



Wonderful PostgreSQL!

SRA OSS, Inc. 日本支社
取締役支社長
石井 達夫

自己紹介

- OSSの開発とビジネスに携わっています
 - OSS活動
 - PostgreSQLのコミット
 - PostgreSQL用のクラスタソフト Pgbpool-II の開発
 - ビジネス
 - PostgreSQLなどのOSSのサポート
 - PowerGres製品の製造、販売
 - PostgreSQLトレーニング



本日のアジェンダ

- PostgreSQL前夜
- PostgreSQL誕生
- PostgreSQL現在
- PostgreSQL未来

PostgreSQL前夜

- PostgreSQLの先祖は、カリフォルニア大学バークレー校での研究プロジェクトで開発された“Postgres”まで遡ることができる
- Michael Stonebraker 教授が指導
- 1994年まで開発継続
- used by assorted “bold and brave” early users
(様々な「大胆で勇気のある」初期ユーザに使われています)
- “THE IMPLEMENTATION OF POSTGRES” より
- 最終バージョンは4.2



Postgresの特徴

- オブジェクト指向の考え方を取り入れた
 - クラス、継承、OID
- 抽象データ型、拡張可能な型システム、配列
- 独自の問い合わせ言語 POSTQUEL
 - ちなみに、”libpq” の ”pq” は ”Post” “Quel” のそれぞれ頭文字を取っている
- fast path
 - バックエンド関数の直接実行
- プログラミング言語中立
 - 今では当たり前だが、当時は新しい考え方だったようだ
- ルールシステム
- タイムトラベル
 - retrieve (EMP.salary) using EMPT['2017-11-03']
where EMP.name = “Sam”

Postgresの実装(1)

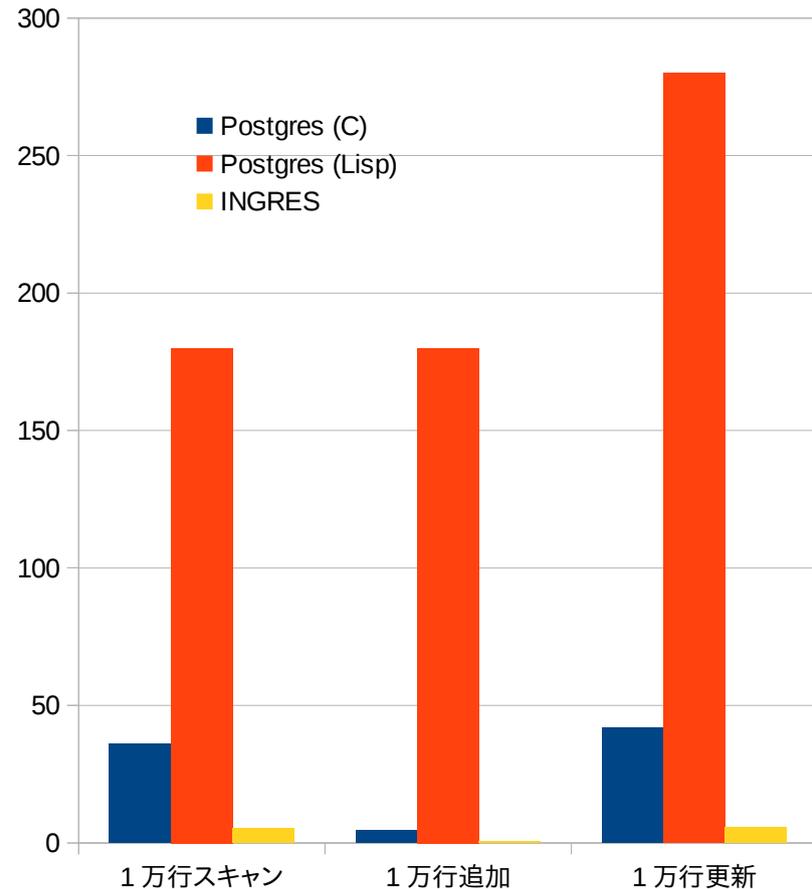
- 追記型のテーブルとトランザクションステータスビットを格納したファイルのみ。トランザクションログなし
 - 動機は、「普通のトランザクションログを使ったDBMSでは面白くないから」
 - トランザクションログがないことによる性能の悪さは、不揮発性メモリでカバーできると考えていたようだ
 - VACUUMの性能問題に悩まされていたようだ

Postgresの実装(2)

- 実装言語の選択
 - SmallTalk: 遅いから却下
 - C++が良さそうだったが、安定したコンパイラがないので辞めた
 - CはINGRESで使っていたので別な言語にしたかった
 - で、結局Lispを選択
 - そしてすぐに後悔(バッファマネージャをLispで書くのは難しい)
 - そこでLispとCを併用することに(Lispで17,000、Cで63,000)
 - しかし、「Lispメモリ喰い問題」に直面
 - 「4MBも」メモリを使っている
 - また、デバッグも困難
 - 結局Lisp部分をCに書き直し、全部で90,000行のCコードになった

Postgresの実装(3)

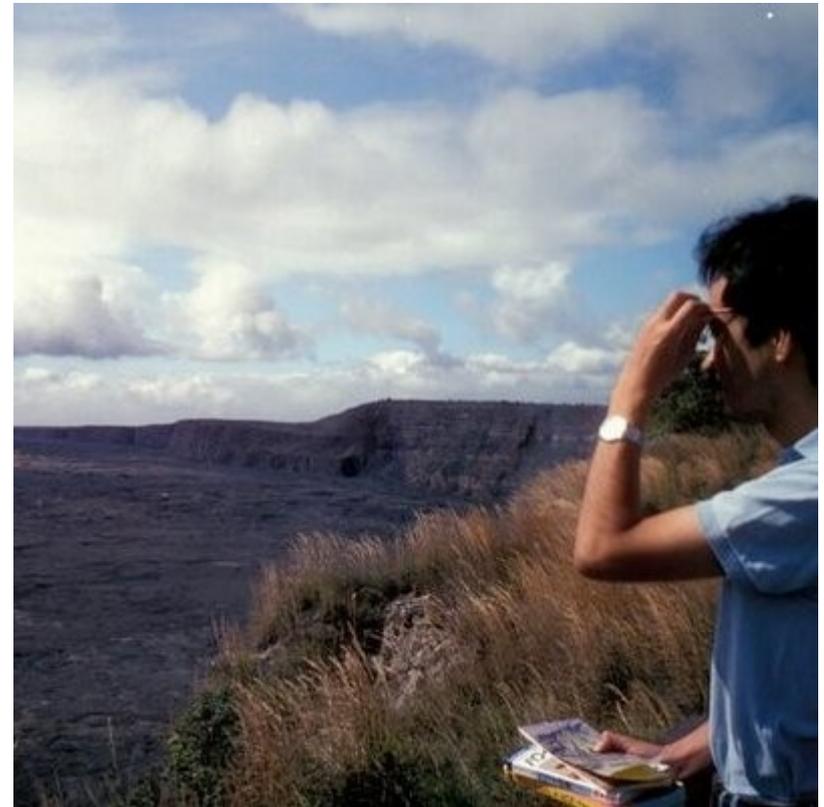
- Postgresの性能
 - C実装とLisp実装の比較
 - 参考のためにINGRESとの比較も
- Cに書き換えてかなり性能が向上したが、それでも商用DB(INGRES)には遠く及ばない性能だった



縦軸は、クエリ実行時間(秒)

当時私は何をしていたかと言うと...

- 1991-1993までハワイのホノルル大学で研究をサポートするエンジニアをしていた
- そこではじめてPostgresと出会う
- 最初に使ったRDBがPostgresで、最初に使った問い合わせ言語がPostquelだった



Postgres95へ

Jolly Chen



Andrew Yu



- Postgresプロジェクト
1994年で終了
- Postgresプロジェクト
に参加していた
Andrew Yu氏とJolly
Chen氏が、Postgres
をベースにPostgres95
を開発

Postgres95での変更点

- 問い合わせ言語を Postquel から、SQLに変更
- より洗練されたコードに書き直し、不要部分を削除して高速化
- 若干のSQL強化(Having句の追加)
- “monitor” に代わって、“psql” の実装

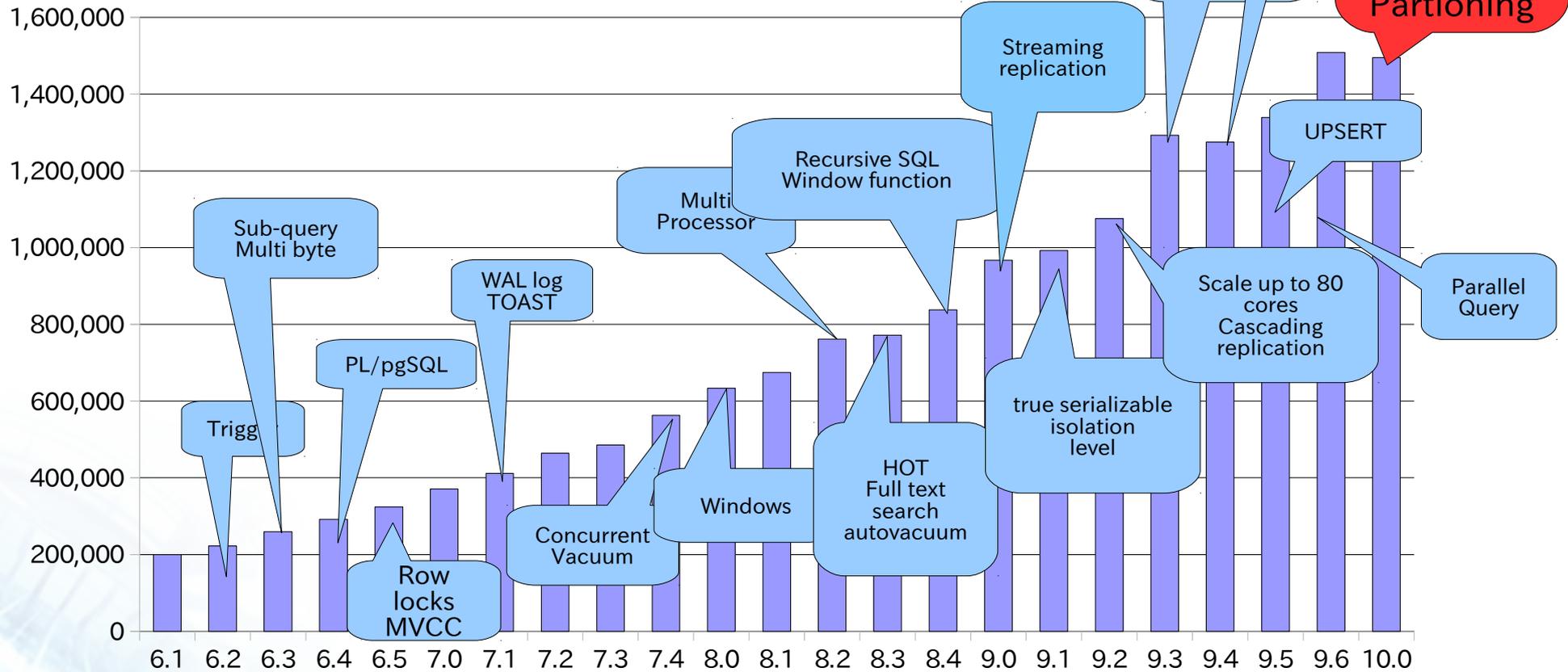
当時私は何をしていたかということ...

- 帰国してしばらくは遊びでPostgresをいじる程度
 - M68kアーキテクチャや、MIPSアーキテクチャのマシンへの移植
 - 当時PostgresはSparcアーキテクチャでしか動かなかった
 - そこへ、現在のJPUGの方から「Postgres95が出た」ということを教えていただき、Postgres95のメーリングリストとWebページを立ち上げた
 - これがのちの「pgsql-jp」メーリングリスト

PostgreSQLの誕生

PostgreSQL History

lines of code



1997

2017

1997年1月、PostgreSQL 6.0誕生

- リリースチームは、Marc G. Fournier, Bruce Momjian 氏など
- Berkely Postgres が 4.0、Postgres95を5.0と考えると、PostgreSQLは6.0から開始すべき、と考えた
 - 自分たちがオリジナルではなく、過去の資産の上に貢献を重ねていくという考え方の表れ
 - 後のPostgreSQLの発展の原動力となった方針



Postgres95チームと、初期のPostgreSQL
コアメンバー

PostgreSQLの愉快的なバグたち

- 修正パッチを送った中で、特に印象深いバグ
 - postmasterに無制限に接続できて、そのうち落ちてしまう
 - フロントエンドとバックエンドのendianが違うと、通信できない
 - ラージオブジェクトの途中でseekしてwriteすると、データがおかしくなる

当初のPostgreSQLになかったもの

- 日本語が使えない
 - パッチを当てて対応(6.3で対応)
- PL/pgSQLがない
 - PL/pgTCLで代用(PL/pgSQLが追加されたのは6.4)
- サブクエリがない
 - SQL関数で代用(6.3で対応)
- トリガがない
 - ルールを使う(6.2で対応)
- 行ロックがない(6.5で対応)
- MVCCがない(6.5で対応)
- トランザクションログ(WAL)がない(7.1で対応)
- Windowsで動かない(8.0で対応)

MVCCの導入

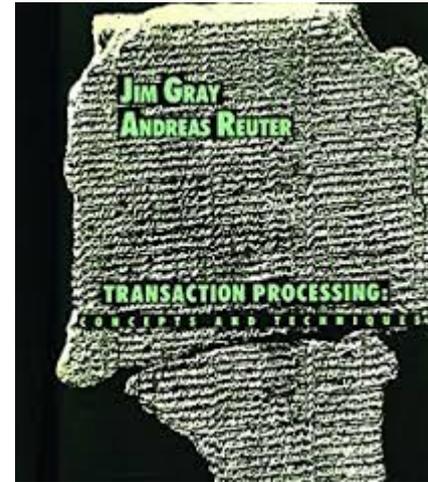
- それまでのPostgreSQLには「タイムトラベル」機能があった
 - 追記型のアーキテクチャを活かし、過去データを、時刻を指定してSELECTできる機能
- これを削除し、代わりにMVCCを実装。過去データはユーザからは見えなくなった
- MVCC導入によるメリット
 - 高いトランザクション並列性
 - 行ロックの導入
 - 当時商用DBでもエスカレーションなしの行ロックは少なかった
 - トランザクション分離レベルの導入
 - READ COMMITTED

トランザクションログの導入以前

- それまでのPostgreSQLは、トランザクションをコミットする度に、データファイルを同期書き込みしていた
- その結果、書き込み性能が非常に低かった
 - 1000万件のデータロードに12時間以上かかる
 - pgbenchで20TPS以下の性能
 - 2 CPUまでしかスケールしない。4 CPUではかえって性能低下

WALをたった一人で実装した伝説のプログラマ: Vadim

- Jim Grayの“Transaction Processing”を読んだだけでWALを実装した
- 議論は1998年からスタート。実装完了したのは7.1(2001年)
 - MVCC込で設計していた



日本PostgreSQLユーザー会の誕生

- 1999年、世界で最初の大規模PostgreSQLユーザーグループとして誕生
- pgsql-jpで集まった方を母体に結成
- セミナー、マニュアル翻訳、懇親会など、活発な活動を開始
- 個人的にもいろいろあった年
 - PostgreSQLユーザー会理事長就任
 - PostgreSQLコミッタに就任
 - 本業でもPostgreSQLを取り扱う

PostgreSQL Anniversary Summit

- 2006年カナダのトロントで、PostgreSQLの誕生10周年を記念して開催
- はじめてPostgreSQL開発者・ユーザがface to faceで集合
- この後、世界各地で開催される国際PostgreSQLカンファレンスの先駆け
- PostgreSQLの開発を加速した
- 参加レポートはこちら
 - <http://itpro.nikkeibp.co.jp/article/COLUMN/20060727/244542/>



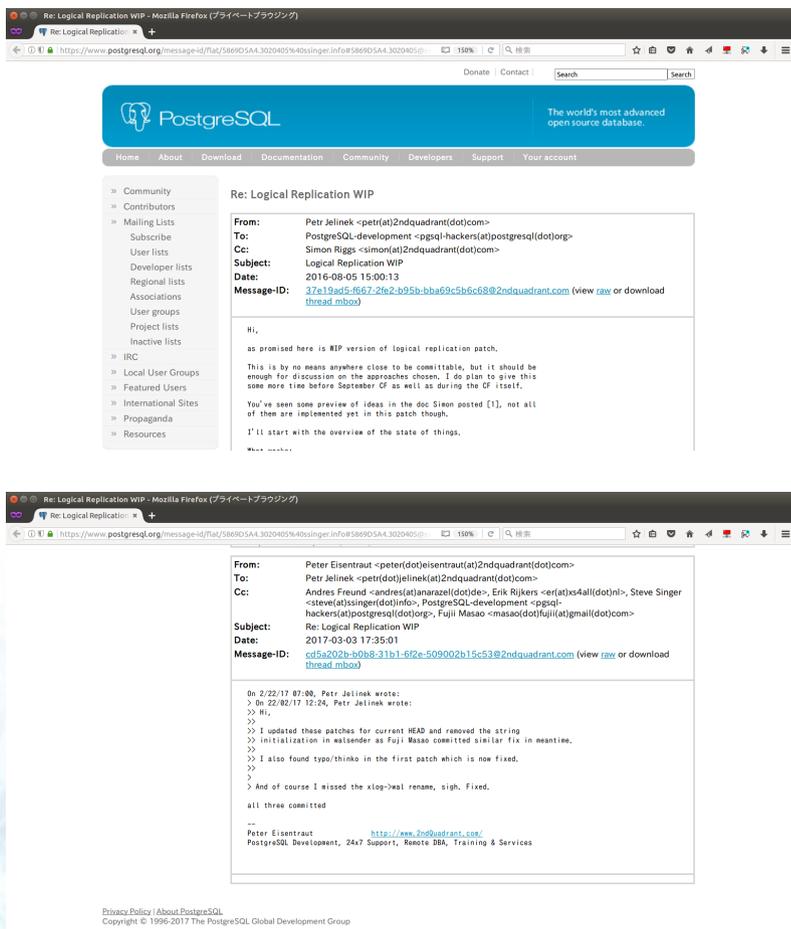
PostgreSQLの開発体制

- 特定の企業がソースコードを管理しない集団管理体制
 - 特定企業の方針転換・栄枯盛衰に影響されないメリット
- 最終決定は数人の「コアメンバー」が行う
- gitリポジトリへのコミット権限を持つ「コミッタ」
- それ以外の「開発者」
- ほとんどの決定は、開発者間の議論で行う
 - 開発者は約100人ほど
 - それ以外にレビューに参加した人などを含めると、300人以上が開発に参加している
- ほとんどの議論は公開されているが、一部クローズドな議論を行うグループがある
 - セキュリティ、リリース日程など

PostgreSQLの開発は 「みんなで徹底議論」が基本

- メーリングリストで機能仕様・設計について徹底議論
 - 時には1-2年も議論が続く
 - その機能は本当に必要か
 - その機能を追加して保守が困難にならないか
 - その機能を追加することにより性能が犠牲にならないか
 - 議論が長々と続いても、スレッドを容易に追えるような仕掛けがある

長い議論スレッドの例 (ロジカルレプリケーション)



- 2016/8/5にWIPパッチ投稿
- 2017/3/3にパッチコミット
- この間222通のメールのやり取り

Commit Fest

- パッチの投稿、レビュー、コミットを管理する仕組み
- パッチが投稿されてもレビューされずに放置されることもある状態を改善するために2008年ころから開始
- 仕組み
 - 2ヶ月毎にCommit Festを開催、パッチの投稿、レビューを行い、可能ならばコミットする
 - コミットされなかったパッチは次回Commit Festに持ち越される
 - そのメジャーリリース用の最終Commit Fest(前回は2017/3)でコミットされなかったパッチは、次のメジャーリリースに持ち越し
 - 専用のWebアプリで管理

Commitfest 2017-09

Search/filter Shortcuts ▾

Status summary: Needs review: 74. Waiting on Author: 32. Ready for Committer: 43. Committed: 84. Moved to next CF: 6. Rejected: 8. Returned with Feedback: 9. Total: 256.

Active patches

Patch	Status	Author	Reviewers	Committer	Latest activity	Latest mail
Bug Fixes						
Fix the optimization to skip WAL-logging on table created in same transaction	Needs review	Heikki Linnakangas (heikki), Michael Paquier (michael-kun), Kyotaro Horiguchi (horiguti)	Michael Paquier (michael-kun), Kyotaro Horiguchi (horiguti)	alvherre	2017-09-13 09:11	2017-09-14 06:34
Failure at replay for corrupted 2PC files + reduce window between end-of-recovery record and history file write	Ready for Committer	Heikki Linnakangas (heikki), Michael Paquier (michael-kun)	Alexander Korotkov (smagen), Takayuki Tsunakawa (maumau)		2017-04-08 14:03	2017-09-02 12:17
replace broken GetExistingLocalJoinPath with CreateLocalJoinPath	Ready for Committer	Etsuro Fujita (efujita)	Robert Haas (rhaas), Tom Lane (tgl), Ryan Murphy (murftown), Ashutosh Bapat (ashutoshbapat)		2017-09-19 12:19	2017-09-19 12:19
Fix a bug that can prevent standby from restarting	Needs review	Kyotaro Horiguchi (horiguti)	Venkata Balaji Nagothi (vbn)		2017-04-07 06:32	2017-09-07 03:33
Fix off-by-one in logical slot resource retention	Waiting on Author	Craig Ringer (ringerc)	Aleksander Alekseev (a.alekseev)		2017-03-09 03:19	2017-09-15 11:26
WAL send timeout and log sequence		Dmitry Fedorov (fedorod)	Kyotaro Horiguchi (horiguti), Michael Paquier (michael-kun)		2017-09-07 06:32	2017-09-07 06:32

Statement timeout behavior in extended queries

Edit Comment Status

Title	Statement timeout behavior in extended queries
Topic	Server Features
Created	2017-02-22 04:06:37
Last modified	2017-09-19 02:41:53 (1 week, 3 days ago)
Latest email	2017-09-19 02:42:38 (1 week, 3 days ago)
Status	<p>2017-09: Committed</p> <p>2017-03: Moved to next CF</p>
Authors	Tatsuo Ishii (ishii)
Reviewers	Takayuki Tsunakawa (maumau) Become reviewer
Committer	Andres Freund (andresfreund)
Links	
Emails	<p>Statement timeout behavior in extended queries ✕</p> <p>First at 2017-02-22 02:50:44 by Tatsuo Ishii <ishii at sraoss.co.jp></p> <p>Latest at 2017-09-19 02:42:38 by Tatsuo Ishii <ishii at sraoss.co.jp></p> Attach thread

Statement timeout behavior in extended queries - Mozilla Firefox

Statement timeout behavior in extended queries x +

https://commitfest.postgresql.org/14/1021/ 120% 検索

Status
 2017-09: Committed
 2017-03: Moved to next CF

Authors
 Tatsuo Ishii (ishii)

Reviewers
 Takayuki Tsunakawa (maumau) Become reviewer

Committer
 Andres Freund (andresfreund)

Links

Emails
[Statement timeout behavior in extended queries](#) x Attach thread
 First at 2017-02-22 02:50:44 by Tatsuo Ishii <ishii at sraoss.co.jp>
 Latest at 2017-09-19 02:42:38 by Tatsuo Ishii <ishii at sraoss.co.jp>
 Latest attachment (0001-Rearm-statement_timeout-after-each-executed-query-v2.patch) at 2017-09-05 02:30:28 from Tatsuo Ishii <ishii at sraoss.co.jp> +

History

When	Who	What
2017-09-19 02:41:53	Andres Freund (andresfreund)	Closed in commitfest 2017-09 with status: Committed
2017-09-19 02:41:53	Andres Freund (andresfreund)	Changed committer to andresfreund
2017-04-04 06:12:23	Takayuki Tsunakawa (maumau)	New status: Ready for Committer
2017-04-03 18:25:59	Andres Freund (andresfreund)	Closed in commitfest 2017-03 with status: Moved to next CF
2017-04-03 08:32:45	Tatsuo Ishii (ishii)	New status: Needs review

Edit Comment Status

PostgreSQLの現在： 重要な機能追加

- ストリーミングレプリケーション
- パラレルクエリ
- 論理レプリケーション

レプリケーション、クラスタリングの 必要性

- PostgreSQLサーバ1台の性能が上がってくると、人間欲が出てくるもの
 - DBクラッシュ時に業務が止まってしまうのは困る
 - かといって、バックアップから戻すのは、非常に時間がかかる
 - レプリケーションで解決→可用性の向上
 - DB性能が足りない
 - 複数台のDBサーバを束ねて性能を向上できないか



ストリーミングレプリケーション 登場以前のレプリケーション技術

- Pgpool-IIのネイティブレプリケーション
 - SQL文を複数サーバに投げることによってレプリケーションする
- Slony-I
 - 更新トリガを利用して更新差分を転送してレプリケーションする
- dblink
- 問題点
 - 性能が出ない
 - すべてのデータオブジェクトをレプリケーションできるわけではない
 - PostgreSQL組み込みのレプリケーション技術ではない

ストリーミングレプリケーションの登場

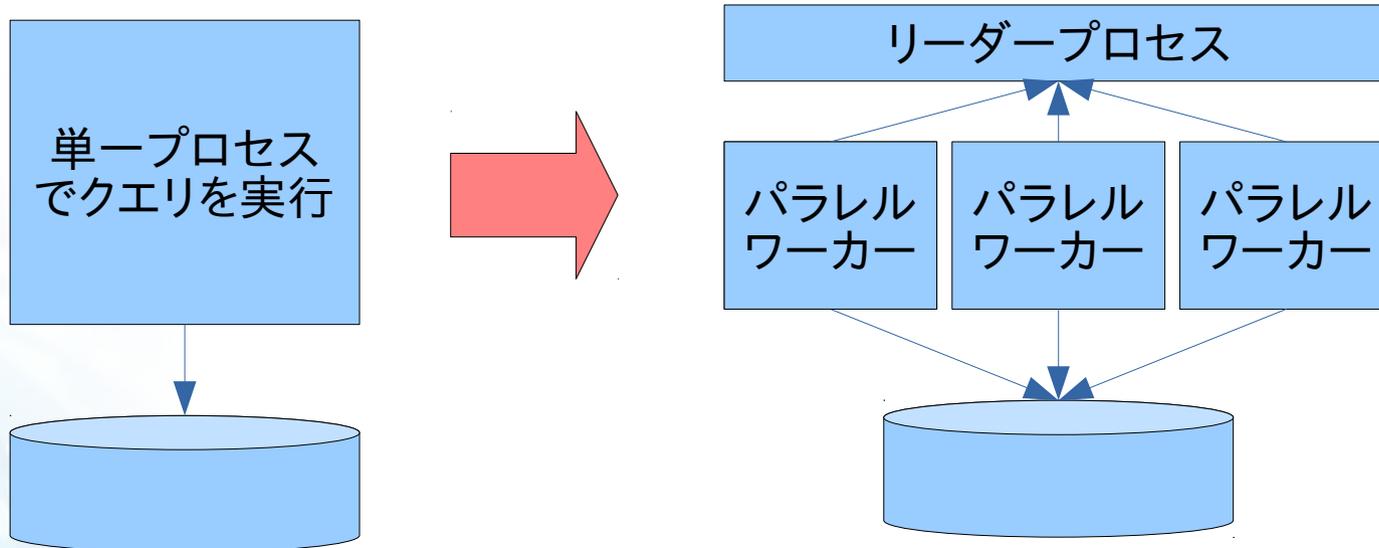
- PostgreSQL 9.0 (2010)で実装
- 既存のレプリケーション技術に対する優位性
 - 組み込みレプリケーション技術
 - 性能的に優れている
 - ほとんどすべてのオブジェクトをレプリケーションできる

ストリーミングレプリケーションの 継続的な改良

- スタンバイ昇格の際に、データを失わない
 - 同期レプリケーションの実装
- レプリケーション遅延のコントロールと一貫性の保証
 - `synchronous_commit = 'remote_apply'`

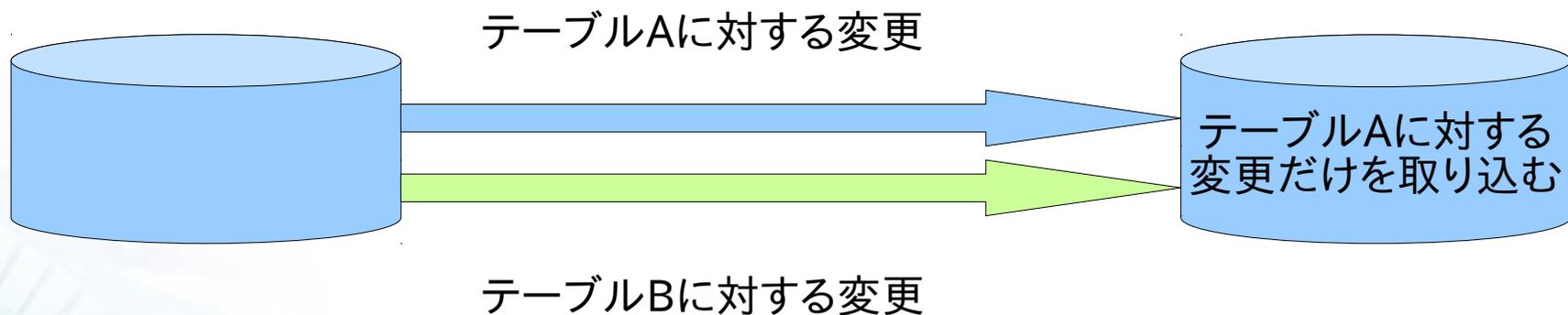
パラレルクエリの登場

- 単一プロセスでクエリを実行するアーキテクチャから、複数のワーカーがクエリを並列に実行するアーキテクチャへの変更
- 十分なメモリとCPUがある環境での検索クエリに威力を発揮



論理レプリケーションの登場

- 変更内容を論理的に後から操作できる形でログとして保存し、そこから特定の変更を取り出して適用する

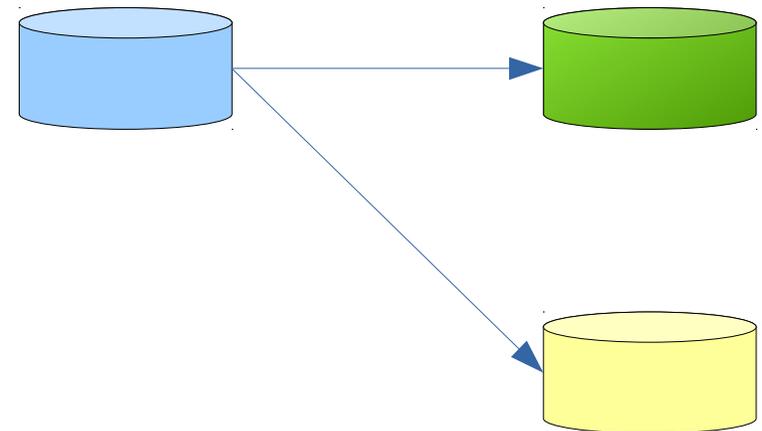


論理レプリケーションの ユースケース

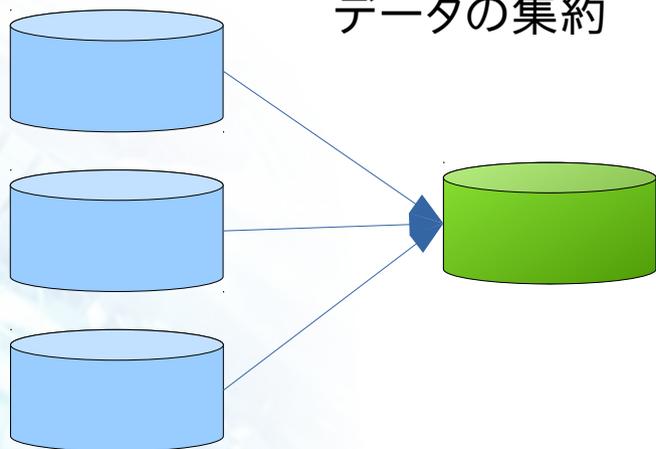
バージョンアップ



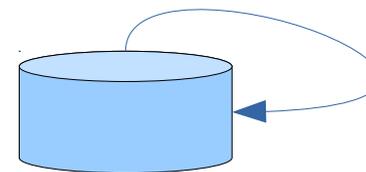
ストリーミングレプリケーション
との併用



データの集約



同一サーバ内での
別DBへのレプリケーション



PostgreSQLの将来（期待をこめて）

- 双方向レプリケーション
- 更新処理へのパラレルクエリの拡張
- 複数ノードをまたがったパラレルクエリ
- エグゼキュータへのJITコンパイル導入
- スキーマ変数
- JOINを含むビューも更新可能
- 図を含むドキュメント

参考URL

- PostgreSQL
 - <http://www.postgresql.org>
- Pgpool-II
 - <http://pgpool.net>
- Pgproto
 - <https://github.com/tatsuo-ishii/pgproto>

謝辞

- スライドで使用した写真の一部は、Bruce Momjian 氏からお借りしたものです。貴重な写真をご提供いただき、感謝します。

Thank you!

