

PostgreSQL 9.6 新機能紹介

Noriyoshi Shinoda

July 2, 2016



自己紹介

篠田典良（しのだのりよし）



– 所属

- 日本ヒューレット・パカード株式会社 テクノロジーコンサルティング事業統括

– 現在の業務

- PostgreSQLをはじめOracle Database, Microsoft SQL Server, Vertica, Sybase ASE等 RDBMS全般に関するシステムの設計、チューニング、コンサルティング
- オープンソース製品に関する調査、検証
- Oracle Database関連書籍の執筆
- 弊社講習「Oracle DatabaseエンジニアのためのPostgreSQL入門」講師

– 関連する URL

- 「PostgreSQL 虎の巻」シリーズ
 - <http://h30507.www3.hp.com/t5/user/viewprofilepage/user-id/838802>
- Oracle ACEってどんな人？
 - <http://www.oracle.com/technetwork/jp/database/articles/vivadeveloper/index-1838335-ja.html>



Agenda

PostgreSQL 9.6 新機能紹介

1. PostgreSQL 9.6概要
2. Parallel Query
3. FOREIGN DATA WRAPPERの拡張
4. レプリケーション関連の新機能
5. SQL文
6. モニタリング
7. パフォーマンスと内部構造
8. Contribモジュール

まとめ





1. PostgreSQL 9.6概要

1. PostgreSQL 9.6概要

PostgreSQLの概要

- オープンソースで開発されているRDBMS
 - MySQLやFirebird等が仲間
- ライセンスはPostgreSQL License (≒BSD License)
- 活発な開発コミュニティ
 - The PostgreSQL Global Developer Group (<http://www.postgresql.org/>)
 - 日本PostgreSQLユーザ会 (<http://www.postgresql.jp/>)
 - PostgreSQL Enterprise Consortium (<http://www.pgecons.org/>)
- 最新バージョン
 - PostgreSQL 9.5 (9.5.3)
 - PostgreSQL 9.6 Beta 2 ← 今日のお話

1. PostgreSQL 9.6概要

PostgreSQLの歴史

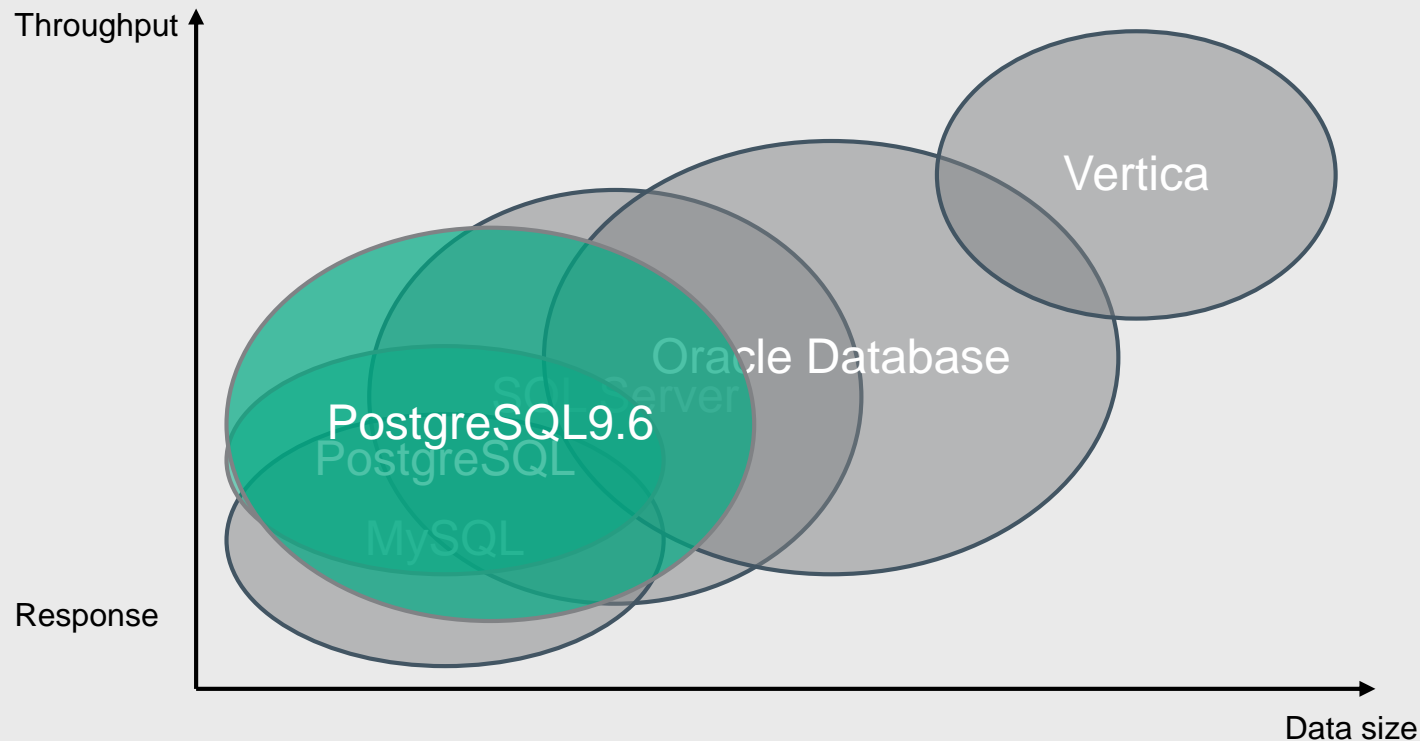
- 1974年 Ingres プロトタイプ
 - HPE NonStop SQL, SAP Sybase ASE, Microsoft SQL Serverの元になる
- 1989年 POSTGRES 1.0～
- 1997年 PostgreSQL 6.0～
 - GEQO, MVCC, マルチバイト
- 2000年 PostgreSQL 7.0～
 - WAL, TOAST
- 2005年 PostgreSQL 8.0～
 - 自動VACUUM, HOT, PITR
- 2010年 PostgreSQL 9.0～
 - レプリケーション, 外部表, JSON, マテリアライズド・ビュー
- 2016年5月 PostgreSQL 9.6 Beta 1
- 2016年6月 PostgreSQL 9.6 Beta 2



1. PostgreSQL 9.6概要

PostgreSQL 9.6の特徴

– スループットの向上と、モニタリングに関する新機能が充実しました。



1. PostgreSQL 9.6概要

本日ご説明する新機能

- Parallel Query
- FOREIGN DATA WRAPPERの拡張
- レプリケーション関連の新機能
 - Multiple Synchronous Replication
 - Synchronous Apply
- モニタリングの強化
- パフォーマンスと内部構造
- Contribモジュールの追加
 - bloom
 - pg_visibility





2. Parallel Query



Parallel Query

マルチプロセスによる並列処理

– 概要説明

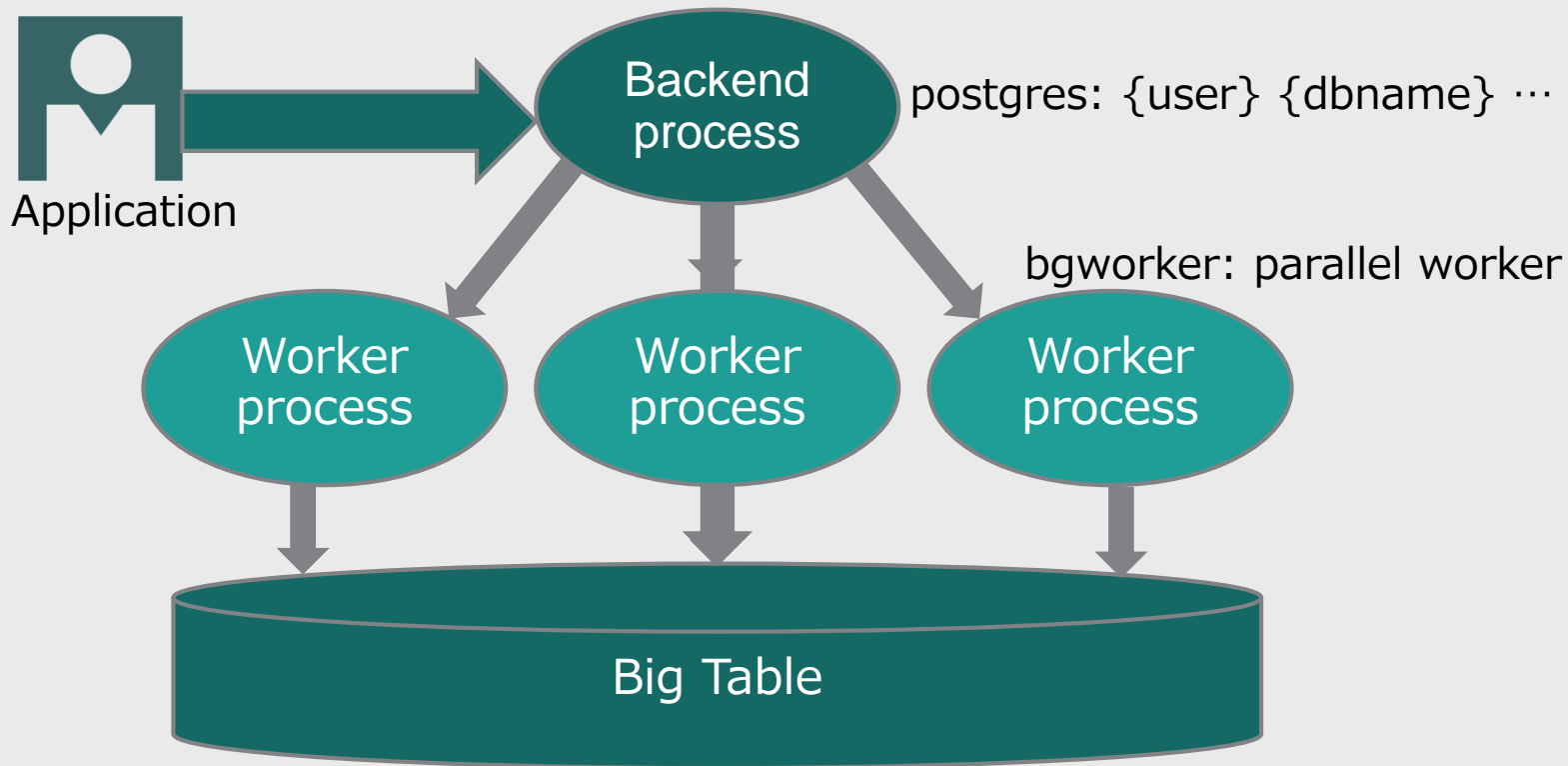
- 従来のバージョンでは、セッションに対応する単一のバックエンド・プロセスがすべてのSQL文を処理していました。
- PostgreSQL 9.6では、Seq Scan / Hash Join / Nested Loop Join / Aggregate処理を複数プロセスを使って処理できるようになりました。
- 並列処理を行うかどうかはコスト比較によって自動的に決定されます。

– アーキテクチャ

- パラレル処理はBackground Worker (9.3～) を使います。
- プロセス間通信にはDynamic Shared Memory (9.4～) を使います。
- パラレル処理API (9.5～) を使います。

Parallel Query

マルチプロセスによる並列処理



Parallel Query

並列度の決め方

- 並列度は基本的にはデータのサイズに依存します。
 - パラメータmin_parallel_relation_size（デフォルト8MB）以下の場合、パラレル処理は行われません。
 - データ量が3倍になるごとに並列度を追加します。
 - サーバーのCPU数は考えていません。
- 並列度の最大値は、MIN (max_worker_processes, max_parallel_workers_per_gather)となります。
- 実際の起動ワーカー数は実行中のワーカーを除外するため、実行計画作成時の計算よりも小さくなる場合があります。
- テーブル・オプションparallel_workersが指定されている場合はテーブルのサイズに依存せずに並列度が決定されます（最大 1024）。

```
postgres=> ALTER TABLE data1 SET (parallel_workers = 5);  
ALTER TABLE
```

Parallel Query

関連パラメーター

– 並列処理に関するパラメーター

パラメーター名	デフォルト値	説明（コンテキスト）
max_parallel_workers_per_gather	2	実行計画決定時の最大並列度（user）
parallel_setup_cost	1000	並列処理の開始コスト（user）
parallel_tuple_cost	0.1	並列処理のタプル処理コスト（user）
force_parallel_mode	off	並列処理の実行計画作成を強制（user）
min_parallel_relation_size	8MB	Parallel Queryを検討する最小サイズ（user）
max_worker_processes	8	ワーカー・プロセスの最大値（postmaster）
dynamic_shared_memory_type	posix	ダイナミック共有メモリーの種別（postmaster）

Parallel Query

制約

- リーダーのみアクセス可能
 - Temporary Table
 - CTE Scan
 - 外部テーブル（更新をサポートする場合以外）
 - **Restricted Parallel Safe関数**の使用
- シリアル処理になるケース
 - DELETE文 / UPDATE文の検索処理
 - 分離レベルがSERIALIZABLE
 - カーソルの使用
 - **Parallel Unsafe関数**の使用

Parallel Query

Parallel Unsafe Function

– Parallel Unsafe Function

- 実行計画全体がシリアルになります。
- `pg_proc.proparallel='u'` である関数が対象です。
- CREATE FUNCTION文で作成した関数のデフォルト状態です。

– Restricted Parallel Safe Function

- SQL文内で使われるとリーダー・プロセスが指定処理を行います。
- `pg_proc.proparallel='r'` である関数が対象です。

– 主なParallel Unsafe / Restricted Parallel Safe関数

カテゴリー	制限	関数名
シーケンス関連	Unsafe	nextval, currval, setval, lastval
LOB関連	Unsafe	lo_*, lread, lwrite
時刻関連	Restricted	age, now, statement_timestamp
乱数関連	Restricted	random, setseed

Parallel Query

Parallel Unsafe Function

```
postgres=> EXPLAIN SELECT * FROM data1 WHERE c1=100 ;  
          QUERY PLAN
```

Gather (cost=1000.00..107137.72 rows=1 width=11)

Workers Planned: 2

-> Parallel Seq Scan on data1 (cost=0.00..106137.62 rows=0 width=11)

Filter: (c1 = '100'::numeric)

(4 rows)

```
postgres=> EXPLAIN SELECT * FROM data1 WHERE c1=nextval('s1') ;  
          QUERY PLAN
```

Seq Scan on data1 (cost=0.00..229052.60 rows=1 width=11)

Filter: (c1 = (nextval('s1'::regclass))::numeric)

(2 rows)



3. FOREIGN DATA WRAPPERの拡張

FOREIGN DATA WRAPPERの拡張

Remote Push-down処理の追加

– FOREIGN DATA WRAPPERとは

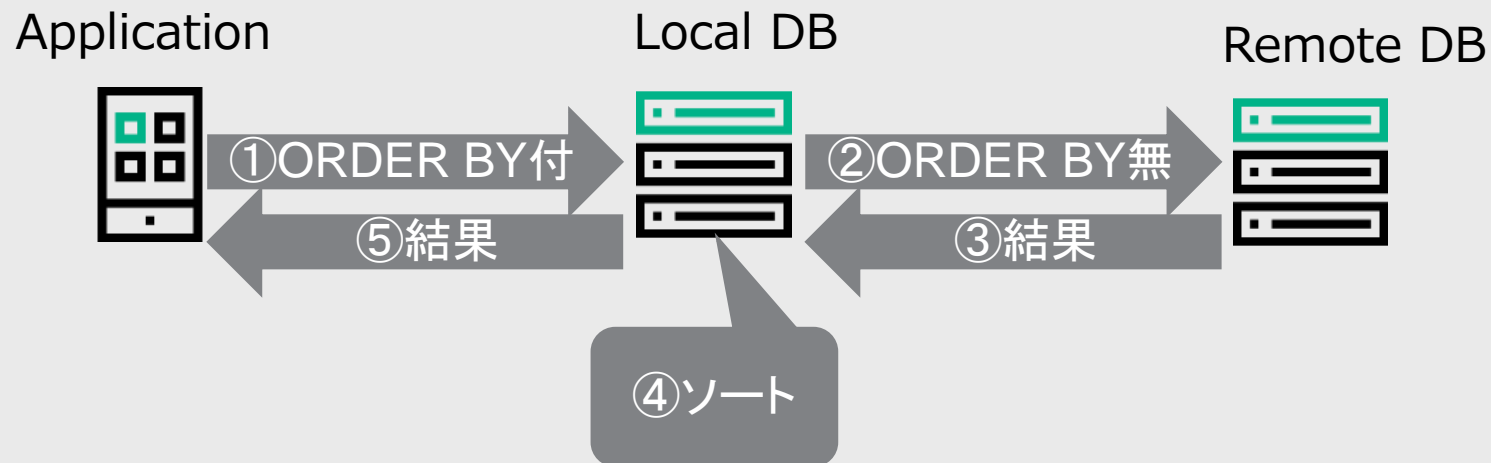
- 外部システム（リモートPostgreSQL, Oracle Database, ファイル等）にSQL文を発光することができるソフトウェアです。
 - 従来のバージョンでは、外部システムからデータをローカルに全部取得してから処理を行っていました。
- PostgreSQL 9.6ではいくつかの処理を外部システムで行うことができるようになりました。

操作	PostgreSQL 9.5	PostgreSQL 9.6
ソート	Local	Remote
更新	カーソル利用 / Tuple単位	直接実行 / SQL単位
結合	Local	Remote
集計	Local	Local

FOREIGN DATA WRAPPERの拡張

PostgreSQL 9.5以前

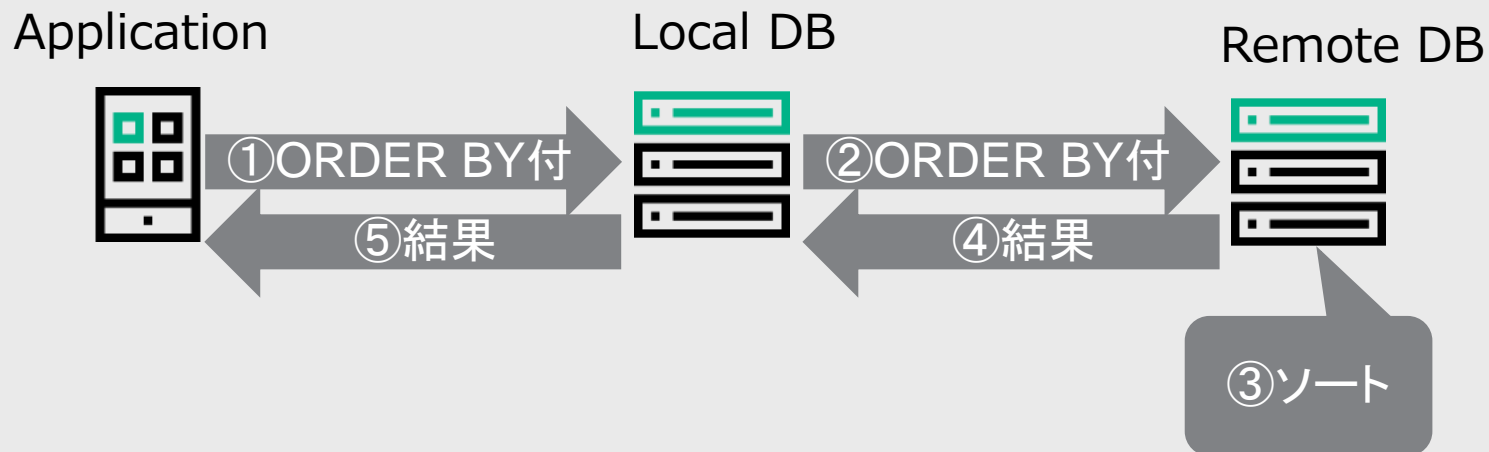
– ソートや結合は全データをローカル環境で実施



FOREIGN DATA WRAPPERの拡張

PostgreSQL 9.6

– ソートや結合はリモート環境で実施可能になりました



FOREIGN DATA WRAPPERの拡張

Join Push-downの実行計画

- PostgreSQL 9.6の場合、ORDER BY句はリモート・インスタンスで実行されます。

```
postgres=> EXPLAIN ANALYZE VERBOSE SELECT * FROM remote1  
ORDER BY 1 ;
```

QUERY PLAN

Foreign Scan on public.remote1 (cost=100.00..139.87 rows=871 width=70)
(actual time=1060.858..3216.021 rows=1000000 loops=1)

Output: c1, c2

Remote SQL: SELECT c1, c2 FROM public.remote1

ORDER BY c1 ASC NULLS LAST

Planning time: 0.370 ms

Execution time: 3257.071 ms

(5 rows)



4. レプリケーション関連の新機能

Multiple Synchronous Replication

複数インスタンスに対する同期レプリケーション

- マスター・インスタンスに対する変更情報をスレーブ・インスタンスに送信
 - 送信方法には「同期」と「非同期」があります。
 - 同期レプリケーション
 - リモート・インスタンスに更新情報が到達してからCOMMIT完了
 - 非同期レプリケーション
 - 先にCOMMITが完了し、リモート・インスタンスに更新情報送信

MASTER



同期／非同期

SLAVE #1



旧バージョンでは同期レプリケーション
できるインスタンスは1つだけ

同期／非同期

SLAVE #2



PostgreSQL 9.6では**複数インスタンス**に
対して**同期レプリケーション**可能

Multiple Synchronous Replication

複数インスタンスに対する同期レプリケーション

- synchronous_standby_namesの構文変更

```
num_sync (application_name1, application_name2, ...)
```

- num_sync部分には同期インスタンス数を指定します。

```
postgres=> SHOW synchronous_standby_names ;  
synchronous_standby_names
```

```
-----  
2 (standby1, standby2, standby3)  
(1 row)
```

```
postgres=> SELECT application_name, sync_state FROM  
pg_stat_replication ;
```

```
application_name | sync_state
```

```
-----+-----  
standby1         | sync  
standby2         | sync  
standby3         | potential  
(3 rows)
```


Synchronous Apply

WALの適用までCOMMIT待機

- synchronous_commitパラメーターに設定値 **remote_apply** 追加
 - スレーブ・インスタンスでWAL適用が完了するまでCOMMIT文を待機します。

設定値	Master	Slave memory	Slave wal	Slave apply
off	非同期			
local	同期	非同期		
remote_write	同期		非同期	
on	同期			非同期
remote_apply	同期			

- recovery_min_apply_delayとの併用
 - スレーブ・インスタンスのrecovery.confファイルにrecovery_min_apply_delay指定があるとCOMMIT文の完了が指定時間待機されます。



5. SQL文

COPY

COPY文の拡張

- COPY文にDELETE文とUPDATE文が指定可能

```
COPY (DELETE FROM table_name ... RETURNING *) TO 'file_path'  
COPY (UPDATE table_name SET ... RETURNING *) TO 'file_path'
```

- RETURNING *句が必須です。
- ファイルに出力されるレコード
 - DELETE文の場合は削除されたレコード
 - UPDATE文の場合は変更後のレコード

その他

その他のSQL文

- CREATE FUNCTION *func* PARALLEL {UNSAFE | RESTRICTED | SAFE}
- ALTER FUNCTION *func* PARALLEL {UNSAFE | RESTRICTED | SAFE}
- ALTER TABLE *table* ADD COLUMN IF NOT EXISTS ...
- ALTER TABLESPACE *tablespace* SET (*effective_io_concurrency* = *value*)
- CREATE EXTENSION *extension* CASCADE
- ALTER OPERATOR *operator* ... SET {RESTRICT | JOIN}





6. モニタリング

モニタリング

pg_stat_activity

– pg_stat_activityカタログの変更点

列名	データ型	変更	説明
waiting	boolean	削除	待機しているか
wait_event_type	text	追加	待機イベントの概要
wait_event	text	追加	待機イベントの名前

– wait_event_type列の値

値	説明
LWLockNamed	特定の軽量ロックによる待機
LWLockTranche	グループに対する軽量ロックによる待機
Lock	重量ロックによる待機（LOCK TABLE文等）
BufferPin	バッファに対するPIN待ち

モニタリング

pg_stat_wal_receiver

– wal receiver

- レプリケーション環境のスレーブ・インスタンス上で起動するプロセスです。
- プライマリから更新情報（WAL）を受け取り、スタンバイ環境を構築します。

– 新規作成されたpg_stat_wal_receiverカタログ

- スタンバイ・インスタンス上で検索します。
- WAL受信状況が確認できます。
- 従来はプライマリ・インスタンス上でWAL送信側のステータスのみ確認できました。



モニタリング

pg_stat_progress_vacuum

– VACUUM

- 更新前レコードを削除する処理です。
- 通常は自動的に実行されます（autovacuum = on）。

– 新規作成されたpg_stat_progress_vacuumカタログ

- VACUUMの状況を確認できます。
- VACUUM処理の実行中のみレコードを参照できます。
- max_dead_tuple列は、maintenance_work_memパラメーターで指定できるメモリー量に格納できるレコード数を示します。





7. パフォーマンスと内部構造

パフォーマンスと内部構造

アイドル・トランザクションの自動切断

- 何もしていないトランザクションを検知
 - 制限時間を過ぎたセッションを自動切断
- パラメータ `idle_in_transaction_session_timeout` を指定
 - 0 = タイムアウトしない (デフォルト値 / 従来と同じ動作)
 - 1 ~ 2,147,483,647 = タイムアウトまでのミリ秒

```
postgres=> SET idle_in_transaction_session_timeout = 1000 ;  
SET
```

```
postgres=> BEGIN;  
BEGIN
```

```
-- wait 2sec
```

```
postgres=> COMMIT ;
```

```
FATAL: terminating connection due to idle-in-transaction timeout  
server closed the connection unexpectedly
```

This probably means the server terminated abnormally
before or while processing the request.

The connection to the server was lost. Attempting reset: Succeeded.

パフォーマンスと内部構造


その他の追加されたパラメーター

パラメーター名	デフォルト値	説明
old_snapshot_threshold	0	スナップショットの生存期間
backend_flush_after	0	backendのFlushを強制する閾値
bgwriter_flush_after	512kB	bgwriterのFlushを強制する閾値
checkpoint_flush_after	256kB	checkpointのFlushを強制する閾値
syslog_sequence_numbers	on	SYSLOGにシーケンス番号を付与
syslog_split_messages	on	長いSYSLOGメッセージを分割
wal_writer_flush_after	1MB	wal writerのFlushを強制する閾値

パフォーマンスと内部構造

その他の変更されたパラメーター

パラメーター名	説明
log_line_prefix	数値を使ったタイムスタンプ (%n) が利用可能に
wal_level	archiveとhot_standbyがreplicaに統合
autovacuum_max_workers	最大値が262,143に縮小
max_connections	最大値が262,143に縮小
max_replication_slots	最大値が262,143に縮小
max_wal_senders	最大値が262,143に縮小
max_worker_processes	最大値が262,143に縮小
superuser_reserved_connections	最大値が262,143に縮小
wal_writer_delay	pg_settings.short_dsc列を変更



7. Contribモジュール

bloom

bloom filterを使ったインデックス

- 使用できるデータ型は integer, char, text です。
- 複数列に同時作成できます。
- 等価検索に使用できます。

```
postgres=# CREATE EXTENSION bloom ;
CREATE EXTENSION
postgres=> CREATE INDEX idx1 ON bloom1 USING bloom (c1, c2, c3) ;
CREATE INDEX
postgres=> EXPLAIN SELECT * FROM bloom1 WHERE c2 = 100000 ;
               QUERY PLAN
```

```
-----
Bitmap Heap Scan on bloom1 (cost=15348.00..15352.01 rows=1 width=20)
  Recheck Cond: (c2 = 100000)
    -> Bitmap Index Scan on idx1 (cost=0.00..15348.00 rows=1 width=0)
          Index Cond: (c2 = 100000)
(4 rows)
```

postgres_fdw

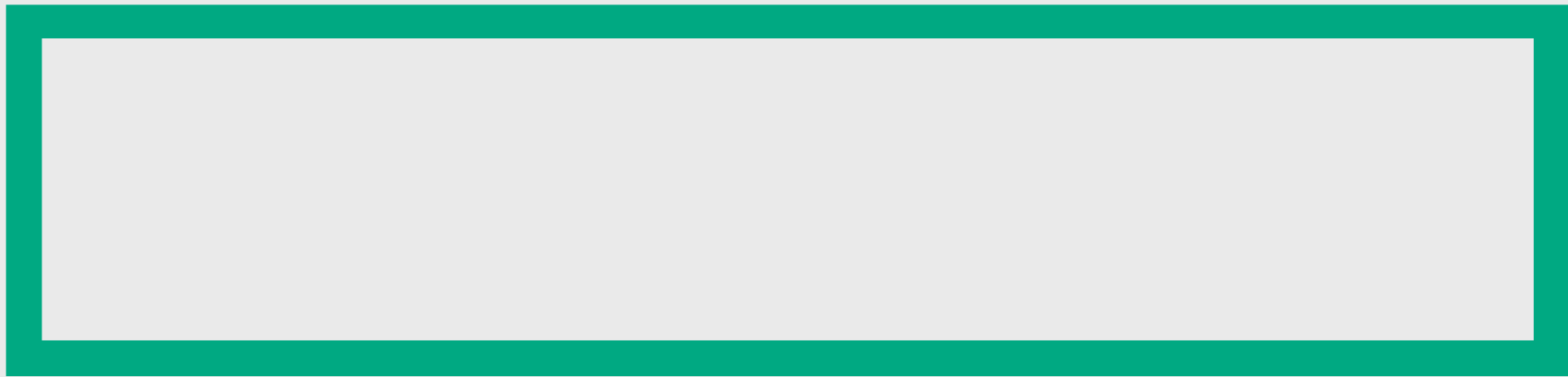
オプションが追加されました

– fetch_sizeオプション

```
postgres=# CREATE SERVER host1 FOREIGN DATA WRAPPER
postgres_fdw OPTIONS
    (host 'host1', port '5432', dbname 'demodb', fetch_size '300') ;
CREATE SERVER
postgres=# CREATE FOREIGN TABLE table1(c1 INT, c2 VARCHAR(5))
    SERVER host1 OPTIONS(fetch_size '300') ;
CREATE FOREIGN TABLE
```

– extensionsオプション

```
postgres=# CREATE SERVER host1 FOREIGN DATA WRAPPER
postgres_fdw OPTIONS
    (host 'host1', port '5432', dbname 'demodb', extensions 'hstore') ;
CREATE SERVER
```



まとめ



まとめ

- PostgreSQL 9.6には、魅力的な新機能が数多く採用されました。
 - Parallel Seq Scan
 - FOREIGN DATA WRAPPERの改善
 - モニタリングの改善
 - レプリケーション関連の新機能
 - Freeze Mapをはじめとするパフォーマンス改善
- 参考URL
 - Commitfests
<http://commitfest.postgresql.org/>
 - めこ@横浜さんのブログ
http://d.hatena.ne.jp/nuko_yokohama/
 - Michael Paquierさんのブログ
<http://michael.otacoo.com/>
 - What's New in PostgreSQL 9.6
<https://wiki.postgresql.org/wiki/NewIn96>



Hewlett Packard
Enterprise

Thank you

noriyoshi.shinoda@hpe.com