PostgreSQL 9.4
評価検証報告

SRA OSS, Inc. 日本支社　高塚 遙
2014-09-11 15:55 ～ 16:30
PostgreSQL 9.4 最新情報セミナー

はじめに

本講演の構成

Part 1　性能アップって、どのくらいですか

Part 2　この新機能は何ですか

講演者について

高塚 遙
SRA OSS, Inc. 日本支社にて
PostgreSQLサポート、各種クラスタ構築技術支援、
マイグレーション技術支援、などを担当

性能アップって、
どのくらいですか？

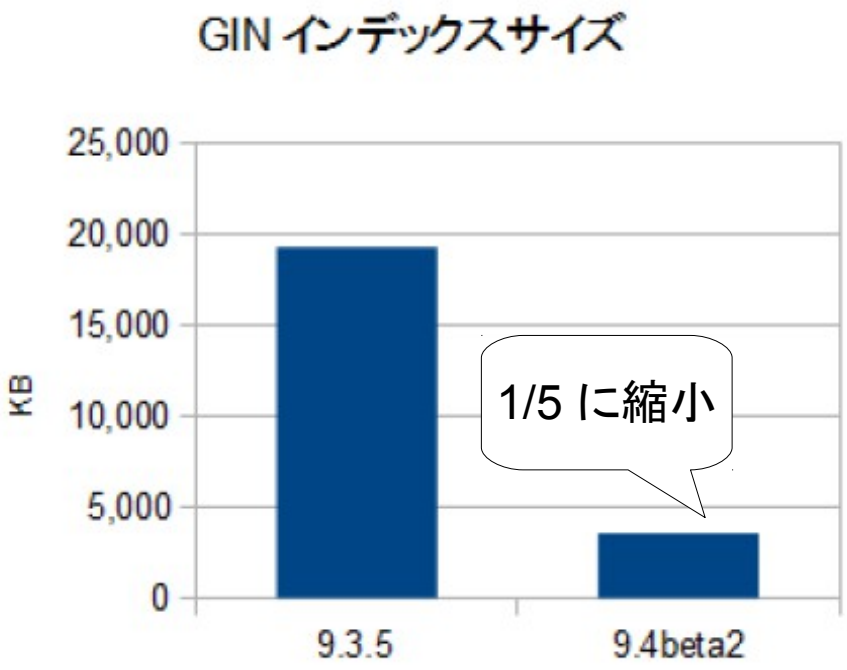
PG 9.4 検証報告 Part1 : GIN 性能改善 (1)

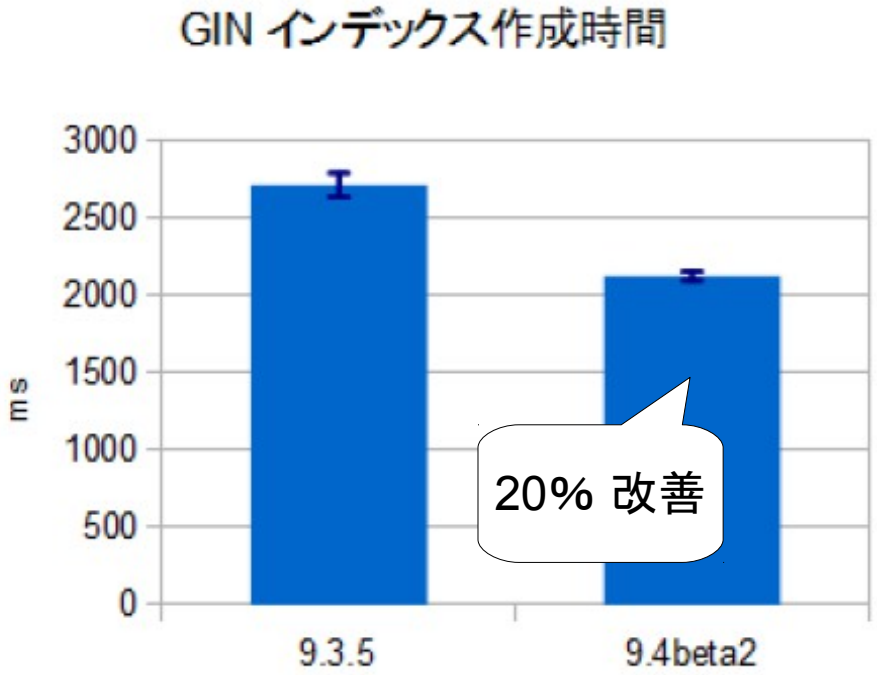
- 基礎的な性能改善度合いを確認
  - 50万件データを使って同マシンで PostgreSQL 9.3 と 9.4 を比較
- 4つの局面で優劣を比較
  - インデックスの物理サイズ
  - 既にデータがある状態からのインデックス作成性能
  - インデックスが作成されている状態からのデータ挿入性能
  - インデックスを使った検索性能

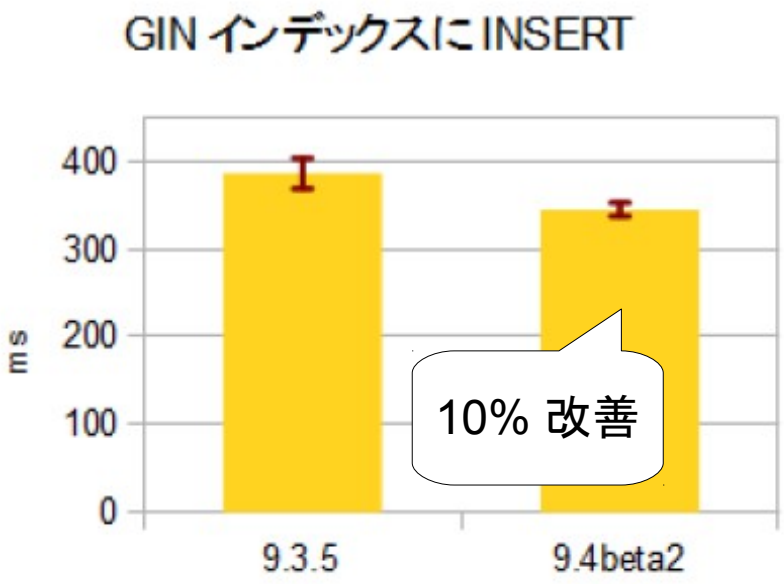
```
db1=# SELECT count(*) FROM ttag
        WHERE tags @> ARRAY['1', '2', '3', '4', '5'];
```

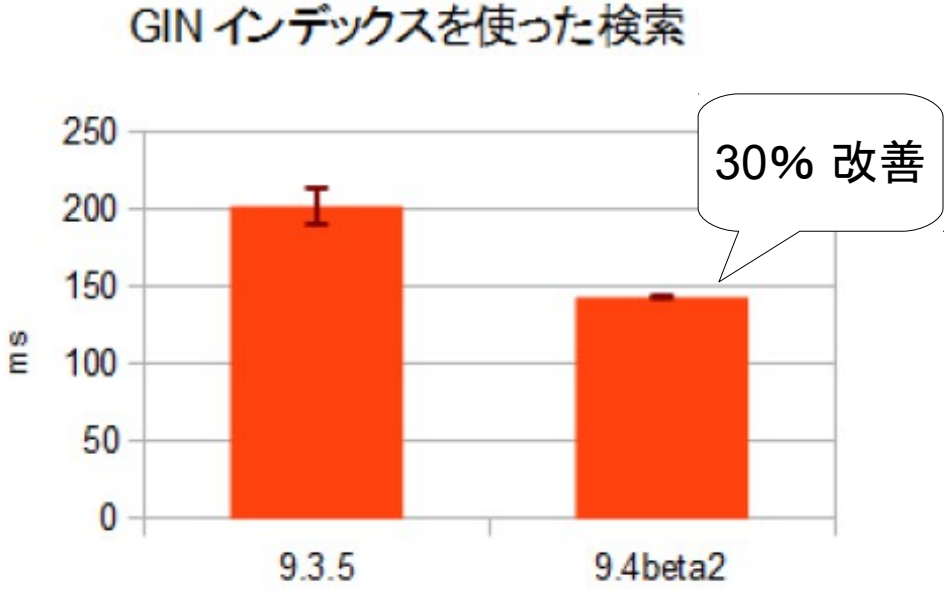
配列カラムに対して、指定した配列を「含む」行の件数を調べる

PG 9.4 検証報告 Part1 : GIN 性能改善 (2)







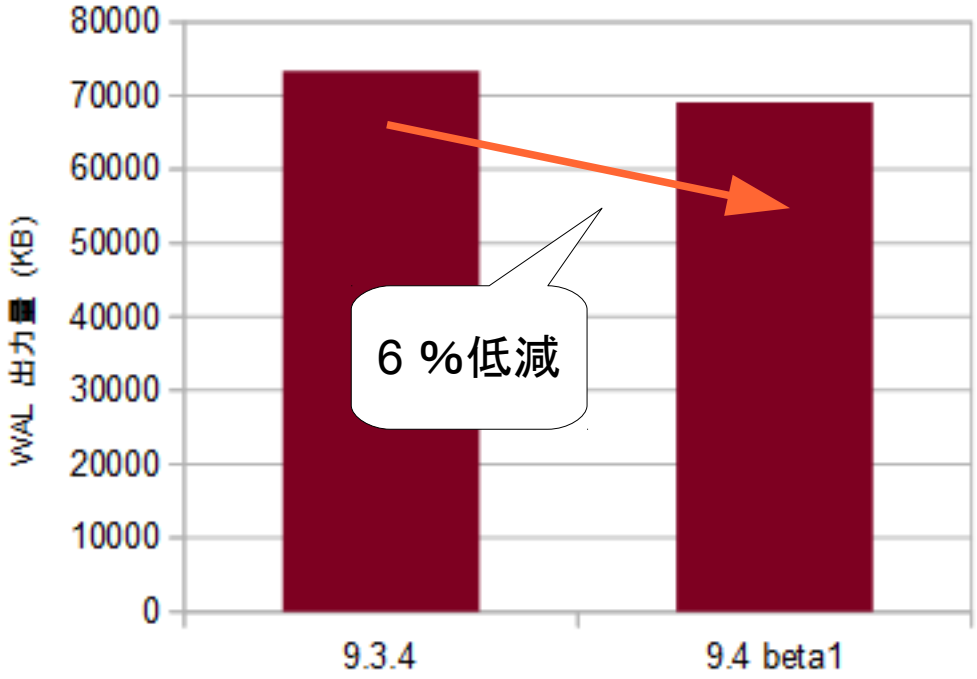


PG 9.4 検証報告 Part1：WALの性能改善（1）

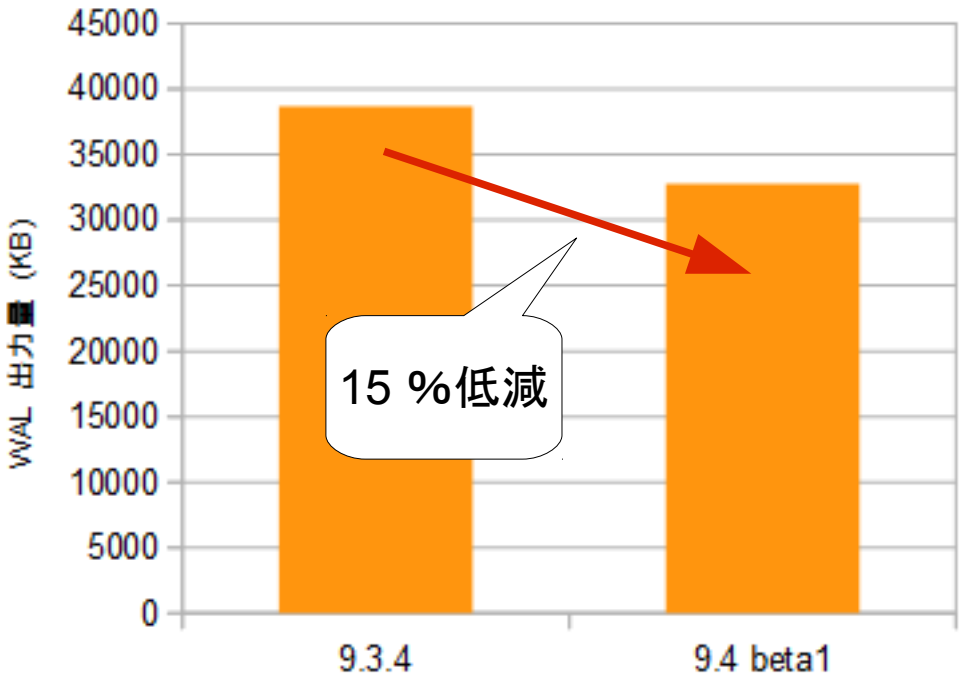
- データ圧縮により WAL出力量が低減
  - 更新トランザクションを 5万回実行
  - カスタムシナリオも実施 — UPDATEだけの単純な更新内容に変更

```
$ pgbench -c 5 -t 10000 -n
```

WAL 出力量（pgbench 標準シナリオ）

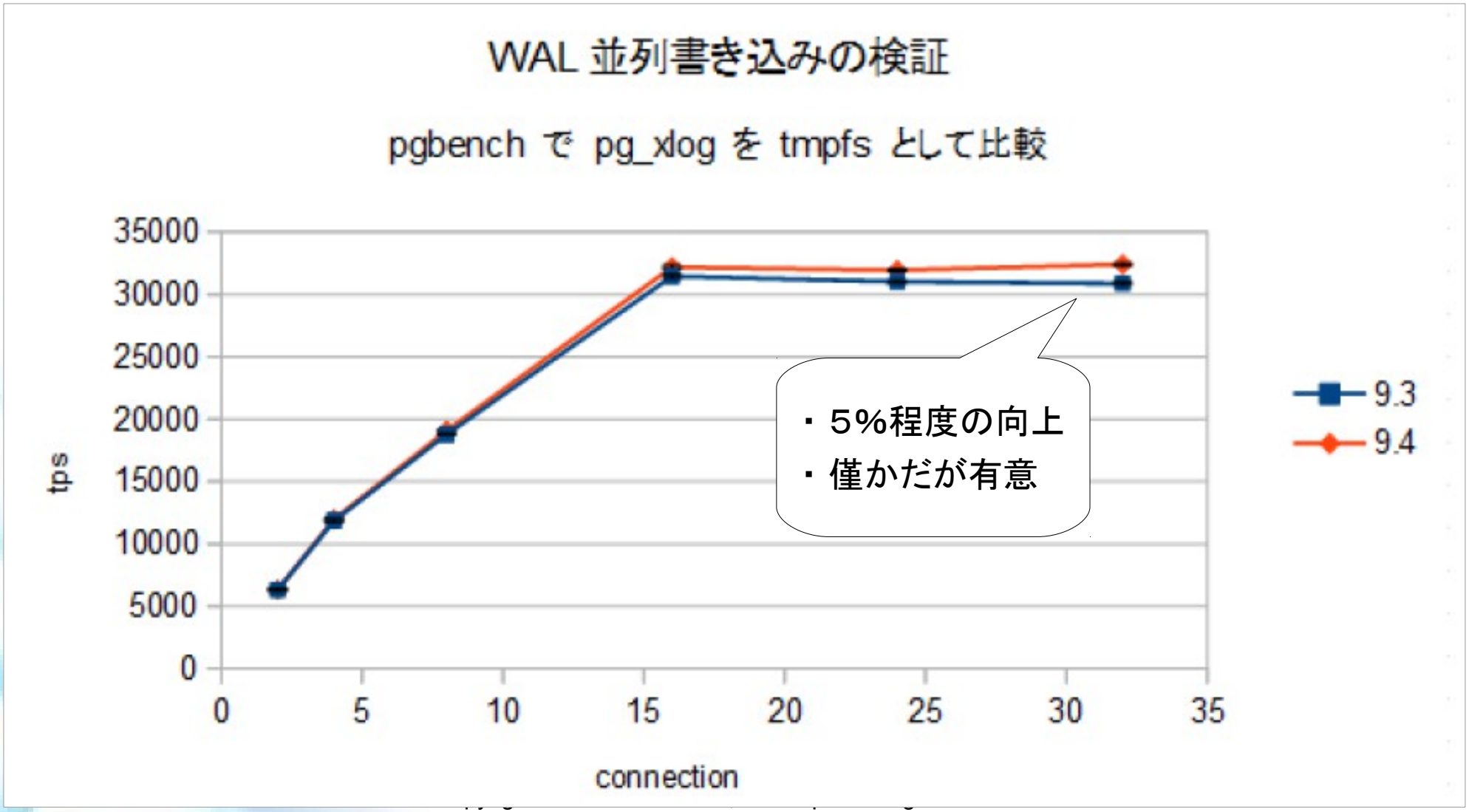


WAL 出力量（カスタムシナリオ）



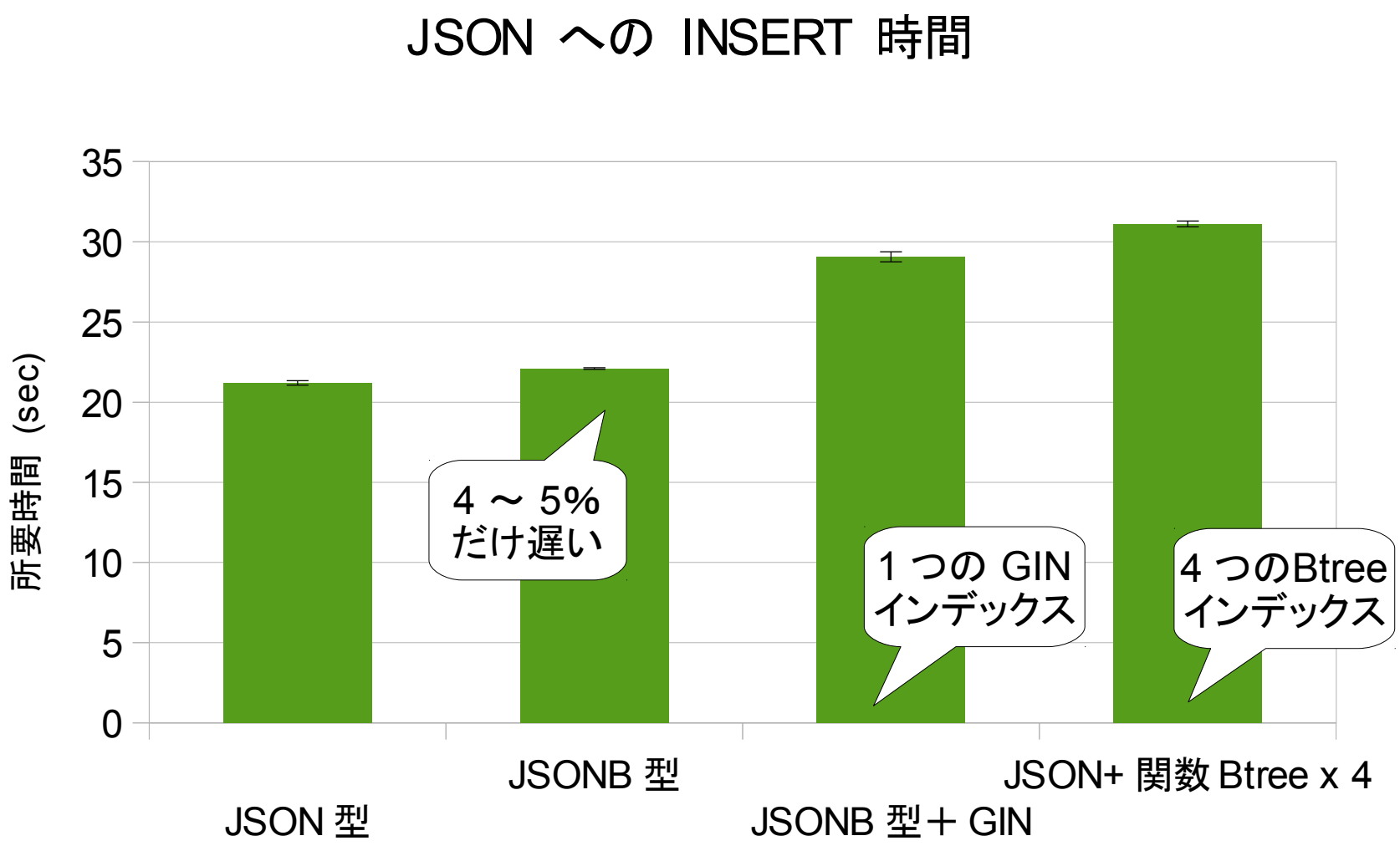
PG 9.4 検証報告 Part1：WALの性能改善（2）

- WAL 書き込みの同時実行性が向上
  - 4core × 2cpu マシンにて RAMディスク上でベンチマーク



PG 9.4 検証報告 Part1 : JSONB の性能 (1)

- 旧来の JSON 型 と JSONB型の比較 — データ投入
  - 10万件のログ情報を投入



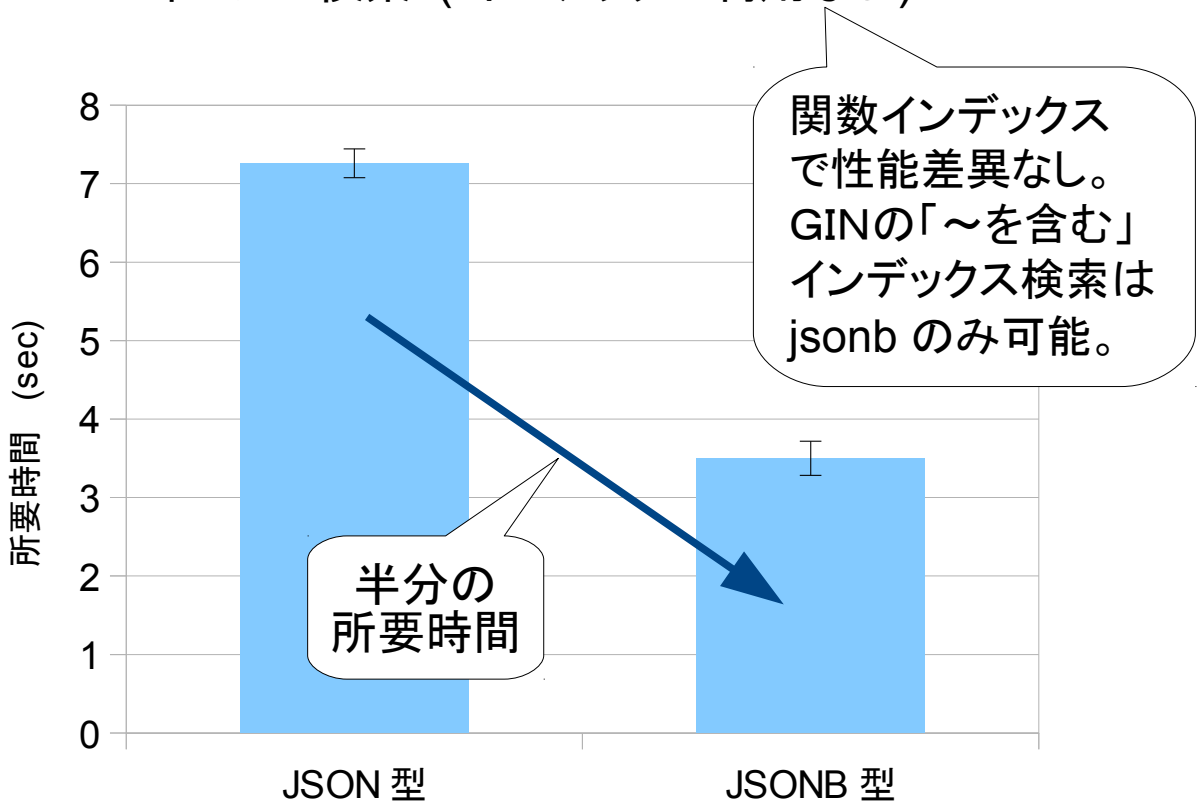
PG 9.4 検証報告 Part1 : JSONB の性能 (2)

- 旧来の JSON 型 と JSONB型の比較 — 検索

```
SELECT id, j-> 'ts', j -> 'mes'
 FROM t_logb
 WHERE (j -> 'lev')::text = '"fatal"'
 ORDER BY id DESC LIMIT 100;
```

テーブル定義	
id	serial
j	json or jsonb

フィールド検索（インデックス利用なし）



データ例

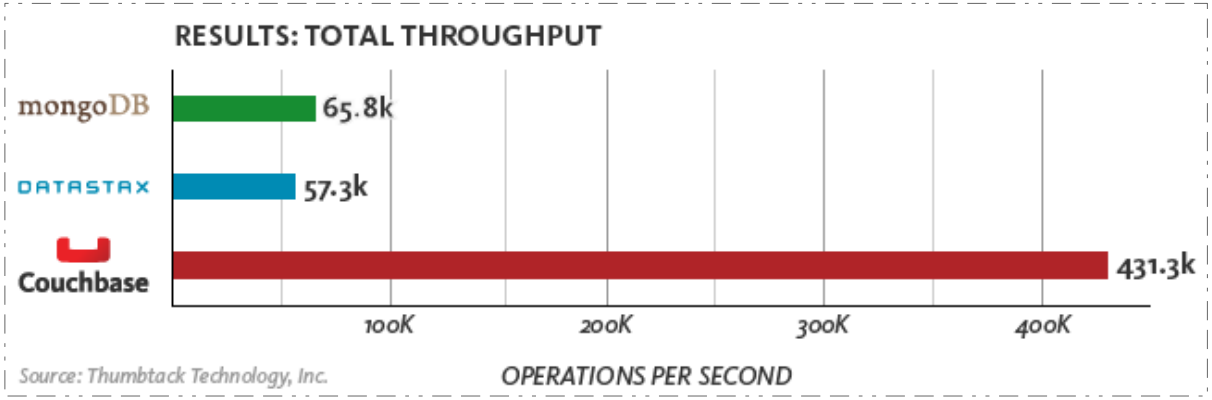
```
1400003    |
{ "ts":"2014-08-23 21:26:21",
  "pid":14212,
  "lev":"error", "tag":"ddd",
  "mes":"ae01d175 ....."
}

1400004    |
{ "ts":"2014-08-23 21:27:41",
  "pid":14858,
  "lev":"warning", "tag":"aaa"
  "mes":"9ff0d058fd ....."
}
```

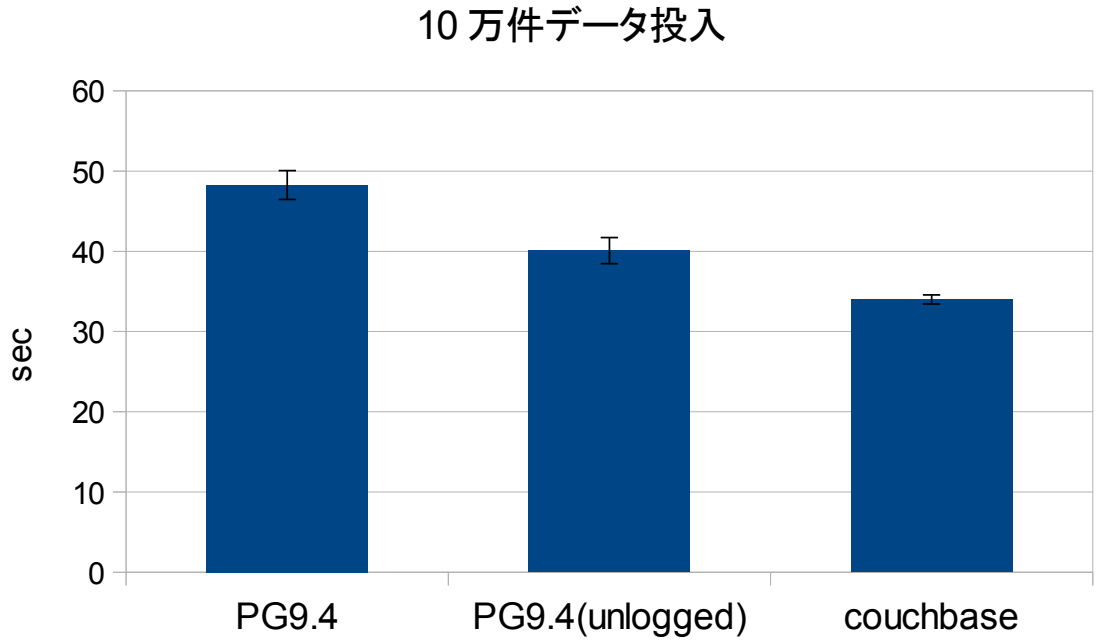
PG 9.4 検証報告 Part1 : JSONB の性能 (3)

- 他のドキュメント型データベースとの比較

- Couchbase Server
  - 速さを訴求する製品
    （右は製品Webサイトより）



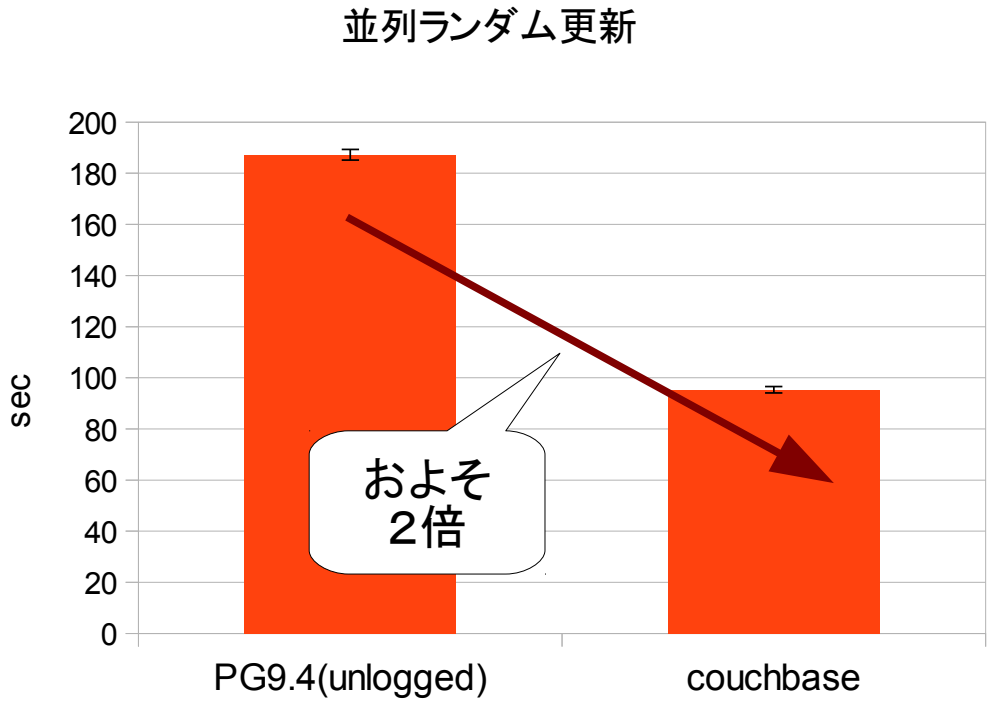
- 前項と同様「整数主キー + JSONカラム + GINインデックス」のテーブルで比較

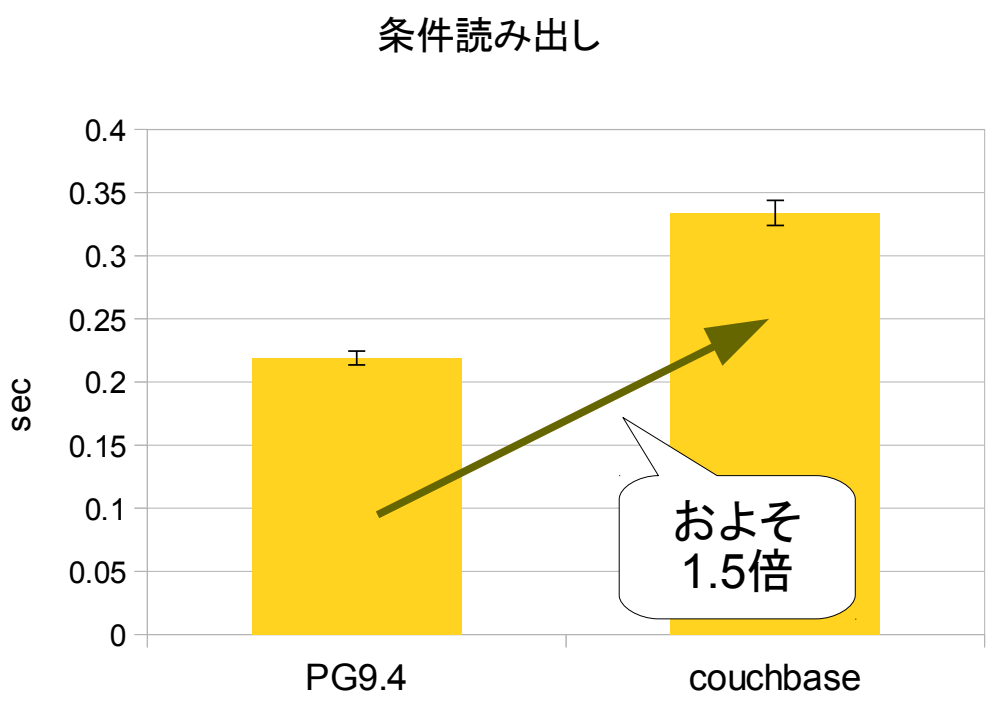


PG、couchbaseとも、大量データ投入専用コマンドの利用なし

PG 9.4 検証報告 Part1 : JSONB の性能 (4)

- 並列ランダム更新
  - ４並列で2件を主キーでランダム読み出し、１件を書き換え
- 条件読み出し
  - JSON内の属性で条件検索／10万件から 600件程度を取り出す
    - GINインデックスが使えるので PostgreSQL が高速





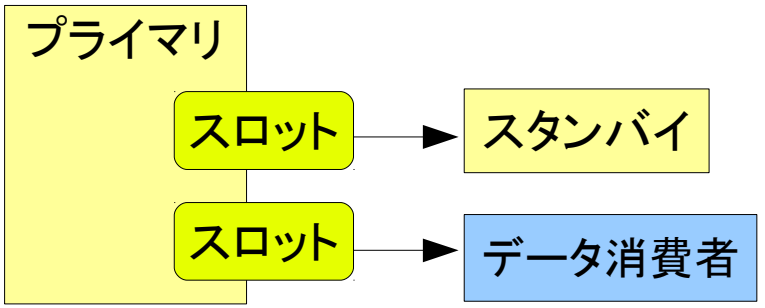
あくまで単体比較であって、NoSQL製品の売りは「スケールアウトのしやすさ」である点にも注意

この新機能は何ですか？

PG 9.4 検証報告 Part 2 : レプリケーションスロット (1)

レプリケーションスロット

＝ プライマリとスタンバイをつなぐ絆



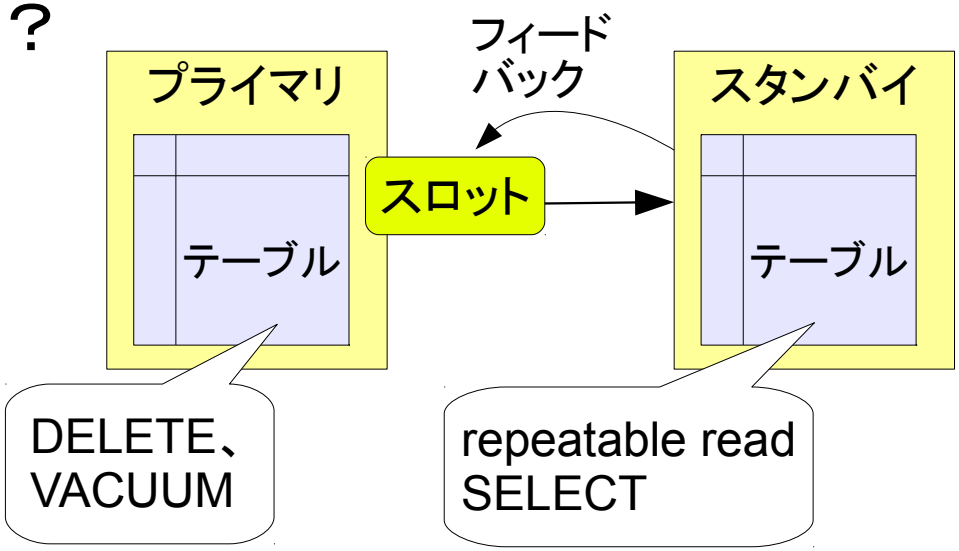
- レプリケーション状態や付帯データを保持管理する
  - これまで特定スタンバイに対応して状態を保持する場所が無かった

```
pg_create_physical_replication_slot('slot_name')
```

- physical レプリケーションスロット
  - スタンバイが必要としている WALファイルを削除してしまうのを防ぐ
  - スタンバイが参照しているデータの物理削除（VACUUM）を防ぐ
  - 従来の対策よりも優れた点がある
- logicalレプリケーションスロット
  - WALによる変更差分の論理表現を記録、蓄積してくれる

PG 9.4 検証報告 Part 2 : レプリケーションスロット (2)

- physicalレプリケーションスロットのアドバンテージ
  - コンフリクト対策、WAL削除対策が従来手段よりも優れている
  - 「どこまで必要」という情報がスロットに保持される
    - スタンバイがダウン中でも機能する
- レプリケーションコンフリクトとは？
  - 主としてスタンバイで参照しているものを削除して発生する
  - VACUUM による行物理削除、テーブル、データベース削除
- WAL削除の問題とは？
  - スタンバイが未だ参照したい WAL ファイルが、プライマリサーバ上で削除されてしまうこと



- 従来は？
  - wal_keep_segment、vacuum_defer_cleanup_age で固定値指定
  - restore_command で ファイル単位レプリケーションも併用

PG 9.4 検証報告 Part 2 : レプリケーションスロット (3)

- logical レプリケーションスロットの可能性

```
=# SELECT pg_create_logical_replication_slot('logi_slot', 'test_decoding');
```

```
=# CREATE TABLE t_test (id int primary key, v text);
=# INSERT INTO t_test
     SELECT g, left(md5(g::text),5) FROM generate_series(1,5) as g;
=# UPDATE t_test SET v = 'XXX' WHERE id = 3;
```

```
=# SELECT * FROM pg_logical_slot_get_changes('logi_slot', NULL, NULL);
 location | xid | data
----------+-----+-----------------------------------------------------
0/D907F60 | 2015 | BEGIN 2015
0/D91CEA8 | 2015 | COMMIT 2015
0/D91CEE0 | 2016 | BEGIN 2016
0/D91CEE0 | 2016 | table public.t_test: INSERT: id[integer]:1 v[text]:'c4ca4'
0/D91CFA8 | 2016 | table public.t_test: INSERT: id[integer]:2 v[text]:'c81e7'
0/D91D038 | 2016 | table public.t_test: INSERT: id[integer]:3 v[text]:'eccbc'
0/D91D0C8 | 2016 | table public.t_test: INSERT: id[integer]:4 v[text]:'a87ff'
0/D91D158 | 2016 | table public.t_test: INSERT: id[integer]:5 v[text]:'e4da3'
0/D91D268 | 2016 | COMMIT 2016
0/D91D490 | 2017 | BEGIN 2017
0/D91D490 | 2017 | table public.t_test: UPDATE: id[integer]:3 v[text]:'XXX'
0/D91D658 | 2017 | COMMIT 2017
```

PG 9.4 検証報告 Part 2 : SQL新機能 (1)

- 更新ビューのCHECKオプション

```
=> CREATE TABLE t_num (num int);
=> CREATE VIEW v_natural AS SELECT num FROM t_num WHERE num > 0
   WITH CASCADED CHECK OPTION;
=> INSERT INTO v_natural VALUES (10);      ← 正の数は格納できる
=> INSERT INTO v_natural VALUES (-100);    ← 負の数は定義外
ERROR: new row violates WITH CHECK OPTION for view "v_natural"
DETAIL: Failing row contains (-100).
```

- 集約クエリの FILTERオプション

```
SELECT to_char(date_trunc('month', ts), 'YYYY-MM') as month,
  coalesce(sum(amount) FILTER(WHERE iid = 1), 0) as sales_i1,
  coalesce(sum(amount) FILTER(WHERE cid = 1), 0) as sales_c1,
  coalesce(sum(amount), 0) as sales_all
FROM t_sale GROUP BY date_trunc('month', ts);
```

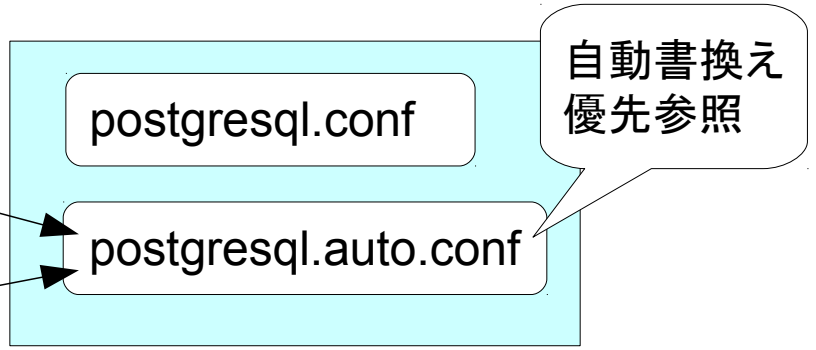
⇒ 実行プランにおいて集約とスキャンが一括して行われる

PG 9.4 検証報告 Part 2 : SQL新機能 (2)

- ALTER SYSTEM で設定変更

```
=# ALTER SYSTEM
   SET work_mem TO '10MB';

=# ALTER SYSTEM
   SET work_mem TO DEFAULT;
```



※ pg_ctl reload や pg_ctl restart に相当する処理は実行してくれない

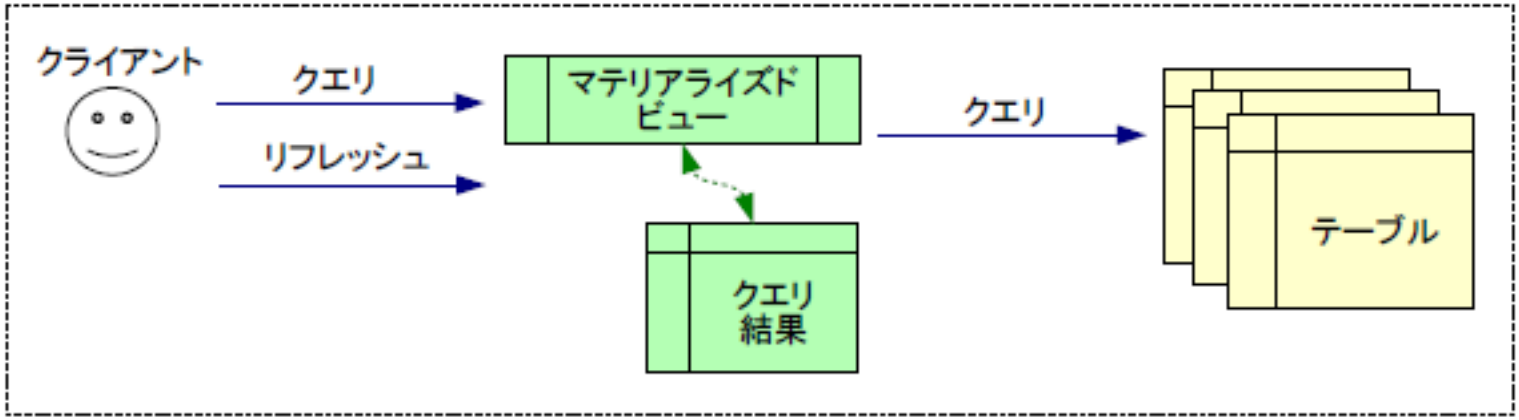
- ALTER TABLESPACE MOVE で一括移転

```
=# ALTER TABLESPACE pg_default MOVE TABLES  TO tblspc1;
=# ALTER TABLESPACE pg_default MOVE INDEXES TO tblspc2;
```

従来は テーブル、インデックスごとに ALTER コマンド実行が必要であった

PG 9.4 検証報告 Part 2 : SQL新機能 (3)

- REFRESH MATERIALIZED VIEW の CONCURRENTLYオプション
  - リフレッシュが、マテリアライズドビューへの SELECT と競合しない
  - マテリアライズドビュー上の ユニークインデックスが必要



- ALTER TABLE のロック競合が一部軽減
  - ... ALTER COLUMN ... SET STATISTICS ... など明らかにテーブルのデータ読み書きと関係ない

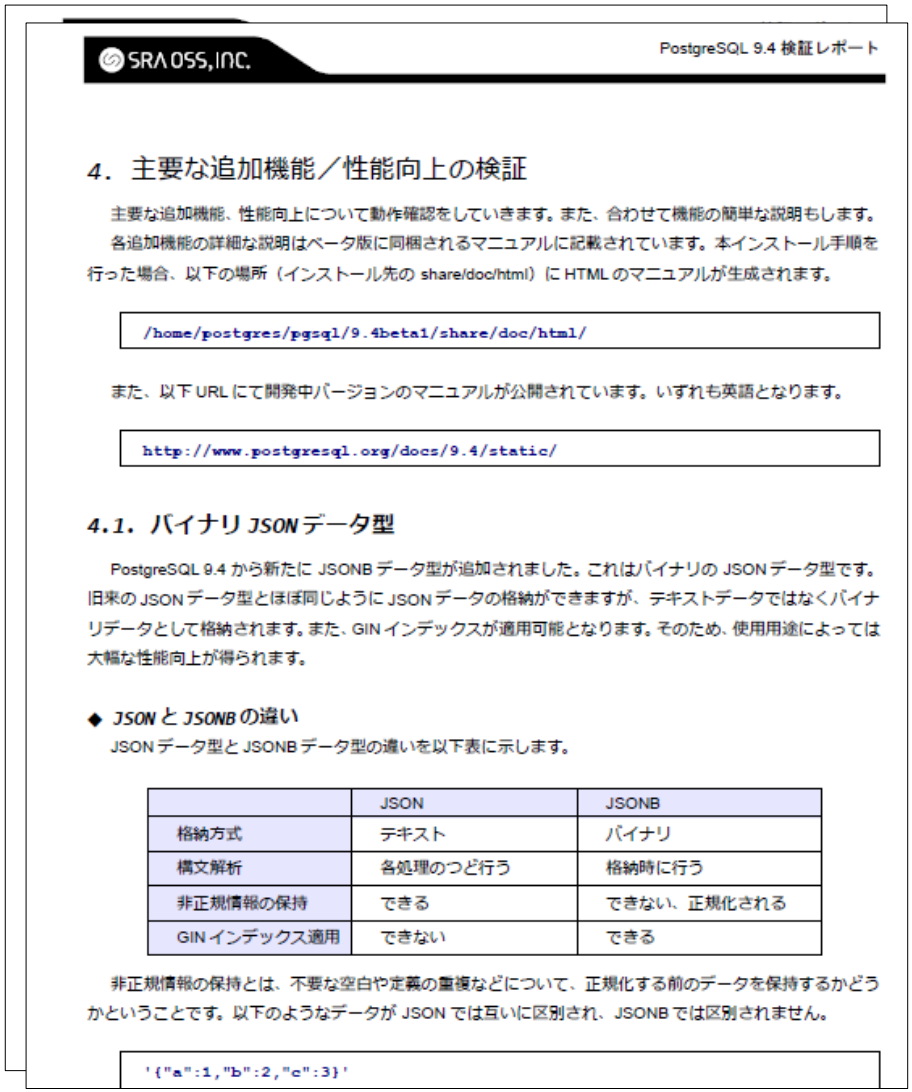
    ⇒ こういったオプションではテーブル読み書きと競合しない、SHARE UPDATE EXCLUSIVE モードに変更 （従来は ACCESS EXCLUSIVE モードで SELECT もブロックされてしまう）

レポート形式の資料

- 「PostgreSQL 9.4 に関する検証報告」

  - 本講演の内容を概ね含みます
  - 実行して試すことのできる検証報告文書となっています

SRA OSS, Inc. 日本支社
（http://www.sraoss.co.jp）
⇒［技術情報］
⇒［PostgreSQL］



SRA OSS,INC. PostgreSQL 9.4 検証レポート

4．主要な追加機能／性能向上の検証

主要な追加機能、性能向上について動作確認をしていきます。また、合わせて機能の簡単な説明もします。
各追加機能の詳細な説明はベータ版に同梱されるマニュアルに記載されています。本インストール手順を行った場合、以下の場所（インストール先の share/doc/html）に HTML のマニュアルが生成されます。

```
/home/postgres/pgsql/9.4beta1/share/doc/html/
```

また、以下 URL にて開発中バージョンのマニュアルが公開されています。いずれも英語となります。

```
http://www.postgresql.org/docs/9.4/static/
```

4.1．バイナリ JSON データ型

PostgreSQL 9.4 から新たに JSONB データ型が追加されました。これはバイナリの JSON データ型です。旧来の JSON データ型とほぼ同じように JSON データの格納ができますが、テキストデータではなくバイナリデータとして格納されます。また、GIN インデックスが適用可能となります。そのため、使用用途によっては大幅な性能向上が得られます。

◆ JSON と JSONB の違い

JSON データ型と JSONB データ型の違いを以下表に示します。

	JSON	JSONB
格納方式	テキスト	バイナリ
構文解析	各処理のつど行う	格納時に行う
非正規情報の保持	できる	できない、正規化される
GIN インデックス適用	できない	できる

非正規情報の保持とは、不要な空白や定義の重複などについて、正規化する前のデータを保持するかどうかということです。以下のようなデータが JSON では互いに区別され、JSONB では区別されません。

```
'{"a":1,"b":2,"c":3}'
```

オープンソースとともに

URL：http://www.sraoss.co.jp/
E-mail：sales@sraoss.co.jp
Tel：03-5979-2701