PostgreSQL インサイド

LZ4 TOAST 圧縮

PostgreSQL 14 でコミットされた機能の紹介

技術者 Blog

唐 海英

南京富士通南大軟件技術有限公司

開発事業本部デジタルソリューション事業部

はじめに

新しい列圧縮オプションである LZ4 が PostgreSQL 14 に導入されました。これは、TOAST の既存の PGLZ よりも高速な圧縮を提供します。この記事では、新しい LZ4 圧縮を使用する場合の使用方法と性能について説明します。

背景

PostgreSQL では、データを保存するための基本単位は「ページ」であり、各ページのサイズはデフォルトで 8 キロバイトです。基本的に、1 行のデータについてページをまたいで保存することはできません。しかし、データ型によっては可変長のものがあり、1 ページのサイズを超える場合があります。この制限に対応するために、大きなフィールド値は圧縮されたり、複数の物理的な行に分割されたりします。この技術は TOAST（The Oversized-Attribute Storage Technique）(https://www.postgresql.org/docs/14/storage-toast.html) と呼ばれています。

デフォルトでは、テーブルに可変長の列があり、行データのサイズが TOAST_TUPLE_THRESHOLD（通常は 2 キロバイト）を超えた場合にのみ、TOAST の機構が利用されます。まず、データが圧縮され、それでもまだデータが大きすぎる場合は、行外に格納されます。列の格納戦略が EXTERNAL / PLAIN に指定されている場合のみ、圧縮は無効になることに注意してください。

PostgreSQL 13 までは、TOAST は PostgreSQL の組み込みアルゴリズムである PGLZ という 1 つの圧縮アルゴリズムしかサポートしていませんでした。周知のとおり、PGLZ よりも高速で、より高い圧縮率を持つ圧縮アルゴリズムは他にもたくさんありますが、それらは PostgreSQL 14 以前では利用できませんでした。PostgreSQL 14 では、高速な可逆圧縮アルゴリズムで知られる LZ4 圧縮 (http://lz4.github.io/lz4/) が追加されました。これにより、TOAST の圧縮・解凍の速度向上に役立つことが期待できます。

LZ4 の使い方

PostgreSQL 14 で新たに追加された LZ4 圧縮機能を使用するための手順を説明します。

まず、LZ4 関連のライブラリーを OS にインストールし、PostgreSQL のコンパイルやパッケージングの際に「--with-lz4」オプションを指定します。

次に、PostgreSQL インスタンスでは、GUC パラメーター「default_toast_compression」を設定することにより、TOAST の圧縮アルゴリズムを指定できます。設定ファイル postgresql.conf を編集して圧縮アルゴリズムに LZ4 を設定するか、「SET」コマンドを使用して現在のクライアント接続（セッション）の圧縮アルゴリズムを変更します。

また、テーブルの作成時に列の圧縮アルゴリズムを LZ4 に指定することもできます。同様に、ALTER TABLE コマンドを使用して、すでに作成されたテーブルの圧縮アルゴリズムを変更することもできますが、新たに変更された圧縮アルゴリズムは ALTER TABLE コマンドの実行後に挿入されたデータに対してのみ有効になります。

以下に、LZ4 を設定・使用する例を示します。

例 1）デフォルトの圧縮アルゴリズムとして「LZ4」を設定する方法

```
postgres=# SET default_toast_compression=lz4;
SET
```


例 2）CREATE TABLE 文で列ごとの圧縮アルゴリズムを指定する方法

```
postgres=# CREATE TABLE tbl (id int, col1 text COMPRESSION pglz, col2 text COMPRESSION lz4, col3 text);
CREATE TABLE
postgres=# ¥d+ tbl
                                        Table "public.tbl"
 Column |  Type   | Collation | Nullable | Default | Storage  | Compression | Stats target | Description
--------+---------+-----------+----------+---------+----------+-------------+--------------+-------------
 id     | integer |           |          |         | plain    |             |              |
 col1   | text    |           |          |         | extended | pglz        |              |
 col2   | text    |           |          |         | extended | lz4         |              |
 col3   | text    |           |          |         | extended |             |              |
Access method: heap
```

describe コマンドの'¥d+'を使って、カラムの圧縮アルゴリズムを表示できます。カラムが圧縮をサポートしていない場合や圧縮アルゴリズムが指定されていない場合は、空白が表示されます。
上記の例では、カラム「id」は圧縮をサポートしておらず、カラム「col1」は PGLZ、カラム「col2」は LZ4、カラム「col3」は圧縮アルゴリズムが指定されていないため、デフォルトの圧縮アルゴリズムが使用されます。

例 3）ALTER TABLE 文で列の圧縮アルゴリズムを変更する方法

```
postgres=# INSERT INTO tbl VALUES (1, repeat('abcdefg',1000), repeat('abcdefg',1000),repeat('abcdefg',1000));
INSERT 0 1
postgres=# ALTER TABLE tbl ALTER COLUMN col1 SET COMPRESSION lz4;
ALTER TABLE
postgres=# INSERT INTO tbl VALUES (2, repeat('abcdefg',1000), repeat('abcdefg',1000),repeat('abcdefg',1000));
INSERT 0 1
postgres=# SELECT id, pg_column_compression(id), pg_column_compression(col1),  pg_column_compression(col2),
pg_column_compression(col3) FROM tbl;
 id | pg_column_compression | pg_column_compression | pg_column_compression | pg_column_compression
----+-----------------------+-----------------------+-----------------------+-----------------------
  1 |                       | pglz                  | lz4                   | lz4
  2 |                       | lz4                   | lz4                   | lz4
(2 rows)
```

上の図では、1 行目の列「col1」は、圧縮方法を PGLZ から LZ4 に変更しても、依然として PGLZ で圧縮されていることがわかります。つまり、既存のデータの圧縮方法は変わらないということです。

注意事項

- CREATE TABLE .... AS ... や INSERT INTO ... SELECT ... を使って他のテーブルからデータを挿入する場合、挿入されるデータの圧縮方法は元のデータと同じです。
- LZ4 をサポートする一方で、pg_dump と pg_dumpall には「--no-toast-compression」というオプションが追加され、圧縮方式をダンプ情報に出力しません。

性能比較

ここでは、LZ4 と PGLZ を圧縮率と圧縮速度の両面で比較するテストを行いました。参考値として、非圧縮データ（データ型の格納戦略に「EXTERNAL」を指定した場合）のテスト結果も併記しました。非圧縮データの場合、圧縮・解凍にかかる時間はありませんが、その分、データの読み書きにかかる時間が長くなります。

準備

テストには以下のデータを使用し、表中の見出しとして以下のように記載しています。

- pg doc：PostgreSQL ドキュメント（1 行につき 1 つの HTML ファイル）
- Silesia Corpus（注 1）から提供されたデータ
    - html：HTML
    - English text：テキスト
    - src：ソースコード
    - exe：実行可能なバイナリー
    - picture：画像

テストサーバーのプロセッサースペックは、Intel® Xeon® Silver 4210 CPU @2.20GHz、10 コア / 20 スレッド / 2 ソケットです。
SQL の実行時間を計測するために「pgbench」を使用し、テーブルのサイズを確認するためにシステム関数「pg_table_size」を使用しています。また、各テストの前に「VACUMM FULL」コマンドを実行し、データストレージをクリアしています。

注 1　圧縮アルゴリズムをテストするためのさまざまな特性を持つファイルのセットです。

- Silesia Corpus（GitHub のページへ）
  https://github.com/MiloszKrajewski/SilesiaCorpus

圧縮率

PGLZ と LZ4 の圧縮率は、データ内の重複項目に関係しており、重複項目が多いほど圧縮率は高くなります。
しかし、圧縮率が低い場合、データサイズが閾値に達しても圧縮は行われません。なぜなら、この場合、圧縮しても効果的にディスク容量を節約できず、逆に解凍に余分な時間とリソースがかかってしまうからです。
PostgreSQL 14 の現在のソースコードによると、PGLZ では 25%以上の圧縮率が必要とされ、LZ4 では圧縮データが非圧縮データよりも大きくならないことが必要とされています。

図 1 では、LZ4 と PGLZ の圧縮アルゴリズムを使用したテーブルサイズを非圧縮のテーブルサイズと比較しました。ほとんどのケースで、PGLZ の圧縮率がわずかに優れていることがわかります。平均して、PGLZ の圧縮率は 2.23 で、LZ4 の圧縮率は 2.07 です。つまり、LZ4 と比較して、PGLZ を使用することで約 7%のディスク容量を節約できます。

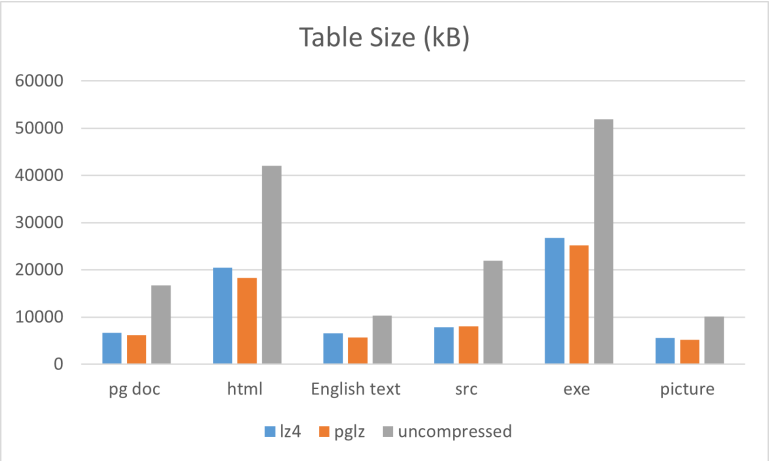

図 1：テーブルサイズの比較

圧縮 / 解凍速度

TOAST のデータは、挿入 / 検索される際に圧縮 / 解凍されます。そこで、いくつかの SQL 文を実行して、異なる圧縮アルゴリズムが圧縮 / 解凍速度に与える影響を見てみました。

INSERT 文

図 2 は、INSERT 文の実行にかかった時間を示しています。
これによると、LZ4 は圧縮されていないものよりもデータの挿入にわずかに時間がかかるのに対し、PGLZ を使うとデータの挿入にかかる時間が大幅に増加することがわかりました。平均すると、LZ4 の圧縮にかかる時間は、PGLZ の圧縮にかかる時間の約 20%に過ぎません。これは非常に大きな改善です。

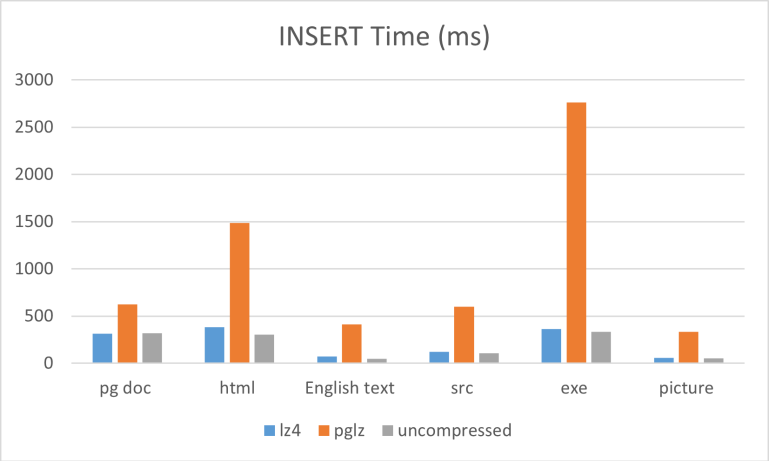

図 2：INSERT 文の性能比較

SELECT 文

図 3 は、SELECT 文の実行にかかった時間を示しています。
データを問い合わせている間、LZ4 は PGLZ と比較して約 20%の時間短縮を実現しており、非圧縮と比較しても明らかな増加は見られないことがわかります。解凍にかかる時間は非常に少なくなっていることが伺えます。

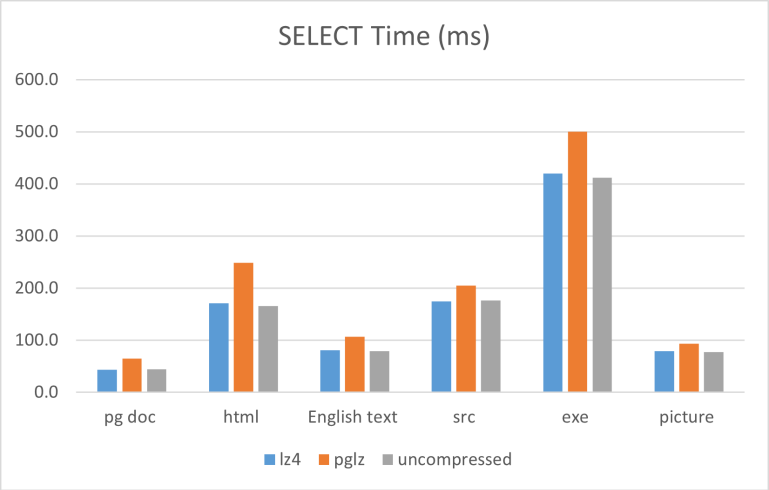

図 3：SELECT 文の性能比較

16 個のクライアントからの INSERT 文

もう 1 つの一般的なシナリオは、複数のクライアントからデータベースにアクセスすることです。
図 4 の結果からわかるように、16 個のクライアントがある場合に、1 つの大きなファイル（HTML、テキスト、ソースコード、実行可能なバイナリー、画像）を LZ4 で圧縮したとき、PGLZ に比べて 60%から 70%改善され、複数の小さなファイル（PostgreSQL 文書）を挿入したときにも、わずかながら改善が見られました。
また、非圧縮に比べて大幅に改善されていますが、これは圧縮することでディスクに書き込まれるデータ量が減るためだと推測します。

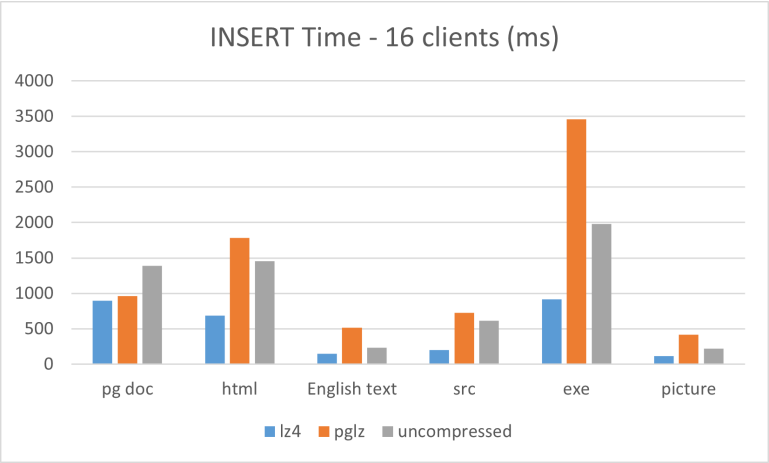

図 4：16 個のクライアントがある場合の INSERT 文の性能比較

16 個のクライアントからの SELECT 文

図 5 に示すように、複数のクライアントからの問い合わせシナリオでも、ほとんどの場合、LZ4 は PGLZ よりも優れた性能を発揮します。

図 5：16 個のクライアントがある場合の SELECT 文の性能比較

文字列関数

さらに、文字列関連の関数をいくつか呼び出して、テキスト処理の速さを比較しようと試みた結果を図 6 に示します。ここでも、どちらも LZ4 が PGLZ を上回っています。また、LZ4 圧縮されたデータを使った各関数にかかる時間は、非圧縮データとほとんど変わらないことから、LZ4 圧縮は文字列演算の速度にほとんど影響を与えないことがわかりました。使用した文字列関数はそれぞれ以下のとおりです。

- SQL1：テキストの長さを取得する length 関数を使用した SELECT 文
- SQL2：部分文字列を検索する substr 関数を使用した SELECT 文
- SQL3：テキストを大文字に更新する upper 関数を使用した UPDATE 文
- SQL4：いくつかのテキストを連結する textcat 関数を使用した UPDATE 文

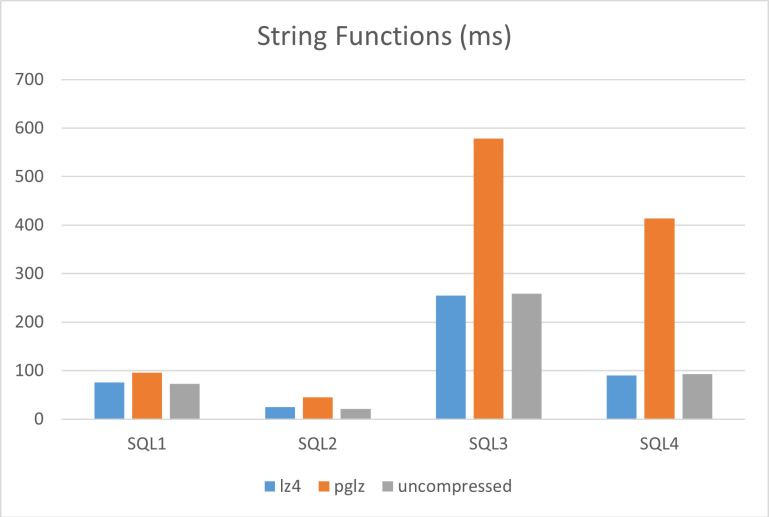

図 6：文字列関数の性能比較

テストの総括

PGLZ と比較して、LZ4 は TOAST データの圧縮および解凍においてより効率的です。クエリの実行速度は非圧縮データに近く、データの挿入速度は PGLZ に比べて約 80%向上し、優れた性能を発揮しています。もちろん、その代償として圧縮率が犠牲になることもありますが、実行速度を向上させたいのであれば、PGLZ ではなく LZ4 を強くお勧めします。
テーブル内のデータが圧縮に適しているかどうかは、事前に検討する必要があります。圧縮率が良くない場合、とりあえずデータの圧縮を試して、諦めることも必要です。これは余計にメモリー資源を浪費することになり、データの挿入速度にも大きく影響するからです。

今後に向けて

TOAST の LZ4 圧縮サポートにより、圧縮と解凍の性能が大幅に向上しました。
LZ4 に加えて、Zstandard のような優れた圧縮アルゴリズムもあります。Zstandard をサポートすることで、ユーザーは PGLZ と比較してさらに優れた圧縮率を得ることができます。また、LZ4 HC では、平均圧縮速度は LZ4 の解凍速度の 98.5%ですが、圧縮率は大幅に向上しています。今後の PostgreSQL では、圧縮アルゴリズムの選択肢が増え、ユーザーがニーズに応じて自由に選択できるようになることを期待しています。
TOAST の他にも、いくつかのシナリオで圧縮を使用する必要があります。私の知る限りでは、WAL（Write Ahead Logging：トランザクション更新ログ）の LZ4 圧縮は現在の開発版（https://commitfest.postgresql.org/33/3015/）ですでにサポートされており、非常に期待されています。

2021 年 11 月 12 日

