PGECons
PostgreSQL Enterprise Consortium

PostgreSQL 9.6での性能強化と大規模DBへの適用性向上

2016年度活動成果報告

PostgreSQLエンタープライズ・コンソーシアム
WG1（新技術検証WG）

アジェンダ

- WG1(新技術検証ワーキンググループ)のご紹介
  - 活動領域と活動スタイル
  - 活動テーマのご紹介
  - 活動報告書のご案内
  - メンバー(参加企業)
- 活動報告
  1. 定点観測(スケールアップ検証[参照系・更新系])
  2. パラレルクエリ
  3. Pgpool-II検証
  4. JSON/JSONB
  5. 全文検索
  6. VACUUM改善
  7. 2016年度活動をふりかえって
- 付録
  - 今回使用した検証環境について

WG1のご紹介

活動領域と活動スタイル

- 活動領域
  - 「大規模基幹業務に向けたPostgreSQLの適用領域の明確化」の中で特に「性能」について調査
  - 2016年度から「新技術検証WG」と名称を変え、PostgreSQLの新機能やトレンドを踏まえた活動テーマを選定
- 活動スタイル
  - 毎年のPostgreSQLの新バージョンにあわせて調査テーマを選定
    - PGEConsセミナーのアンケート結果も参考に
  - テーマごとに担当メンバを決めて初期検討・実機検証を実施
    - 実機検証は1か月程度
    - 測定データを共有の上で、PostgreSQLの内部構造を調査して課題・問題を洗い出す
  - 調査結果を元に報告書を作成
    - 担当メンバが執筆
    - WGメンバによるピアレビューで品質を担保

WG1の今年度の活動テーマ

PostgreSQL 9.6での性能強化と大規模DBへの適用性向上

1. 定点観測(スケールアップ検証[参照系・更新系])
   - 多コアCPUでのPostgreSQL 9.6の到達点を検証
2. パラレルクエリ
   - 多コアCPUを使って新機能の実力を検証
3. Pgpool-II検証
   - レプリケーション方式の性能比較
4. JSON/JSONB
   - IoTを想定し、JSONやJSONB型の適切な使い方を調査
5. 全文検索
   - 3種類の代表的な検索手法(インデックス)の比較
6. VACUUM改善
   - PostgreSQL 9.6のXID凍結処理の改善について調査

過去のWG1活動テーマ(PostgreSQL 9.5)

- スケールアップ検証(参照系・更新系)[定点観測]
  多コアCPU(72)で9.5の検索・更新性能を9.4と比較して検証
- Parallel Vacuum検証
  Vacuumを行うワーカ数を変えた場合の性能特性を検証
- BRIN INDEX 検証
  btree、パーティショニングとの挿入・参照性能の比較、
  BRINの性能特性を検証
- OS比較検証
  RHEL 6系と7系との性能への影響、ファイルシステムやI/Oスケジューラの違いによる性能特性を調査

過去のWG1活動テーマ(PostgreSQL 9.4)

- スケールアップ検証(参照系)[定点観測]

  多コアCPU(60)で9.4の検索性能の検証
  (9.3との比較、page checksumの性能影響)

- スケールアップ検証(更新系)/WAL改善

  多コアCPU(60)で9.4のWAL改善による更新性能検証

- 仮想化環境(KVM)検証

  KVMを使用した環境での性能特性への影響を検証

- コンテナ環境(Docker)検証

  Dockerを使用した環境での性能特性への影響を検証

過去のWG1活動テーマ(PostgreSQL 9.2/9.3)

- **スケールアップ検証**

  多コアCPU(80)サーバでの参照性能の到達点を検証

- **スケールアウト検証**

  ストリーミングレプリケーション、pgpool-IIレプリケーション、Postgres-XCの特性を検証

- **パーティショニング検証**

  パーティションテーブルへの投入・検索・メンテナンス性能検証

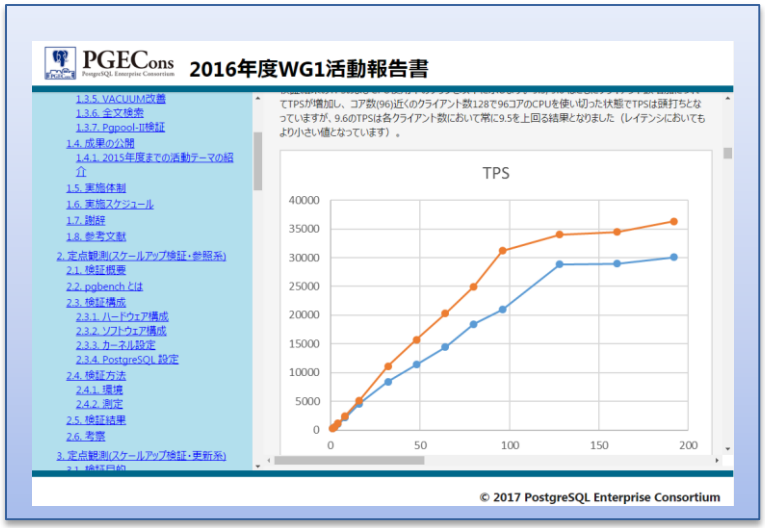
- **ハードウェア活用(SSD)検証**

  SSD採用時の性能向上を配置パターン別、アクセスプラン別に検証

活動報告書のご紹介

- 2016年度からHTMLで閲覧できます
  https://pgecons-sec-tech.github.io/tech-report/works_2016.html

- 検証テーマごとに詳細に解説
  - 検証環境の具体的なハード・ソフト構成
    - postgresql.confやベンチマークプログラムのパラメータなど、性能改善のヒントとなるノウハウも豊富
  - 検証結果データ
    - 一部のテーマでは、詳細な分析に用いたプロファイリングのデータを記載
  - 詳細な分析
    - ソースコード解析による、ボトルネックの考察など



WG1参加メンバ

- SRA OSS, Inc.日本支社
- NTTテクノクロス株式会社
- 日本電気株式会社（主査）
- 日本電信電話株式会社
- 日本ヒューレット・パッカード株式会社
- 富士通株式会社

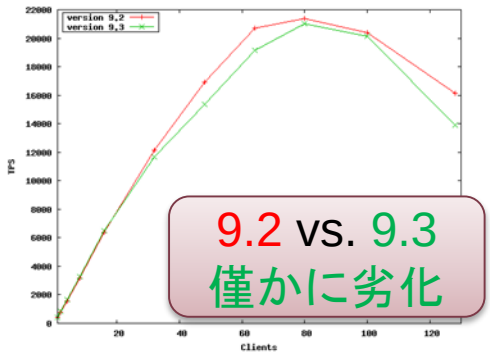
(企業名50音順・敬称略)

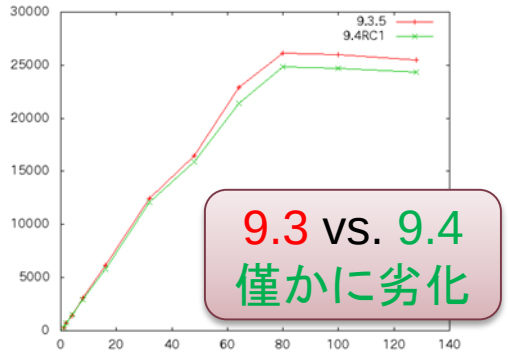
活動報告1

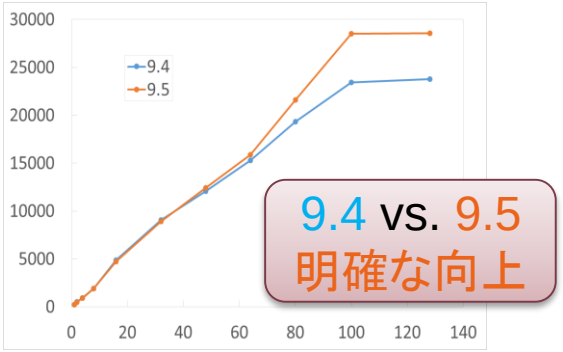
定点観測/スケールアップ検証

スケールアップ検証 [参照系] (定点観測)

- 目的
  - メニーコアCPU上での PostgreSQL 9.6 のスケーラビリティを検証
  - PostgreSQLの性能改善の傾向を知る
    - PGECons発足当初 (2012年度、PostgreSQL 9.2) から継続的に実施 (定点観測)
    - 直前のバージョンである PostgreSQL 9.5 との比較を実施
- 過去の結果
  - バージョン 9.2 の時点でスケール性は確認できていた
  - バージョン 9.5 で明確な最大スループットの向上が確認された



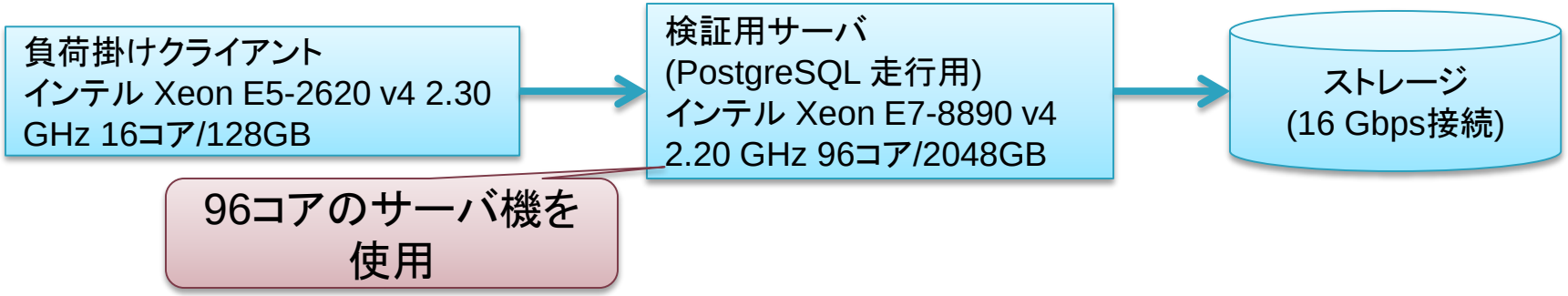




検証方法[参照系]

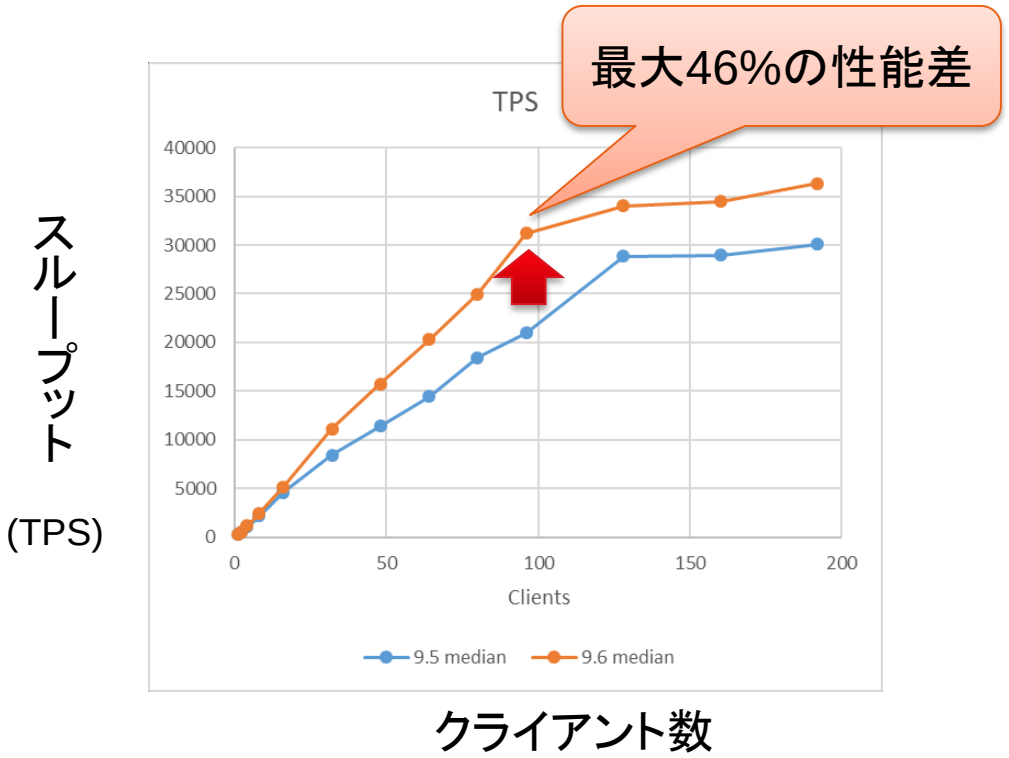
- CPUに十分な負荷が掛かる条件でpgbenchを実行
  - オンメモリでの走行
    - スケールファクタ1,000(DBサイズは15GB)
    - 事前に pg_prewarm を実行した
  - SQL1文の負荷を大きく
    - 1回の実行で1万行取得するカスタムスクリプトを使用
  - 多数のクライアントから同時にリクエスト
    - クライアント数を変化(1～192)させスループットを測定

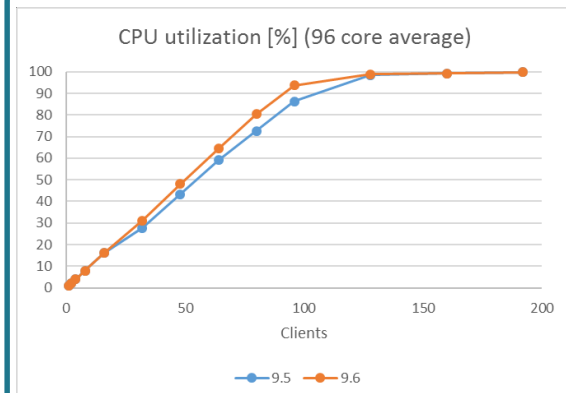
検証環境1を使用



測定結果［参照系］

- 9.6はクライアント数192で最高性能36,300TPSに達した
  - CPU利用率はほぼ100%となり、有効にCPUを利用できている
  - 9.5も同条件で最高性能である30,000TPSに達した
  - 9.6は対9.5で最大46%の性能向上





CPU利用率は9.6, 9.5ともほぼ100%に達しており、有効に使用している

結果まとめ

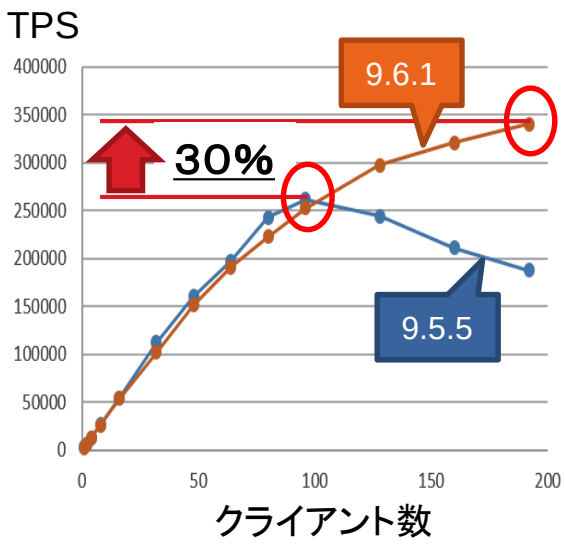
- [参照系]
  - 9.5と比較し、9.6は多クライアント接続におけるスループットの向上が確認できた
    - 最大46%の性能向上を確認
    - 比較的少ないクライアント数からスループット差が確認できた

定点観測(スケールアップ検証[更新系])概要

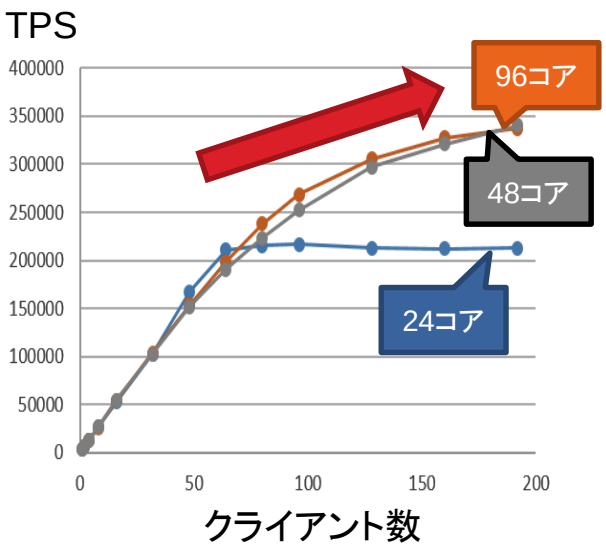
- 検証の目的
  1. PostgreSQLの更新系のCPUスケーラビリティの達成状況
  2. PostgreSQL 9.5から9.6の更新性能改善の確認
- 測定条件・環境
  - マシンは定点観測(スケールアップ検証[参照系])と同じ
  - 測定ツールとして、pgbenchを利用し、SF=7000(約100GB)、FILLFACTOR=80のDBを作成
  - pgbench_accountsへのUPDATEのみとする
  - データベースクラスタ(/data)とWAL(/pg_xlog)を分割した2ボリュームを使用
  - サーバのコア数とpgbenchのクライアント数を変更して測定
  - 測定中にCHECKPOINTを発生させないように設定

9.5.5/9.6.1 スケールアップ検証結果

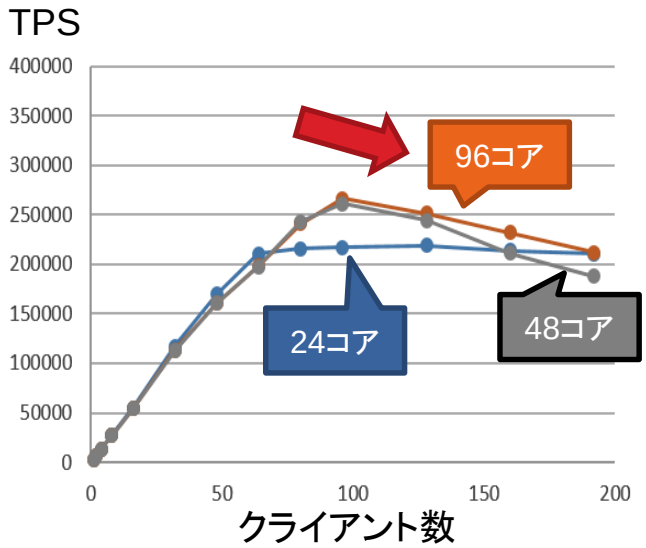
96コアのスループット比較



9.6.1のスループット



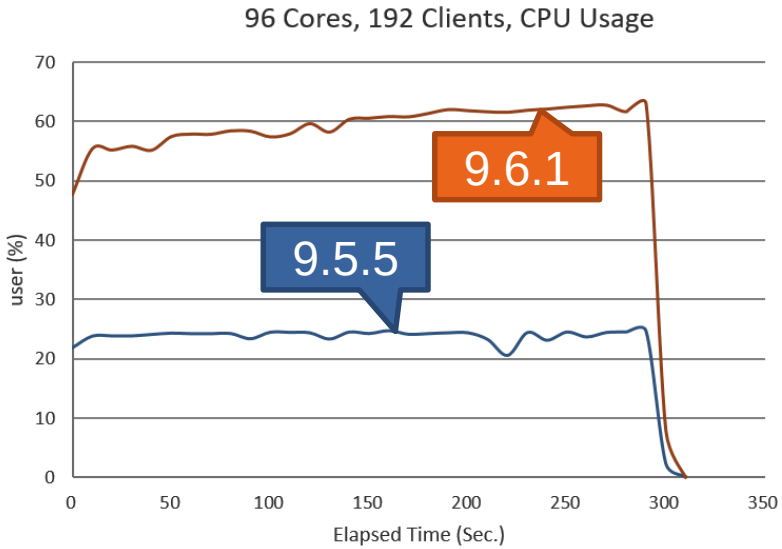
9.5.5のスループット



- 9.6.1では9.5.5よりスループットのピーク値が約30%向上
- 96コアでは9.5.5ではクライアント数が96前後でスループットが頭打ちになっているが、9.6.1では192までスループットが伸びることを確認

考察・まとめ

- 9.6.1は9.5.5と比較して、<u>更新処理においてスループットが向上していることが確認できた</u>
- 192クライアント、96コアでのCPUの利用率を見ると9.6.1がCPUをより利用できていることが分かる
- 9.6では、以下のロック競合の改善が行われたことが性能に起因していると考えられる
  - ProcArrayLock：
    複数プロセスが同時にコミットするときに利用されるロック
  - ClogControlLock:
    トランザクションの読み書きをする際に利用するロック



活動報告2

パラレルクエリ 検証

PostgreSQLにおけるパラレルクエリ概要

PostgreSQL 9.6では以下の3処理が実装

- **パラレルスキャン**
  SELECT文で発生するシーケンシャルスキャンのみパラレル処理可能
- **パラレル結合**
  ネステッド・ループ結合、ハッシュ結合がパラレル処理可能
- **パラレル集約**
  SUM関数、AVG関数等の集計処理がパラレル処理可能。
  ただし、Window関数による集約はパラレル処理されません。

**PostgreSQL 10 ではインデックス・スキャンや
マージ結合もパラレル処理可能となる予定**

パラレルクエリ実行プロセスと主な関連パラメータ

- パラレル処理の実行時には、ワーカープロセスがバックグラウンドプロセスとして起動される。
- 起動ワーカー数はクエリ対象となるテーブルの大きさと、ワーカー数を制御するパラメータによって決定。
- デフォルト値ではパラレル処理は無効となっている。

パラメータ	デフォルト値	説明
max_parallel_workers_per_gather	0	一つのクエリで起動できるワーカー数の最大値を指定
max_worker_processes	8	システム全体で起動できるワーカー数の最大値を指定
min_parallel_relation_size	8MB	パラレル処理を行うテーブルの最小サイズを指定。この値を超えると、テーブルサイズが3倍となるごとに起動されるワーカー数がひとつ増える。

検証目的

- 検証A：パラレルクエリの並列度と処理性能の関係を確認
  - テーブルサイズから計算される並列度まで性能向上する？
  - 並列度をさらに大きくした場合、効果がある？
  - 大きすぎると性能劣化する？
- 検証B：パーティションによる性能向上との比較
  - パーティション化するとパラレルクエリの並列度が低くなる
  - パーティションによる絞り込みの効果と比較
- 検証C：テーブル結合時の挙動確認
  - 並列度の異なるテーブルを結合するとワーカはどう起動される？

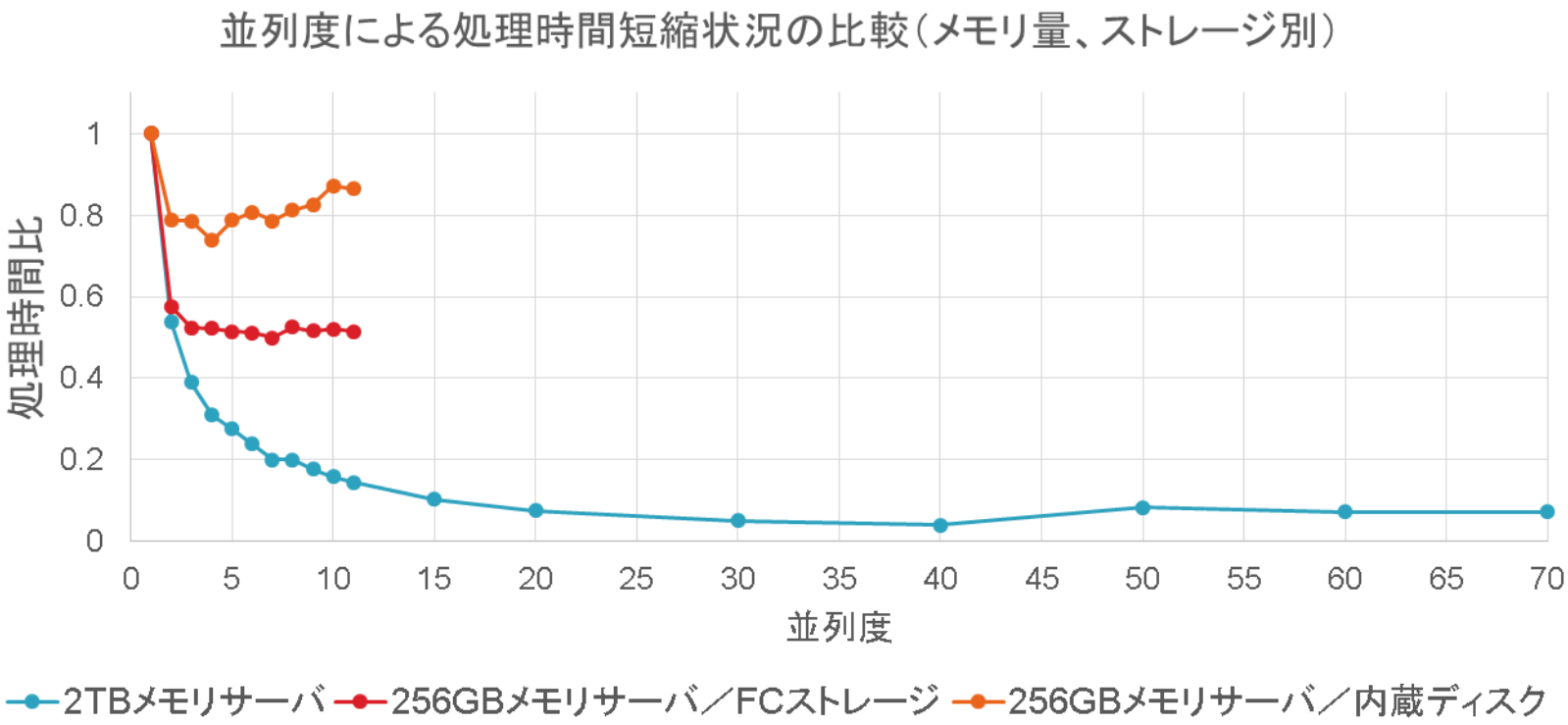
本日は時間の都合で検証Cは割愛します

測定環境／方法

- 定点観測で使用しているサーバに加え、2CPU(28core)／256GBメモリのサーバの2台を使用
- 検証A
  - 一日768万行×366日分のデータを保持するaccess_logテーブル(約230GB)を用意
  - 768万行(一日分相当)のデータを集計するクエリを実行
  - パラレルクエリの並列度を変化させ、処理時間を測定
  - 2TBメモリサーバ、256GBメモリサーバで測定
  - 256GBサーバではデータファイルをFCストレージに配置した場合と内蔵ディスクに配置した場合の2パターンを測定
- 検証B
  - access_logテーブルを日単位でパーティション化したテーブルへのクエリと単一テーブルへのクエリの処理時間を比較
  - 768万行(一日分相当)のデータを集計するクエリについて、パーティションプルーニングが無効、および有効な2パターンの処理時間を測定

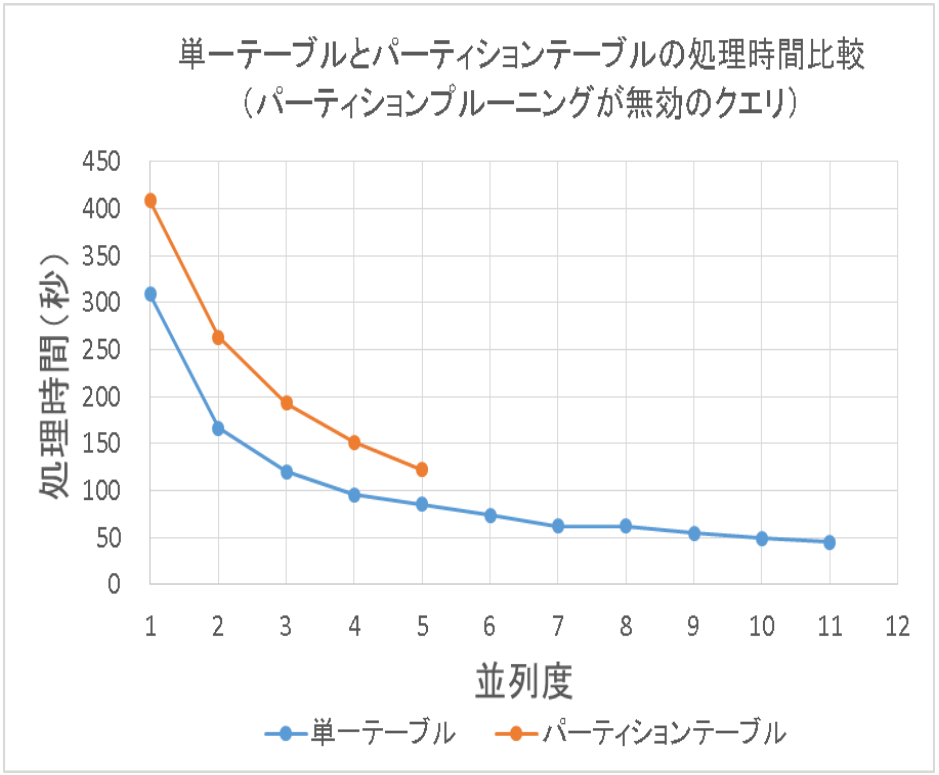
検証A測定結果

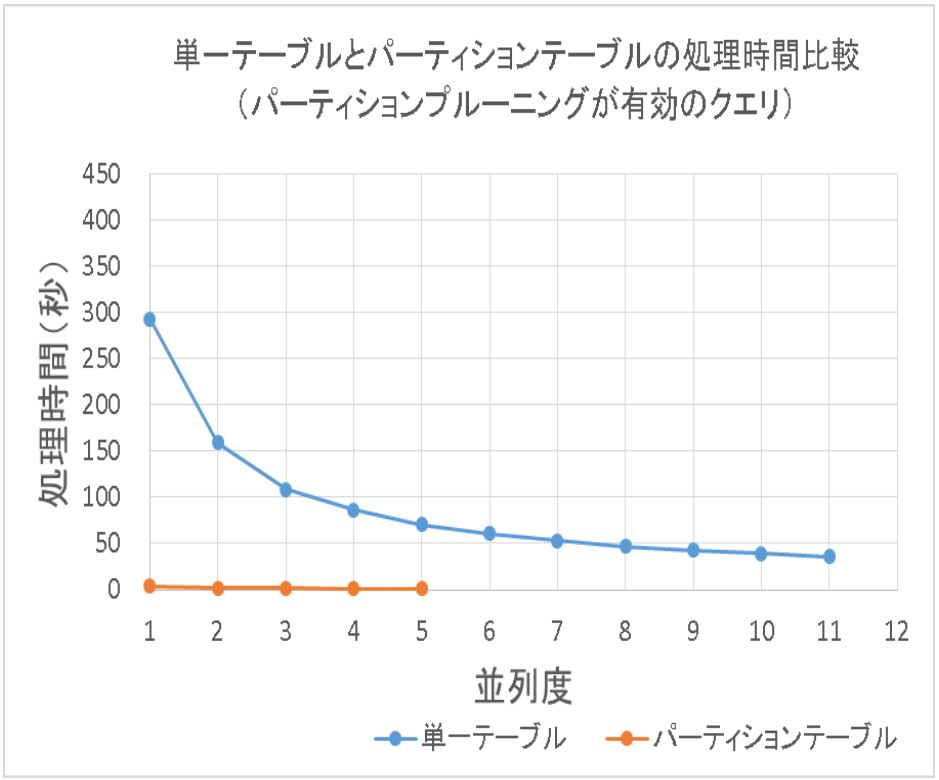
ワーカ数40まで性能向上、処理時間は約1/25に短縮
（ディスクI/Oのボトルネックがない場合）



検証B測定結果

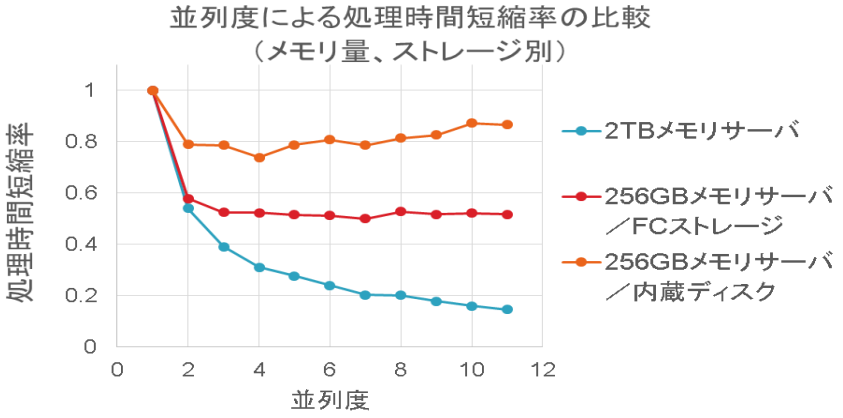
パラレルクエリによりパーティションの弱点を補完

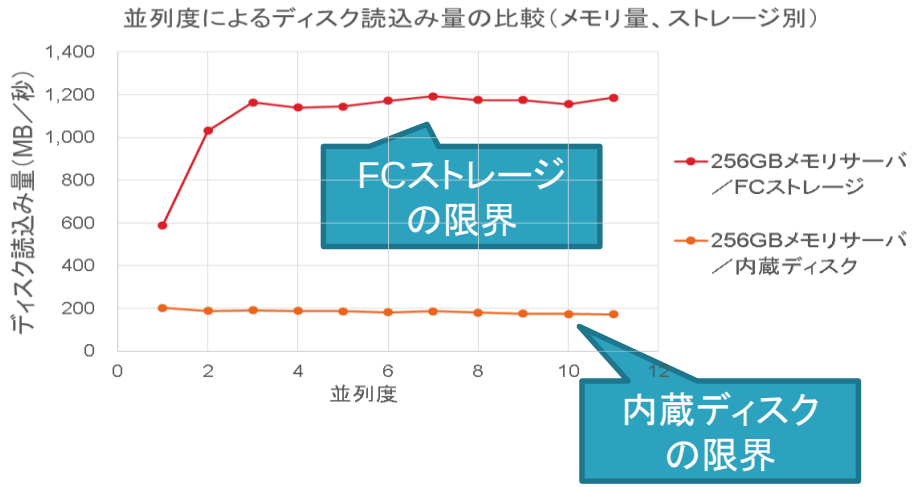




パラレルクエリ検証まとめ

- パラレルクエリにより大幅な処理時間短縮が期待できる
  - 40ワーカで1/25
  - I/Oボトルネックにならない構成が理想的





- PostgreSQL 10でのパラレルクエリの進化にも期待
  - インデックス・スキャン
  - マージ結合

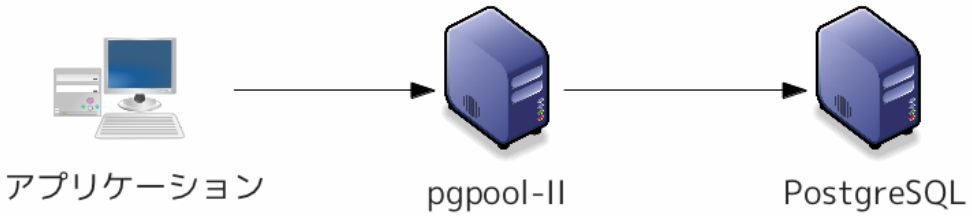
活動報告3

Pgpool-II検証

検証目的

- Pgpool-II で利用可能なレプリケーション方式の性能比較を行う

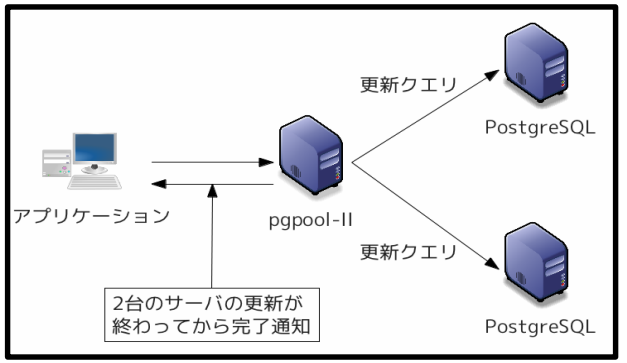
- Pgpool-II とは
  - クライアントとPostgreSQLの間で動作するツール

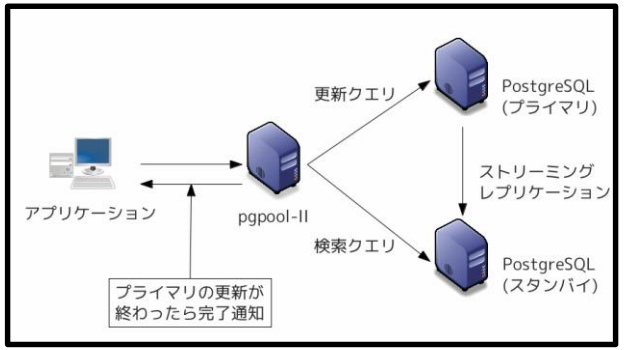


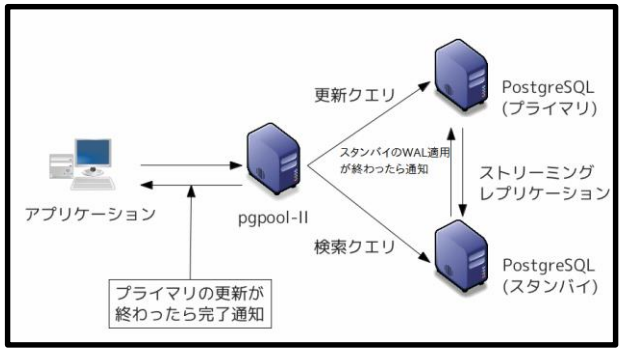
  - クラスタリング、コネクションプーリング、ロードバランシング等の機能を持つ

検証目的

- レプリケーション方式
  - ネイティブレプリケーション
    - Pgpool-II 独自のレプリケーション機能
    - 完全同期
  - ストリーミングレプリケーション（非同期）
    - PostgreSQL 本体のレプリケーション機能
    - 非同期
  - ストリーミングレプリケーション（同期）
    - PostgreSQL 本体のレプリケーション機能
    - 本検証では9.6から登場した完全同期（remote_apply）設定を用いる

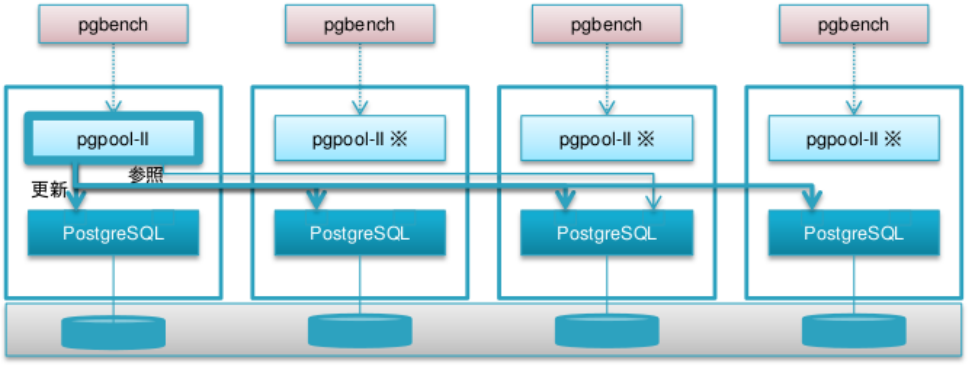






検証方法

- 各サーバと同居した Pgpool-II に対して各クライアントが同時に pgbench を実行してTPSを計測し、合計値と平均値を求める
- 検証パラメータ
  - レプリケーション方式 3種類
  - サーバ台数 (=クライアント台数) 1～4台
  - pgbench シナリオ 参照系と更新系の2種類



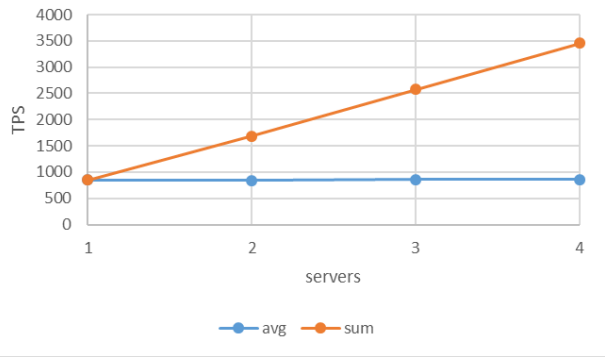
※ 2 台目以降の Pgpool-II・PostgreSQL間の矢印は省略

検証結果

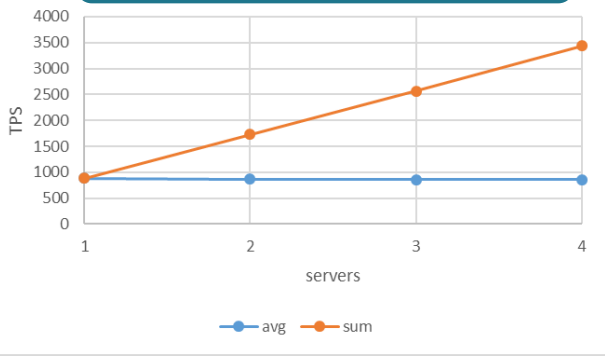
■ 参照系

いずれのレプリケーション方式においても同様にスケールアウトする

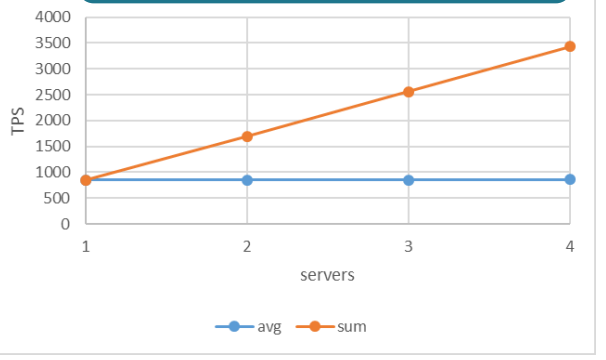
ネイティブ



非同期



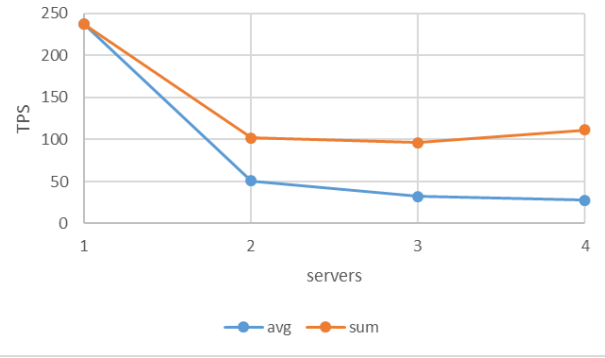
同期



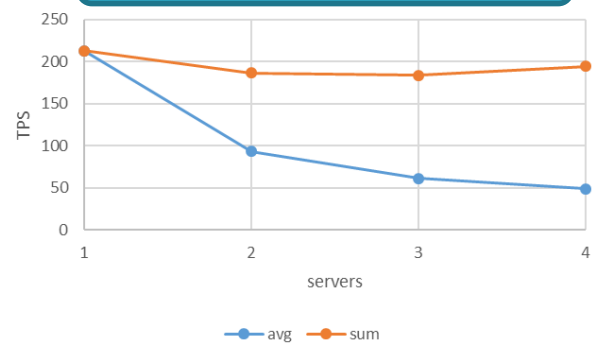
■ 更新系

ネイティブ < 同期 < 非同期

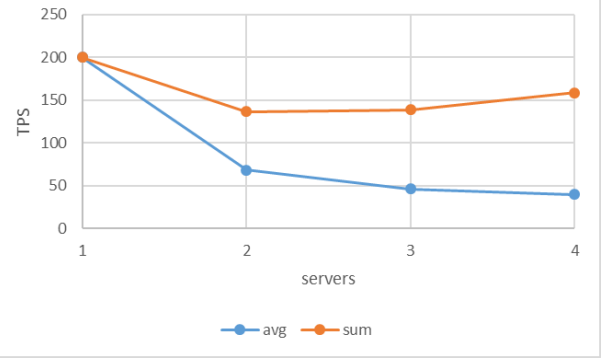
ネイティブ



非同期



同期



まとめ

- 参照性能はいずれのレプリケーション方式においても同様にサーバ数に比例してスケールアウトする
- 各レプリケーションの特性は以下の通り

	長所	短所
ネイティブレプリケーション	- 各サーバのバージョンが異なっても使用可能	- 使用できるSQLに一部制限あり
ストリーミングレプリケーション(非同期)	- サーバが増えても更新性能が劣化しない	- 非同期のためレプリケーション遅延が存在する
ストリーミングレプリケーション(同期)	- サーバが増えたときに完全同期の方式でネイティブレプリケーションに更新時スループットで勝る	- 9.6 以降でしか完全同期(remote_apply)のストリーミングレプリケーションは使用不可

活動報告4

JSON/JSONB検証

JSON/JSONB検証概要

- 検証の目的
  1. IoTデータのような非定型・大量データの格納/検索にJSONやJSONBが利用可能か
  2. どういったテーブル定義/インデックス定義が適切か
- 測定条件・環境
  - マシンは定点観測(スケールアップ検証[参照系])と同じ
  - 約9.5億件のセンサデータが格納されるモデルを想定
    - 100機器、10秒に1回センセデータを送信、3年分
  - 格納性能の測定
    - ロード時間/インデックス作成時間/テーブル・インデックスサイズ
    - インデックスとして、B-treeとBRINを使用。
  - 検索性能の測定
    - 3年分データから最近の一ヶ月分のデータの範囲を検索する時間

検証モデル

検証モデル一覧

モデル名	モデル概要
RDBモデル	センサデータ内の各要素を通常のPostgreSQLの通常のデータ型として表現したモデル
JSONモデル	センサデータ全体をPostgreSQLのJSONデータとして表現したモデル
JSONBモデル	センサデータ全体をPostgreSQLのJSONBデータとして表現したモデル
ハイブリッドJSONモデル	必ず検索キーとして使用する要素をPostgreSQLの通常のデータ型、残りの要素はPostgreSQLのJSONデータとして表現したモデル
ハイブリッドJSONBモデル	必ず検索キーとして使用する要素をPostgreSQLの通常のデータ型、残りの要素はPostgreSQLのJSONデータとして表現したモデル

検証モデル

- 検証モデルのレコードのイメージを示す

RDBモデル

deviceID	ts	Latitude	Longitude	Temparature	・・・	AtmospheiricPressure

JSON/JSONBモデル

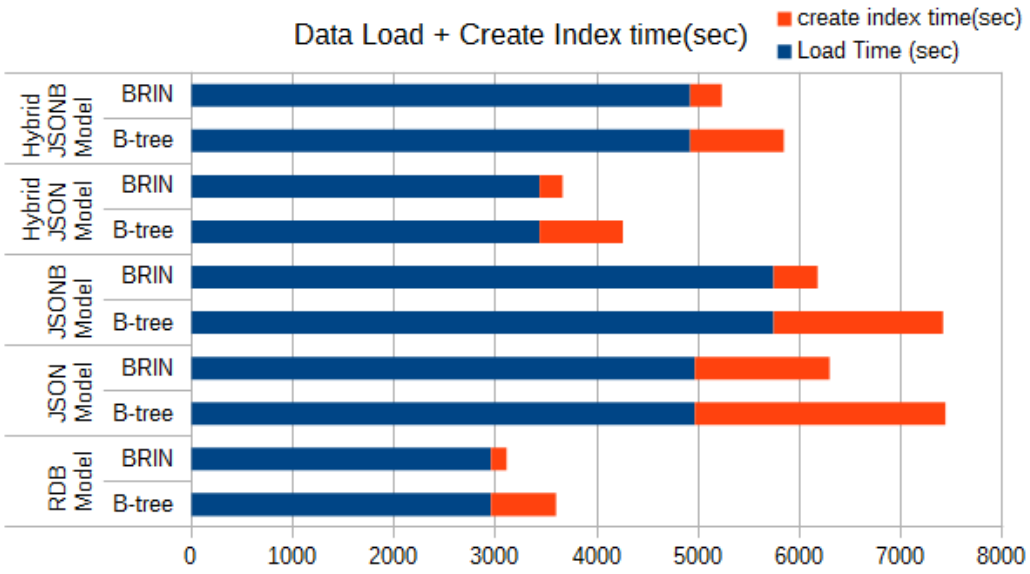
info（JSON/JSONB type)

ハイブリッドJSON/JSONBモデル

deviceID	ts	info（JSON/JSONB type)

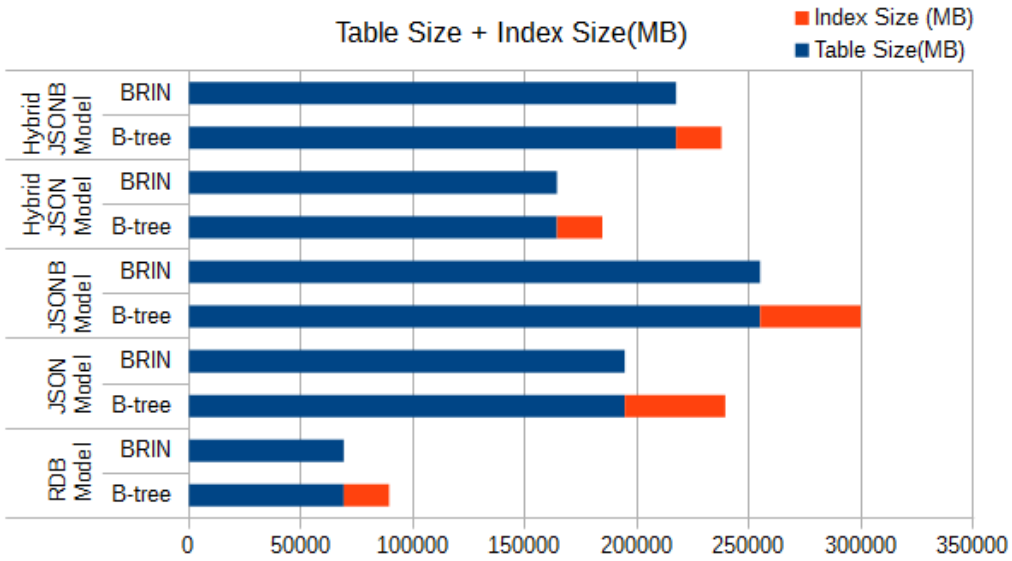
検索キー

格納性能



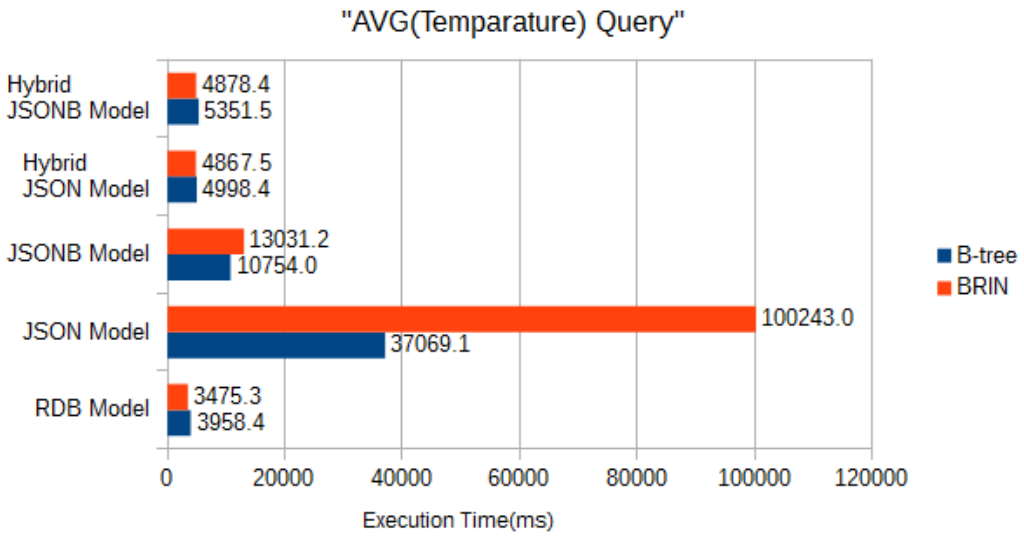
- 格納性能全般
  - （高速）RDBモデル＞ハイブリッドモデル ＞JSON/JSONBモデル（低速）
- JSONとJSONBの比較
  - ロードはJSONが、インデックス作成はJSONBが高速
- インデックス種別
  - B-treeよりBRINのほうが作成は高速。インデックスサイズの差も大きい

テーブルサイズ/インデックスサイズ



- テーブルサイズ
  - （小）RDBモデル＜ハイブリッドモデル ＜JSON/JSONBモデル（大）
- JSONとJSONBの比較
  - テーブルサイズはJSONよりJSONBのほうが大きくなる
- インデックス種別
  - B-treeと比較するとBRINは非常にインデックスサイズが小さい

検索性能



- 検索性能全般
  - （高速）RDBモデル≒ハイブリッドモデル ＞JSON/JSONBモデル（低速）
- JSONとJSONBの比較
  - 検索はJSONBのほうがJSONより優位、ハイブリッドモデルでは差はない
- インデックス種別
  - 今回のデータ量/検索パターンでは、B-treeもBRINも同等（JSON除く）

考察・まとめ

- 検索キーとして使う要素が確定している場合、ハイブリッドモデルのように、JSON/JSONB列とは分離した通常の列に持たせることで、性能の低下を抑えつつ、スキーマレスな情報の管理も同時に可能
- 非常に大量なレコード数を蓄積し、かつタイムスタンプ情報のような論理的な値と物理的な格納順序に強い相関がある場合、BRINインデックスはインデックス作成時間とインデックスサイズで有利となる

活動報告5

全文検索検証

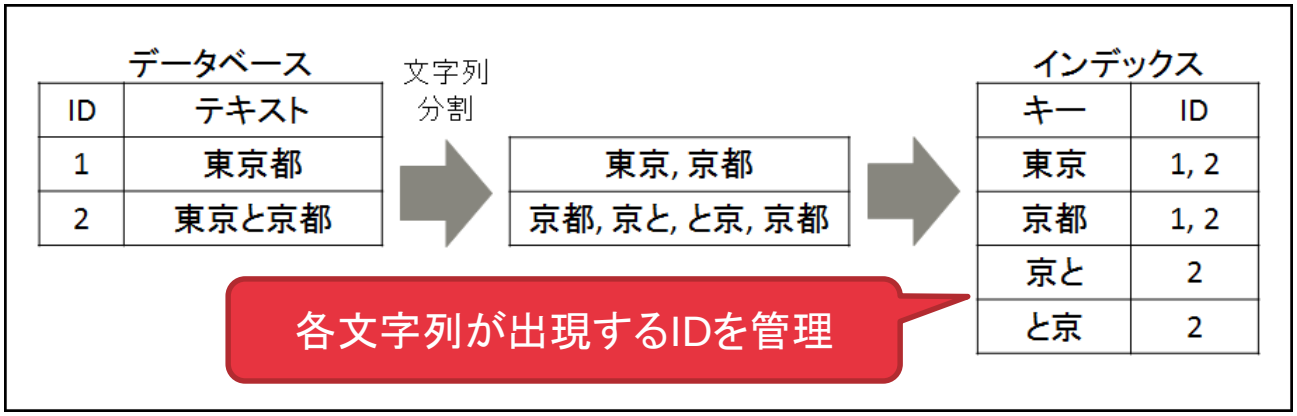
背景・概要

- PostgreSQLの適用領域の拡大、取り扱うデータ量の増加にともない、全文検索の重要性がますます高まっている
- 本検証では、特に国内においてニーズの高い「日本語検索」に対応した代表的な検索手法(インデックス)である以下の3つを比較
  1. pg_trgm (ピージー・トリグラム)
  2. pg_bigm (ピージー・バイグラム)
  3. PGroonga (ピージールンガ)
- Wikipediaの文章データ(41GB、250万レコード)をテーブルに格納してインデックスを作成し、キーワード検索にかかる時間を比較する
  - Wikipediaのダンプデータを利用し、idと本文からなるテーブルを作成する

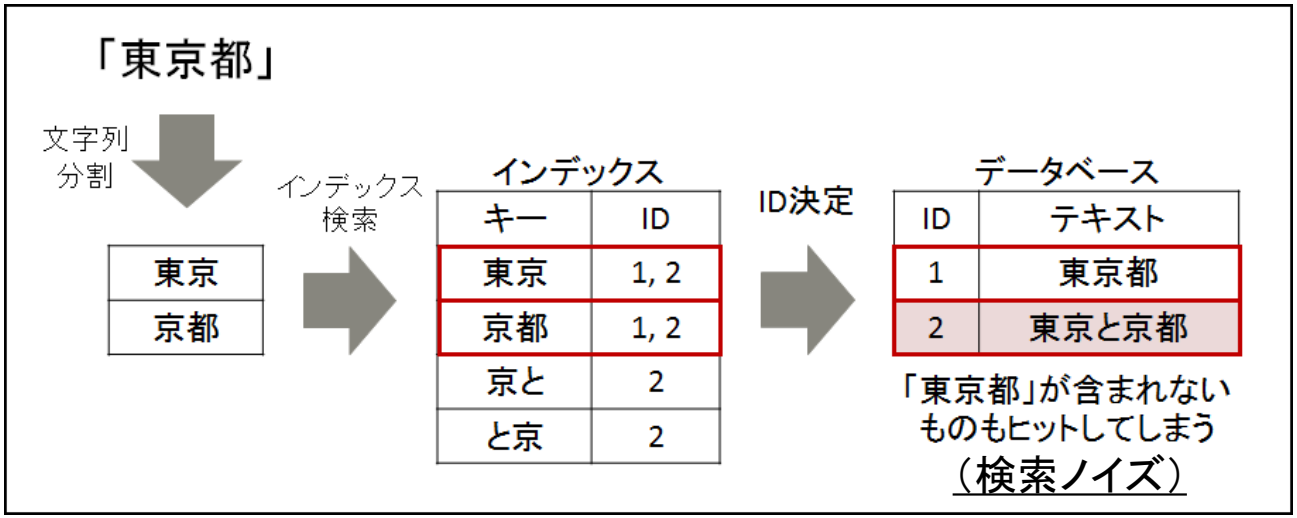
id	Text(text型)
1	PostgreSQL(ぽすとぐれすきゅーえる:発音例)はオープンソースのオブジェクト関係データベース管理システムである。…
2	MySQL(まい・えすきゅーえる)は、オープンソースで公開されている、関係データベース管理システム(RDBMS)の一つである。…
…	…

PostgreSQLにおける全文検索の仕組み

- テキストを数文字単位で区切り、インデックスを作成する（N-gram方式）
- インデックス作成時



- インデックスによる検索時



インデックスのみの結果をPostgreSQLが精査し、
検索ノイズを除外する（Recheck処理）

検証方法

- 文章データにCREATE INDEX文でインデックスを作成したテーブルに対し、標準SQLのLIKE文で検索を行い、かかった時間を測定
- マシンは以下の環境（検証環境2）を使用
  - サーバ／クライアントは同じマシン上で実行

機種	富士通PCワークステーション CELSIUS R670-2
CPU	インテル Xeonプロセッサ X5650@2.66GHz 6コアx2 合計12コア
メモリ	48GB
内蔵ストレージ	HDD 500GB SATA x 2

- 作成時間と作成されたデータ量の比較

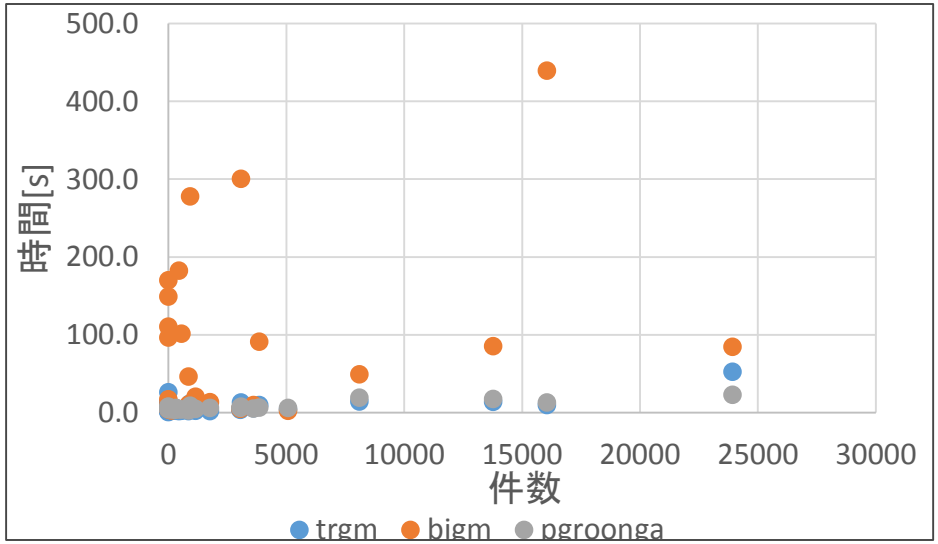
インデックス	インデックス作成時間	データ増分
pg_trgm	26時間46分	20GB
pg_bigm	8時間16分	10GB
PGroonga	6時間46分	131GB

  - インデックスの形式により、作成時間とデータ増分が大きく異なっている
  - 特に、PGroongaでは元の文章データ（41GB）よりも多くの容量を使用している

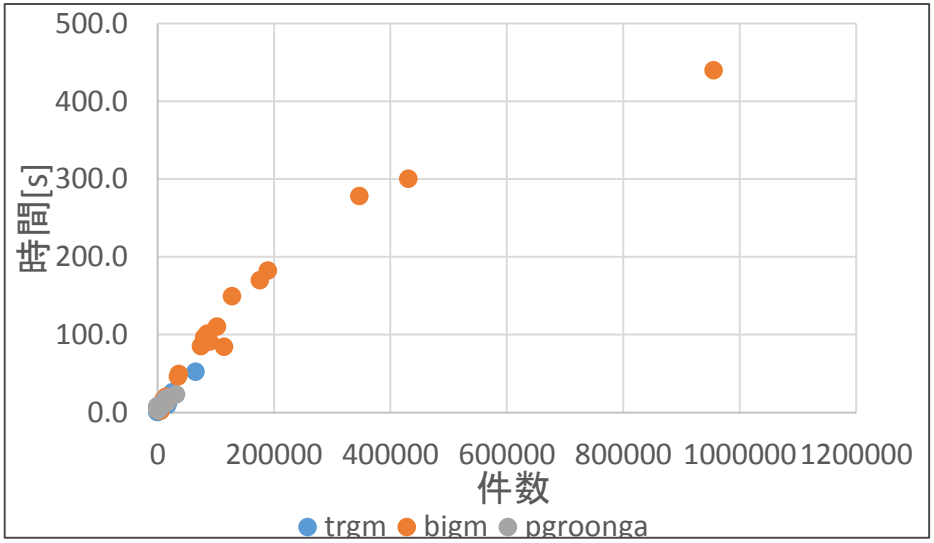
測定結果：英語キーワード

- PGroonga, pg_trgmに比べ、pg_bigmで時間がかかっている
- 検索時間と検索結果の件数の相関はほぼ見られなかった
  - 原因は、インデックスによる検索後にRecheck処理による時間がかかること
  - pg_bigmでは、文章を2文字単位で区切ってインデックスを作成するため、他のインデックスに比べると検索ノイズが多くなってしまい、結果的にRecheck処理に時間がかかってしまう
- Recheck前の件数（インデックスのみの件数）と時間には相関が見られ、検索時間に与える影響が大きいことが見て取れる

件数と時間の相関

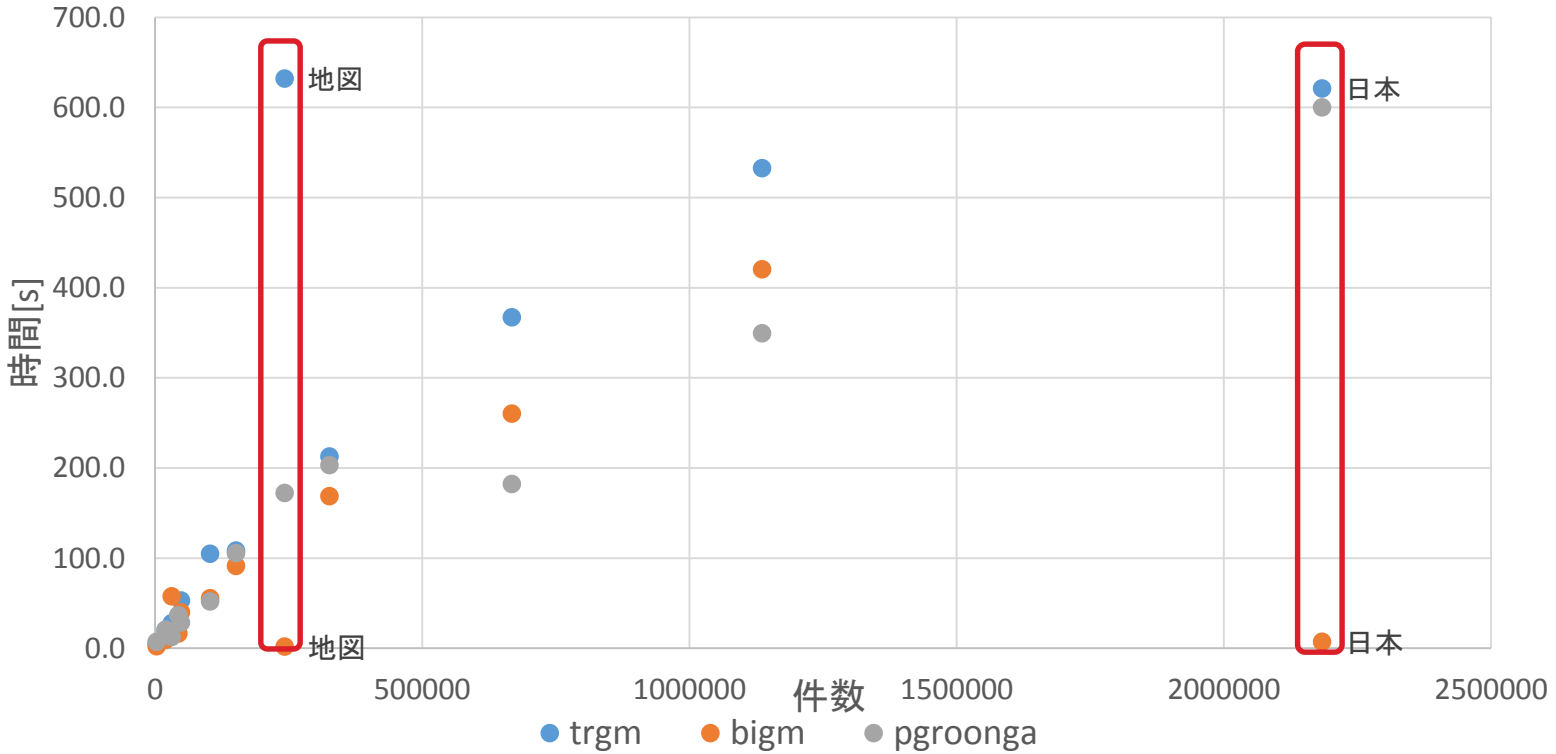


Recheck前件数と時間の相関



測定結果：日本語キーワード

- 検索時間と件数の相関を見ると、検索時間が件数にほぼ比例
- 単語が2文字の場合に、インデックスの特性が見られる
  - pg_bigmは、文章を2文字で区切ってインデックス作成を行うため、非常に高速
  - pg_trgmは、文章を3文字で区切ってインデックス作成を行うため、シーケンシャルな検索が実行されており時間がかかっている



結果まとめ

- 代表的な全文検索手法(インデックス)3種の比較を行った
- 本検証での3種の強みと弱み

インデックス	強み	弱み
pg_trgm	英語キーワード	2文字のキーワード 日本語キーワード
pg_bigm	2文字のキーワード	英語キーワード
PGroonga	ほとんどの場合に 高速である	インデックスサイズが大きい

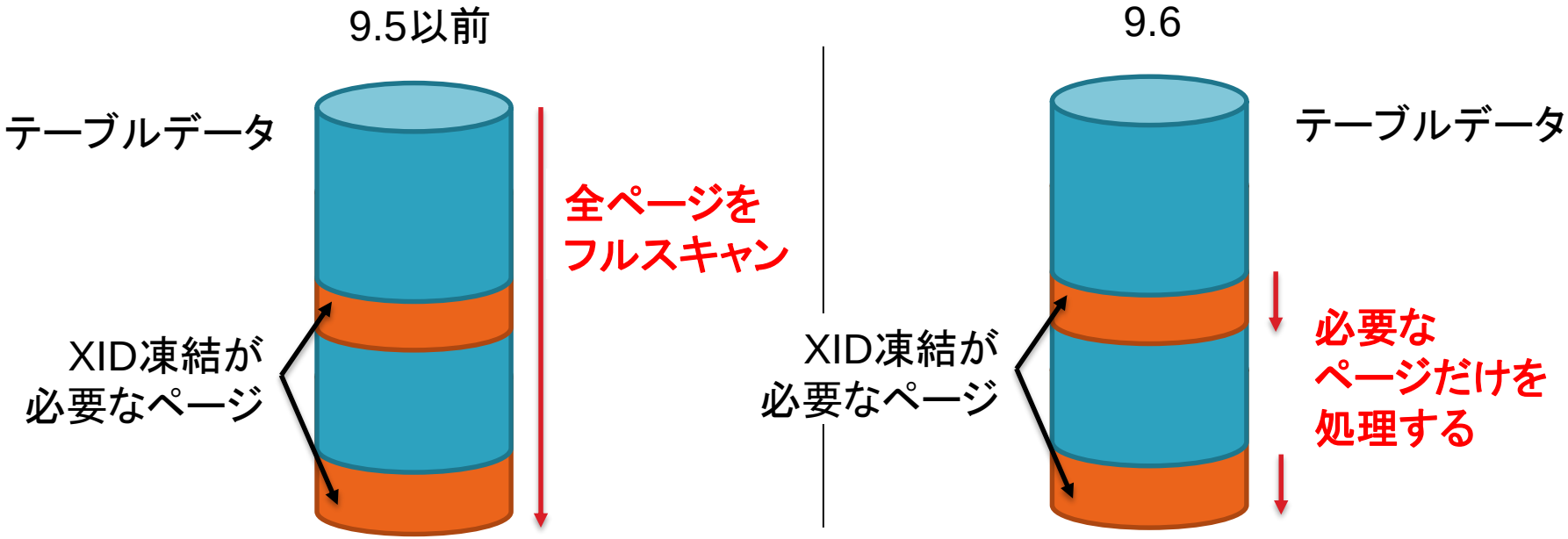
- インデックスを利用した検索では、データ読み取り時間に加え、インデックス読み取り時間の考慮も必要
  - PGroongaはインデックス自体の容量が大きく、今回の検証環境ではメモリに乗り切らないサイズであったため、インデックスの効率よりもI/Oがボトルネックとなってしまったのではないかと考えられる

活動報告6

VACUUM改善

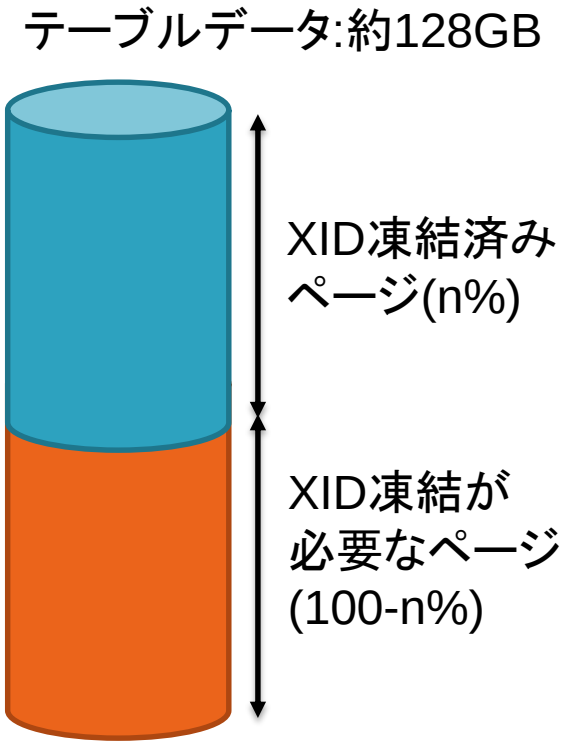
PostgreSQL 9.6におけるVACUUM FREEZE

- 9.6では、ページ内容がすべてトランザクションID (XID) が凍結済みというフラグをVisibilityMapに追加
- フラグのあるページではVACUUM FREEZEをスキップ
- 古いまま更新されないデータの多いテーブルに有効



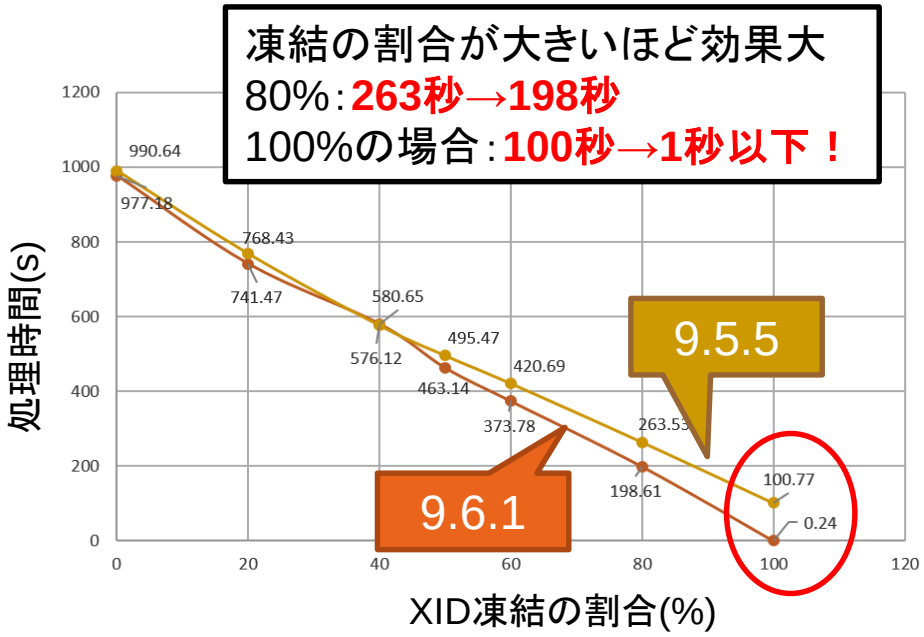
検証目的・検証方法

- XID凍結済みのページ数の割合を数パターン用意し、VACUUM FREEZEを実施した際の性能を測定
- 9.5と9.6を比較し、性能の改善度合いを確認
- 測定条件
  - XID凍結済みの割合を数パターンにわける
    - 128GB×n%相当の行数INSERT
    - VACUUM FREEZE
    - 128GB×(100-n)%相当の行数INSERT
  - 上記のテーブルに対して、VACUUM FREEZEの処理時間を測定、XID凍結の割合との関係を確認する

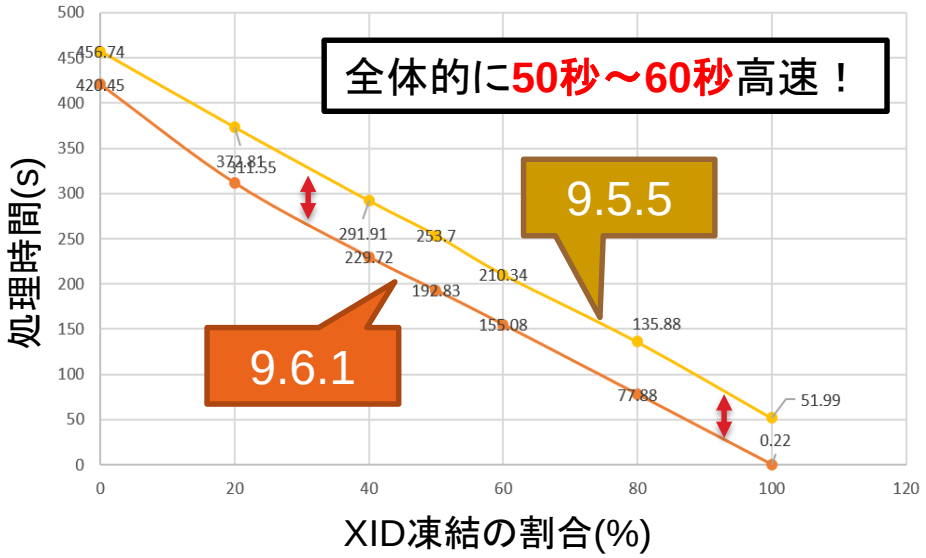


検証結果

- 仕様通り9.6ではXID未凍結のページだけスキャンしていることを確認。100%凍結済みなら1秒以下で完了
- オンバッファでない場合は最大で100秒、オンバッファでは全体的に50秒前後高速化されることを確認



shared_bufferにオンバッファなし



shared_bufferに全てオンバッファ

考察・まとめ

- 大規模なDBで計画的にVACUUM FREEZEを行うシステムでのメンテナンス時間短縮に大きく期待できる
  - 9.5で取り入れられたvacuumdbユーティリティの複数JOB実行と組み合わせて、従来より大幅に時間短縮も可能
- shared_bufferに乗り切っていないテーブルの場合、XID凍結の割合が多いほどディスクI/Oが軽減するため効果が非常に大きくなる
- オンバッファではXID凍結済みページの多寡に限らず9.6の方が高速
  - sarの%utilを見てみると9.5.5と9.6.1とでは、大きく傾向が異なっていた。チェックポイントの改善といった他の改善による影響が考えられる

活動報告7

2016年度活動をふりかえって

2016年度活動をふりかえって

- 検証を通して、PostgreSQLが時代に即した
  高機能・高性能化を遂げていることを実感しました
  - メニーコアCPUへの対応強化
  - 大量データへの適用性向上
  - 多様化するデータ型への対応

- 9.6の機能・性能について中身の濃い議論ができました
  - 新機能についての情報交換
  - 検証テーマごとに測定パターンから議論
  - プロファイリングを含む結果の分析

2016年度活動をふりかえって

- さまざまなメディアで紹介されたとおり、
  競合することもある各企業のメンバーが、
  共通の目的のもと活動し、交流を深めるということは
  とても意義のあることです。

- WG検討会では報告書に載せきれない貴重な情報を
  数多く聞くことができました。来年度はぜひ一緒に
  活動しましょう！

検証環境1 （提供：日本ヒューレット・パッカード株式会社）

HPEソリューションセンター

HPE ProLiant DL360 Gen9 (3台)

CPU: Xeon E5-2620v4 (8Core) × 2 [16Core]
Memory: 128GB / HDD: 1.2TB 10k x 8

HPE ProLiant DL380 Gen9 (2台)

CPU: Xeon E5-2690v4 (14Core) × 2 [28Core]
Memory: 256GB / HDD: 1.2TB 10k x 16

HPE ProLiant DL580 Gen9

CPU: Xeon E7-8890v4 (24Core) × 4 [96Core]
Memory: 2TB / HDD: 1.2TB 10k x 10

10GbE Network Switch

HPE 5700-32XGT-8XG-2QSFP

16Gb FC Switch

16Gb FC Switch

VPN

HPE SN3000B 16Gb 24/24

RAID10 for DL580 [定点観測] (3TB) x 4
RAID10 for DL380#1 [パラレルクエリ] (3TB) x 4
RAID10 for DL380#2 [JSONB] (3TB) x 4
RAID10 for DL360 [パラレルクエリ] (3TB) x 4

HPE 3PAR StoreServ 8440

HDD: 1.8TB x 72

インターネット経由
リモートアクセス

今回使用した検証環境について

- 今年度の性能検証は日本ヒューレット・パッカード株式会社様から検証環境をご提供いただきました。
- PostgreSQLエンタープライズ・コンソーシアムとして御礼を申し上げます。

PGECons

PostgreSQL Enterprise Consortium