PostgreSQL 14 最新情報解説

2021-08-18

SRA OSS, Inc. 日本支社
高塚 遥

講演の概要:

- 2021年秋リリース予定

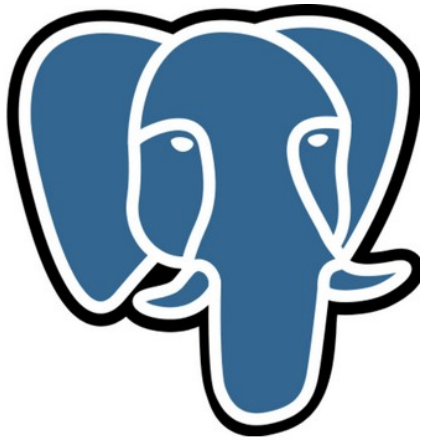
  PostgreSQL 14

  の新機能や性能向上を解説します。
- 非互換の点についても説明します。

講演者:

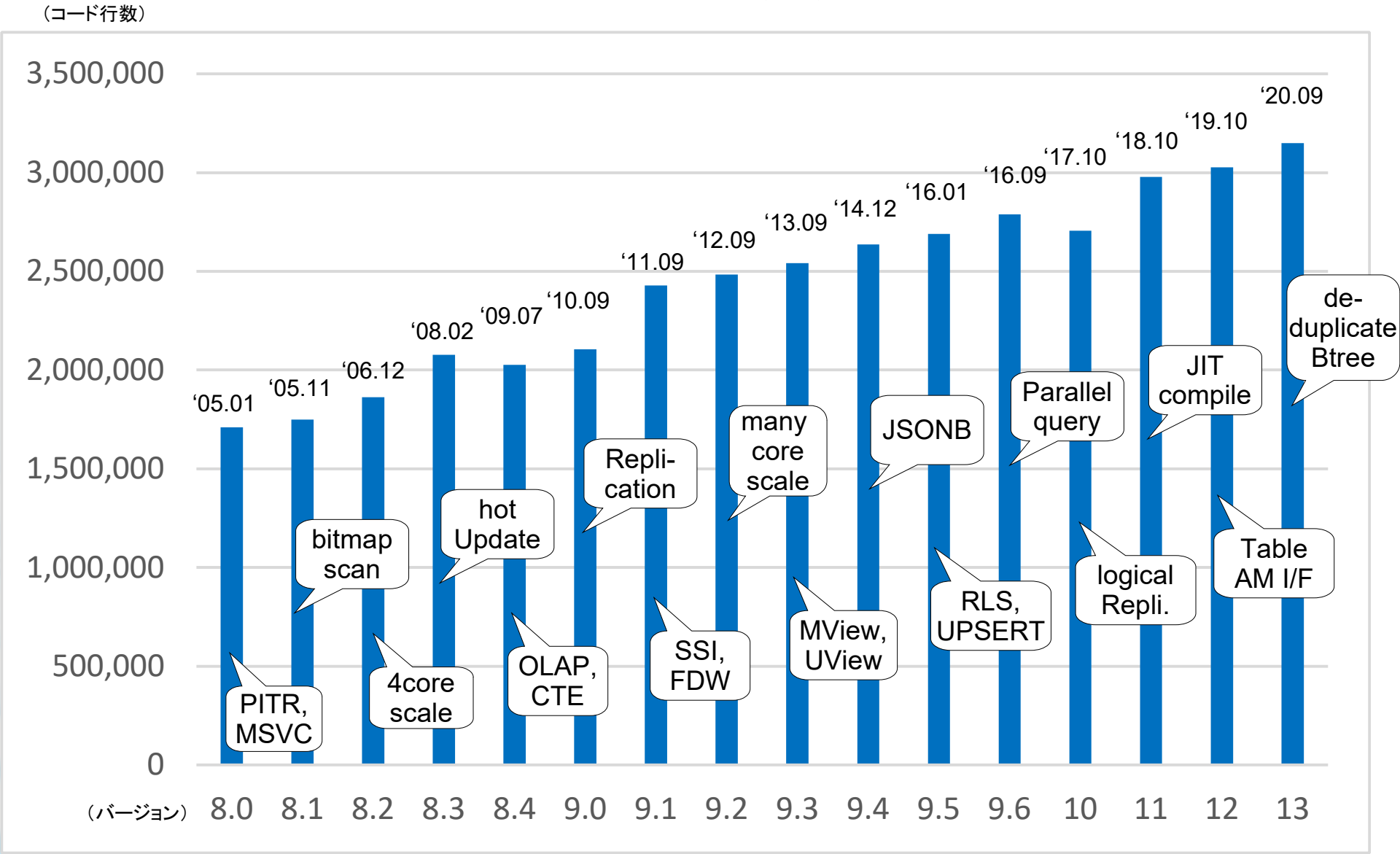
- 高塚 遥
- 普段は ヘルプデスク、コンサルティングなど、PostgreSQLの支援業務を各種手掛ける
- (特非)日本PostgreSQLユーザ会 理事

PostgreSQL について



- 多機能リレーショナルデータベース(RDB)
  - 豊富な商用RDB 置き換え実績
- 長い歴史を持つ
  - 1980年代のPOSTGRES、1996年 PostgreSQL 6.0
- BSDタイプの PostgreSQLライセンス
- 1社主導でないオープンソースソフトウェア
  - 主要開発者・開発管理者が各PostgreSQLビジネス企業に所属

これまでのリリース



PostgreSQL 14 の概要

性能向上

- Btree Index 肥大化防止
- 式の拡張統計情報
- LZ4によるデータ圧縮
- postgres_fdw改善
- ロジカルレプリケーション改善
- パラレルクエリ改善

SQL新機能

- マルチ範囲型
- SEARCH/CYCLE句
- 添え字構文

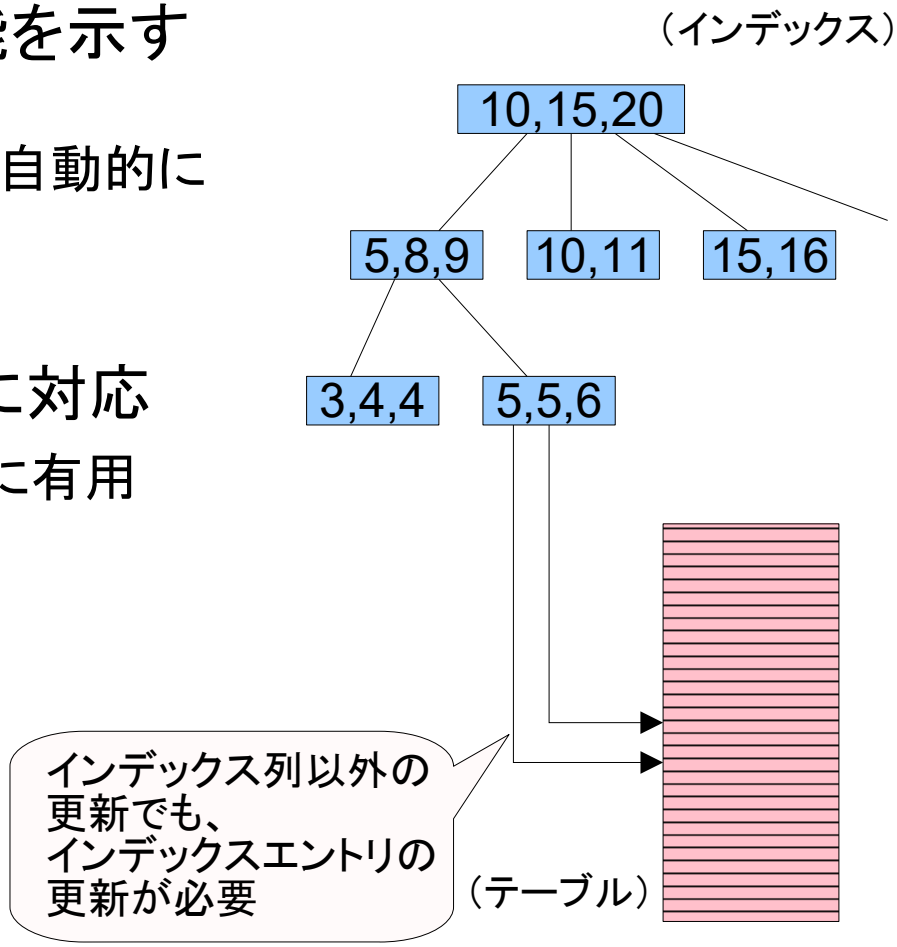
運用管理

- セッション制御機能
- pg_read_all / pg_write_all ロール
- 実行時統計ビュー拡張
- pg_amcheck
- pg_rewind拡張

性能向上

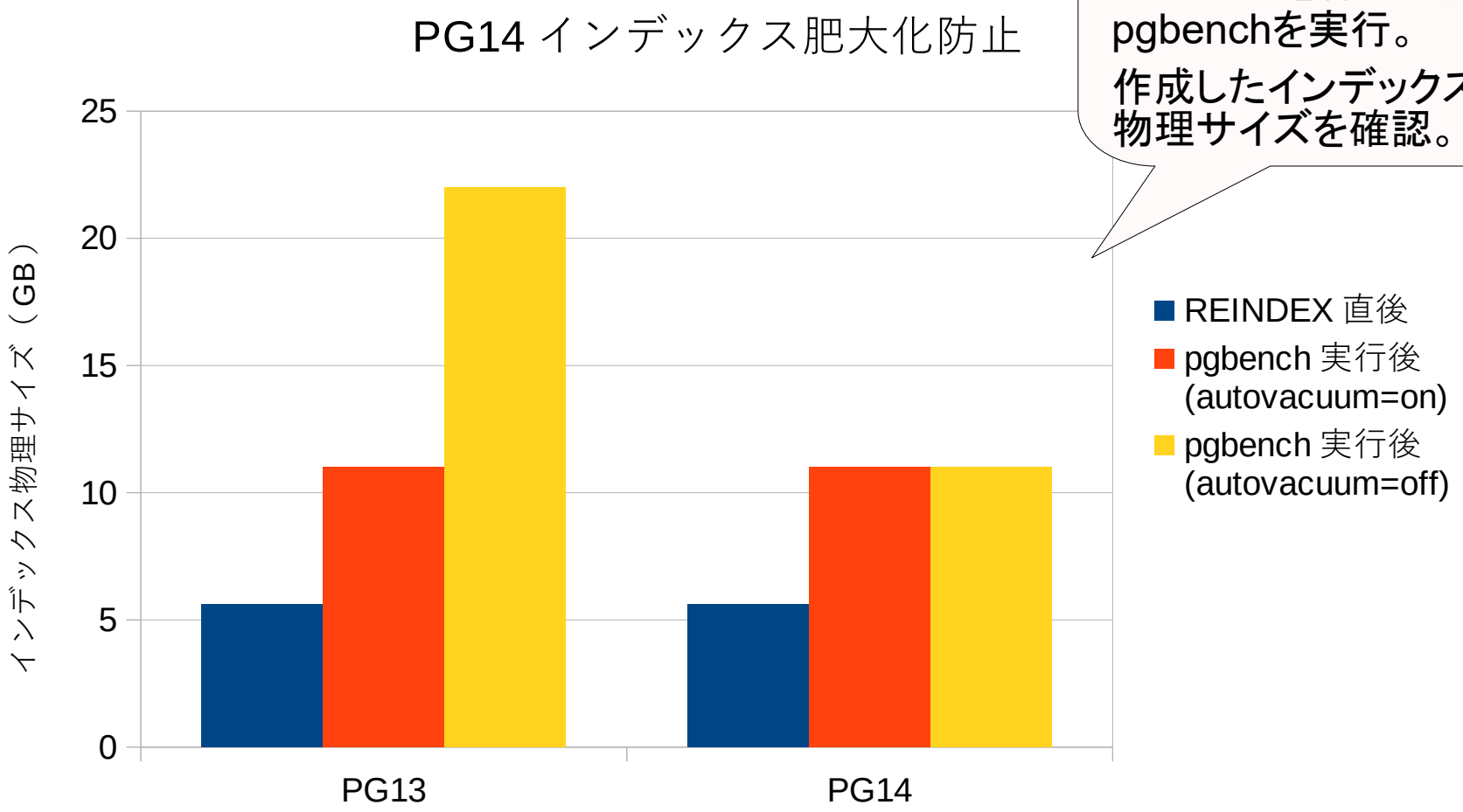
Btree Index肥大化防止

- 更新された項目に削除可能を示すヒント情報を付加
  - 同値のインデックスエントリを自動的に整理できるように
- UPDATE で生じる肥大化に対応
  - HOT機能が効かないケースに有用
  - ページ分割を試みるときに、削除可能項目を整理する
  - VACUUM処理を軽減



性能向上

更新しない列に Btree インデックスを作成。
UPDATEを繰り返す pgbenchを実行。
作成したインデックスの物理サイズを確認。



性能向上

式の拡張統計

- 拡張統計情報とは？
  - CREATE STATISTICS で作る追加のプランナ統計情報
- 式の統計情報の動作

連番データ生成

```
db1=# CREATE TABLE t_exstat (a int);
db1=# INSERT INTO t_exstat SELECT generate_series(1, 1000);

db1=# CREATE STATISTICS s ON mod(a,10), mod(a,20) FROM t_exstat;
db1=# ANALYZE t_exstat;

db1=# EXPLAIN (ANALYZE, TIMING OFF)
       SELECT * FROM t_exstat WHERE mod(a,10) = 0 AND mod(a,20) = 0;
                          QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on t_exstat
    (cost=0.00..25.00 rows=50 width=4) (actual rows=50 loops=1)
   Filter: ((mod(a, 10) = 0) AND (mod(a, 20) = 0))
   Rows Removed by Filter: 950
```

性能向上

LZ4によるデータ圧縮(1)

- 従来から可変長データは暗黙にデータ圧縮される
  - 1レコードで 2KB超の場合にTOASTテーブルに圧縮格納
- 圧縮アルゴリズムが選択可能
  - pglz(従来方式)、lz4

```
default_toast_compression = lz4
```

  - テーブルの列単位で指定

```
ALTER TABLE 《テーブル》ALTER 《列》
  SET COMPRESSION lz4;
```

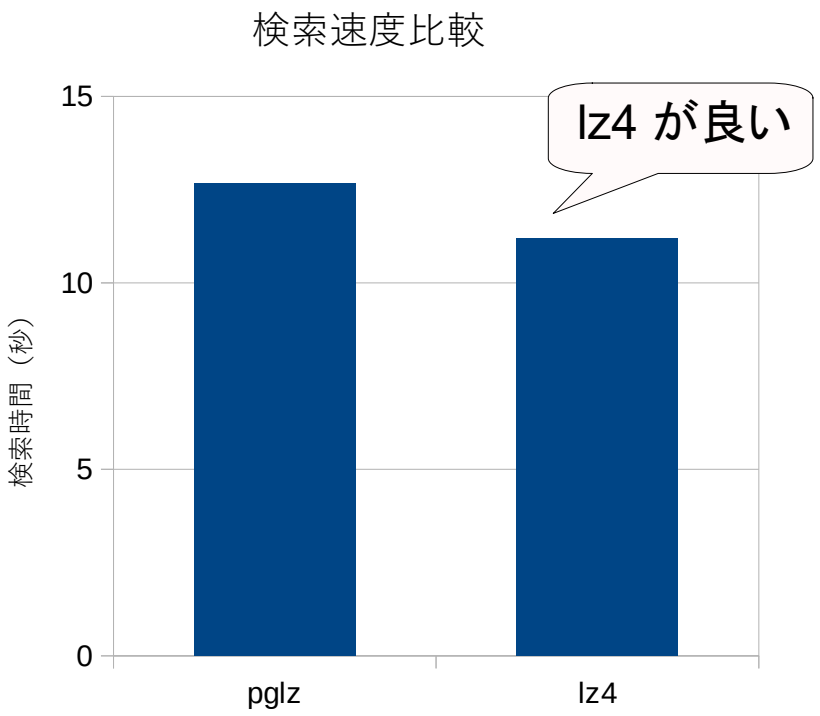
  - 適用は個別のフィールド(行の列)単位

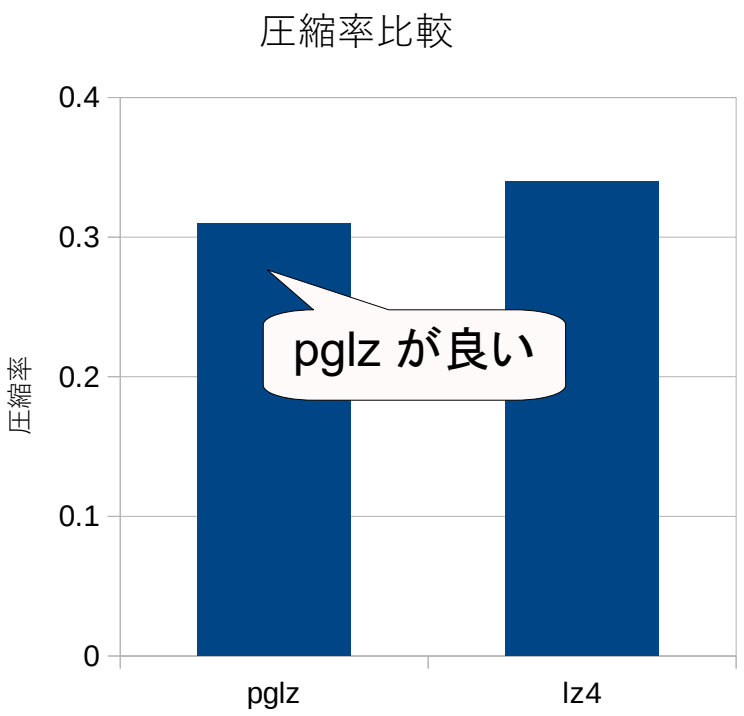
```
db1=# SELECT id, pg_column_compression(col1) FROM tbl1;
 id | pg_column_compression
----+-----------------------
  1 | lz4
  2 | pglz
```

性能向上

LZ4によるデータ圧縮(2)

- 性能特性の違いは？
  - PostgreSQL文書（SGMLファイル）で検証





性能向上

postgres_fdw改善

他に、接続管理のオプションや関数の追加も

- 外部テーブルのTRUNCATE
  - 従来は不可 → 参照先でTRUNCATEが実行される
- 外部テーブルの非同期スキャン
  - パーティションテーブルで並列検索

```
Append
  ->  Async Foreign Scan on ft_tbl1
  ->  Async Foreign Scan on ft_tbl2
```

- 外部テーブルへの一括挿入
  - 従来は 1行INSERTの繰り返し → 複数行挿入のINSERTを使用

```
INSERT INTO ft_test1
      SELECT g, 'XXXXXXXXXX' FROM generate_series(1, 1000) g;
```

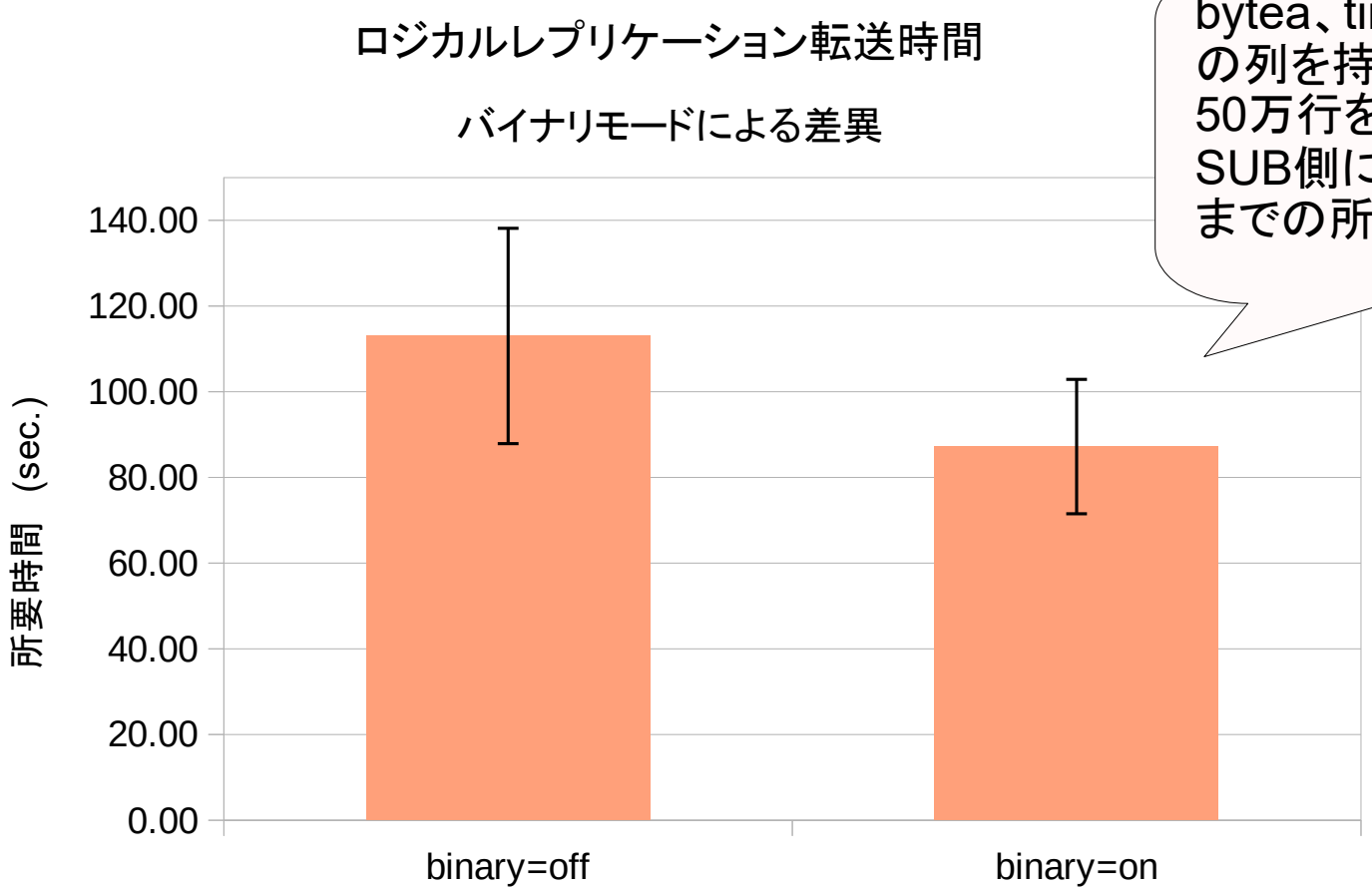
性能向上

ロジカルレプリケーション改善(1)

- ストリーム送信
  - 従来: 変更送信はコミット契機、それまでPUB側に蓄積
    → streaming=on で コミット前送出、SUB側蓄積
- 初期同期中更新を複数トランザクションに
  - 初期同期トランザクションの巨大化を防ぐ
- バイナリ転送モード
  - 従来: 転送はテキスト形式
    → binary=on でバイナリ転送可能

性能向上

ロジカルレプリケーション改善(2)



bytea、timestamp 型の列を持つテーブルに50万行を挿入、SUB側に反映されるまでの所要時間を比較。

性能向上

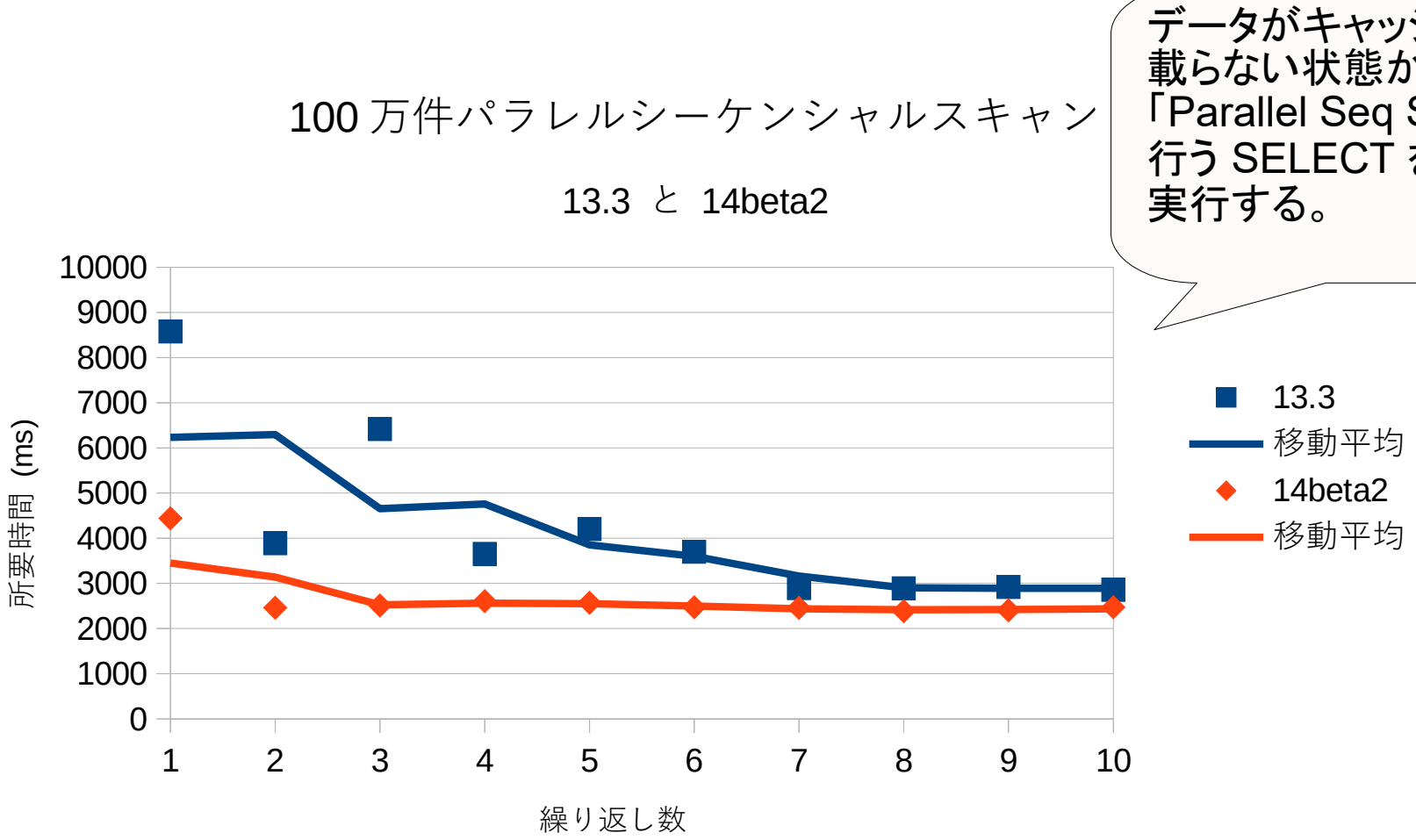
パラレルクエリ改善（1）

- パラレルシーケンシャルスキャンの改善
  - 各パラレルワーカが1ブロックずつ処理していた
    → 複数ブロックを処理するように変更、
    ストレージ I/O アクセス効率の向上をねらう
- PL/pgSQL の RETURN QUERY パラレル対応
- REFRESH MATERIALIZED VIEW パラレル対応

性能向上

パラレルクエリ改善（2）

データがキャッシュに載らない状態から始めて、「Parallel Seq Scan」を行う SELECT を繰り返し実行する。



SQL機能

マルチ範囲型

- 複数の範囲値からなるデータ
- 範囲型同様に「含まれるか？」「重なるか？」の演算子を持つ
- 範囲型同様にGiSTインデックス利用可

データ型	説明
int4multirange	integer 型のマルチ範囲型
int8multirange	bigint 型のマルチ範囲型
nummultirange	numeric 型のマルチ範囲型
tsmultirange	timestamp 型のマルチ範囲型
tstzmultirange	timestamp with time zone 型のマルチ範囲型
datemultirange	date 型のマルチ範囲型

datemultirangeの値の例

{[2021-07-01,2021-07-03),[2021-07-05,2021-07-10),
[2021-07-12,2021-07-17),[2021-07-19,2021-07-22)}

SQL機能

SEARCH / CYCLE句（1）

- CTE再帰問い合わせ（WITH RECURSIVE）の新構文：
- 幅優先探索、深さ優先探索を指定できる
- 循環参照の検出、循環参照での検索停止ができる
- 再帰検索の経路データを自動で作ってくれる

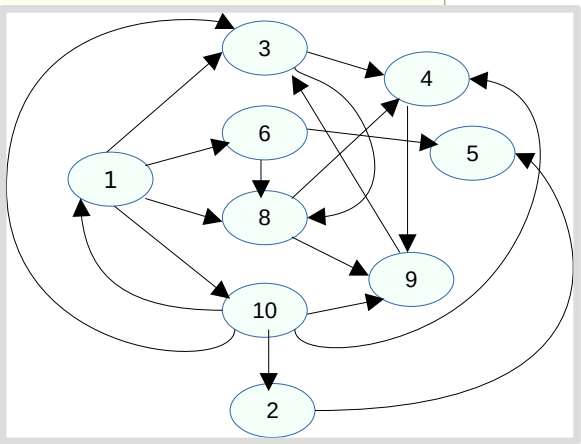
SQL機能

SEARCH / CYCLE句(2)

```
CREATE TABLE
 t_sns_follow
 (id1 int, id2 int);
```

```
=# SELECT chain AS follow_chain, cyclemark, cyclepath
    FROM (
     WITH RECURSIVE search_tree (id2, chain) AS (
      SELECT id2, ARRAY[id1,id2] FROM t_sns_follow f WHERE f.id1 = 1
      UNION ALL
      SELECT f.id2, st.chain || ARRAY[f.id2] FROM t_sns_follow f
        JOIN search_tree st ON (f.id1 = st.id2)
     ) SEARCH DEPTH FIRST BY id2 SET search_seq
       CYCLE id2 SET cyclemark USING cyclepath
     SELECT chain, search_seq, cyclemark, cyclepath FROM search_tree
    ) v
    ORDER BY search_seq;

   follow_chain    | cyclemark |       cyclepath
-------------------+-----------+-----------------------
 {1,3}             | f         | {(3)}
 {1,3,4}           | f         | {(3),(4)}
 {1,3,4,9}         | f         | {(3),(4),(9)}
 {1,3,4,9,3}       | t         | {(3),(4),(9),(3)}
 {1,3,8}           | f         | {(3),(8)}
 {1,3,8,4}         | f         | {(3),(8),(4)}
```



SQL機能

添え字構文

- hstore型、jsonb型で配列型と同様の添え字をサポート

```
=# SELECT j['1']['A'] FROM
    (VALUES ('{"1":{"A":100, "B":200}, "2":{"C":300}}'::jsonb)) AS v(j);
  j
-----
 100
(1 row)


=# SELECT h['B'] FROM (VALUES ('A=>100, B=>200'::hstore)) AS v(h);
  h
-----
 200
(1 row)
```

json型は非対応

運用管理

セッション制御機能

- サーバ側機能として長期アイドルセッションを切断
  - idle_session_timeout:
    アイドルセッションを強制切断するタイムアウト

    ← 従来は運用管理者側が cronジョブなどで対応

- クライアント側切断を早期検知
  - client_connection_check_interval
    クライアントの接続状態を検査する周期

    ← 従来はサーバ側でSQL実行中には検出されなかった

運用管理

新システムロール

- pg_read_all_data
  - 全てのデータを参照だけできる
    （SELECT権限、USAGE権限）

シンプルな方法で、
参照専用ユーザを
作ることができる

- pg_write_all_data
  - 全てのデータを更新だけできる
    （USAGE権限、INSERT権限、DELETE権限、UPDATE権限）

- pg_database_owner
  - 暗黙に現在のデータベース所有者をメンバとしているロール。
    - pg_database_owner に権限を付与すると、
      権限判断時点のデータベース所有者に権限を付与したことになる

運用管理

実行時統計情報の追加(1)

- COPY進捗情報
  - pg_stat_progress_copyビュー

```
=# SELECT * FROM pg_stat_progress_copy;
-[ RECORD 1 ]----+---------  ← 実行中のコピーがある場合のみ行が表示される
pid              | 3987
datid            | 16406
datname          | db1
relid            | 16407
command          | COPY TO      ← COPY TO か COPY FROM か
type             | FILE         ← 他に PROGRAM、PIPE、CALLBACK
bytes_processed  | 37164345     ← ここまでで処理したバイト数
bytes_total      | 0            ← COPY TO では 全体サイズが不明なため 0
tuples_processed | 556350       ← ここまでで処理した行数
tuples_excluded  | 0            ← ここまでで処理した行数の除外分
```

運用管理

実行時統計情報の追加(2)

解釈は実装を把握していないと難しい

- WAL書き出し情報
  - pg_stat_walビュー:
    - WALレコード単位の処理量、ディスク書き出し・フラッシュの統計
- ロジカルレプリケーションの統計情報
  - pg_stat_replication_slotsビュー

```
=# SELECT * FROM pg_stat_replication_slots;
-[ RECORD 1 ]+---------
slot_name    | sub1
spill_txns   | 0          ← spill書き出ししたトランザクション数
spill_count  | 0          ← spill書き出しした数
spill_bytes  | 0          ← spill書き出ししたバイト数
stream_txns  | 1          ← ストリーム送出したトランザクション数
stream_count | 1          ← ストリーム送出した回数
stream_bytes | 67109019   ← ストリーム送出したバイト数
total_txns   | 1          ← 処理トランザクション数
total_bytes  | 67109019   ← 処理バイト数
```

運用管理

pg_amcheck

- テーブル、インデックスの内部データ構造を検査
- amcheck拡張を簡単に使うためのクライアントコマンド

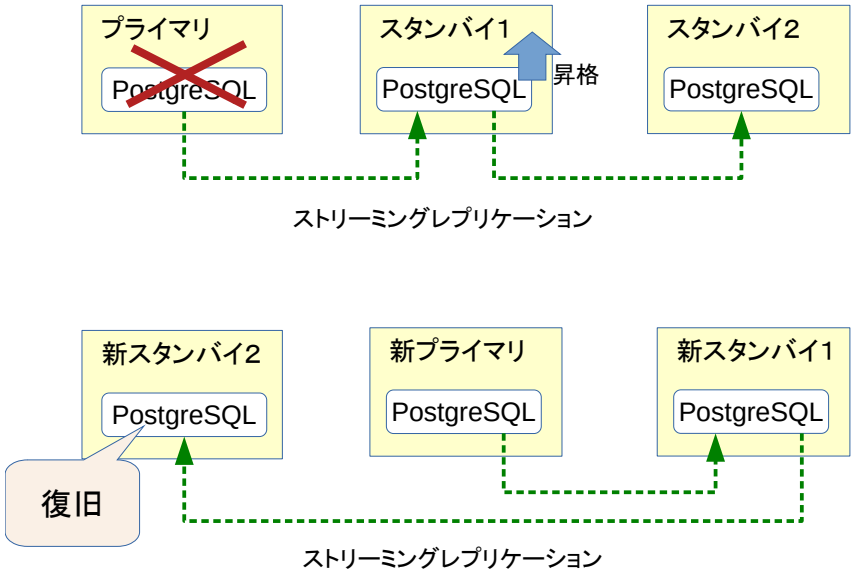
```
$ pg_amcheck db1

$ pg_amcheck -d db1 -s public -t pgbench_accounts -v
pg_amcheck: including database "db1"
pg_amcheck: in database "db1": using amcheck version "1.3" in
schema "public"
pg_amcheck: checking heap table
"amcheckdb"."public"."pgbench_accounts"
pg_amcheck: checking btree index
"amcheckdb"."public"."pgbench_accounts_pkey"
```

運用管理

pg_rewind 拡張

- WALを巻き戻すツール
  - あるスタンバイを別のプライマリに追随させるために使用
    （新プライマリの昇格点よりも先行しているスタンバイを巻き戻し）
- スタンバイをターゲットにした巻き戻しに対応
  - ユースケース：
    カスケードの場合



非互換

非互換の変更点

- pg_hba.conf で clientcert=1という記法が廃止
  - verify-ca（=従来の 1 と同じ意味）または verify-full を指定
- V2プロトコル廃止
  - V3プロトコル非対応クライアントは、ほぼ無い
  - サーバ側プリペアドステートメントを使用しないようにするため、あえて protocolVersion=2 を指定している場合に注意
- 後置単項演算子の廃止
  - 該当する組込演算子は 階乗の！のみ（代替は factorial関数）
- EXTRACT 戻り値が float8型 から numeric型
- pg_standby廃止

まとめ

- バージョン14の拡張は：
- 実用的な機能追加が中心
- 例年通りのスケジュールで進行中
  - 2021年秋リリース予定
  - 仕様小変更は未だありそう

- 検証レポート を公開

『SRAOSS PostgreSQL14 検証』
で検索

https://www.sraoss.co.jp/tech-blog/pgsql/pg14report/