純なテーブルに対するデータ挿入を測定してみました。対象テーブルは 1 つの int 型のプライマリキー列と、1 つの text 型の列を持ちます。
別のテーブルからそのテーブルに、INSERT SELECT 文で 100 万件の行を挿入するのにかかった時間は次のとおりです。このとき、postgres_fdw の batch_size パラメーターの設定値は 100 です。つまり、一度に最大で 100 行を外部サーバーに送ります。

| **処理条件** | **処理時間** |
|---|---|
| FDW なしのローカルテーブル | 6.1 秒 |
| FDW を使ったリモートテーブル（PostgreSQL 13 の場合） | 125.3 秒 |
| FDW を使ったリモートテーブル（PostgreSQL 14 の場合） | **11.1 秒** |

すばらしい！11 倍も速くなりました。
次に、同じ列からなる対象テーブルを 8 つのハッシュパーティションに分割しました。同様に、INSERT SELECT 文で 100 万行を挿入するのにかかった時間は次のとおりです。

| **処理条件** | **処理時間** |
|---|---|
| FDW なしのローカルテーブル | 8.6 秒 |
| FDW を使ったリモートテーブル（PostgreSQL 13 の場合） | 113.7 秒 |
| FDW を使ったリモートテーブル（PostgreSQL 14 の場合） | **12.5 秒** |

これも良いですね！9 倍の速度向上です。
なお、この測定では、外部サーバーを同じホストに配置しました。「ping localhost」で測定されるネットワーク・レイテンシーは 34 マイクロ秒です。これほど小さなレイテンシーでも大きな性能向上が見られたということは、クラウドなどのより大きなネットワーク・レイテンシーを伴う環境では、この機能の効果はさらに大きいでしょう。

## さらなる高速化への期待

この機能により、1 つの外部テーブルへのデータ追加が劇的に速くなりました。ただ、これは単一の CPU を使った逐次処理の高速化です。
一方、私の同僚の Greg Nancarrow は、複数の CPU を使って INSERT SELECT 文を並列に処理できるようにする開発に取り組んでいま

 

す。また別の開発者たちは、CREATE TABLE AS SELECT（CTAS）文を並列化しようとしています。
これらを組み合わせることで、シャード化された OLTP データベースへのデータ移行や、データウェアハウスへの ETL 処理がさらに高速化する可能性に期待します。今後、また良いお知らせができることを楽しみにしています。

## 謝辞

この機能が比較的早くコミットされたことについて、Tomas Vondra 氏や Amit Langote 氏をはじめとする開発者の方々に深く感謝いたします。彼らの素早く熱心なフィードバックのおかげで、この機能を実現できたと考えています。

2021 年 3 月 26 日

 