
PGECons
PostgreSQL Enterprise Consortium

# 大規模DBへの適用性が向上した PostgreSQL 9.5 の性能検証

## 2015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 (性能WG)

# アジェンダ

- WG1(性能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]
  2. WAL改善/スケールアップ検証[更新系]
  3. Parallel Vacuum 検証
  4. BRIN Index 検証
  5. OS 比較検証
  6. 2015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
  - 2015年度は上記に加えて以下の活動テーマを選定
    - 大規模DB向けのPostgreSQL 9.5の新機能
    - Red Hat Enterprise Linuxの新旧バージョンの比較
- 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の活動テーマ（赤字は2015年度活動テーマ）

## 大規模DBへの適用性が向上したPostgreSQL 9.5の性能検証

### ■ 大規模DBの特徴

- 使用ユーザー数が多い
- データ量が多い
- 高性能が求められる

### ■ 稼動環境の多様化

- 柔軟なDB稼動環境

### ■ 対応方法

- スケールアップ
- スケールアウト
- スキーマ・クエリの最適化
- ストレージ高速化
- 多様な環境適用

### ■ 主な検証内容

- スケールアップ
  - ●多コアCPU検証（※）
- スケールアウト
  - ●コネクションプール（pgpool-II）
  - ●クラスタDB（Postgres-XC, Postgres-XL）
- スキーマ・クエリの最適化
  - ●BRIN index 適用性
  - ●Parallel Vacuum 適用性
  - ●パーティショニング
- ストレージ高速化
  - ●SSD有効活用手法
- 多様な環境適用
  - ●RHEL 6/7 比較検証
  - ●仮想化（KVM）検証
  - ●Docker検証

※「定点観測」と呼び毎年実施

# 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**

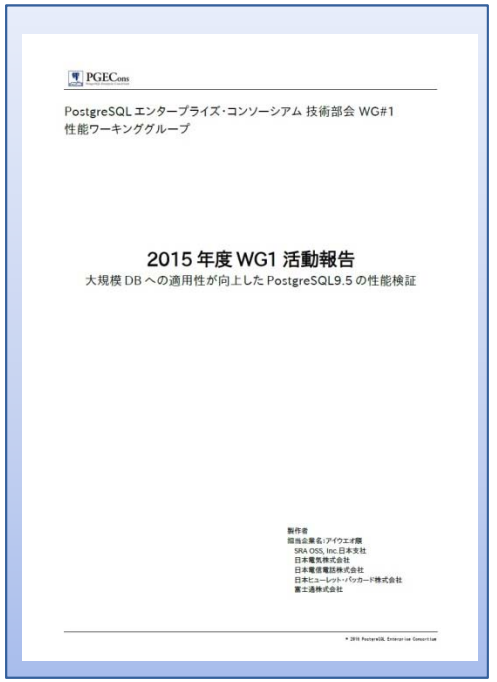
  パーティションテーブルへの投入・検索・メンテナンス性能検証

- **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

- 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


PGECons

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#1
性能ワーキンググループ

2015 年度 WG1 活動報告
大規模 DB への適用性が向上した PostgreSQL9.5 の性能検証

# WG1参加メンバ

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

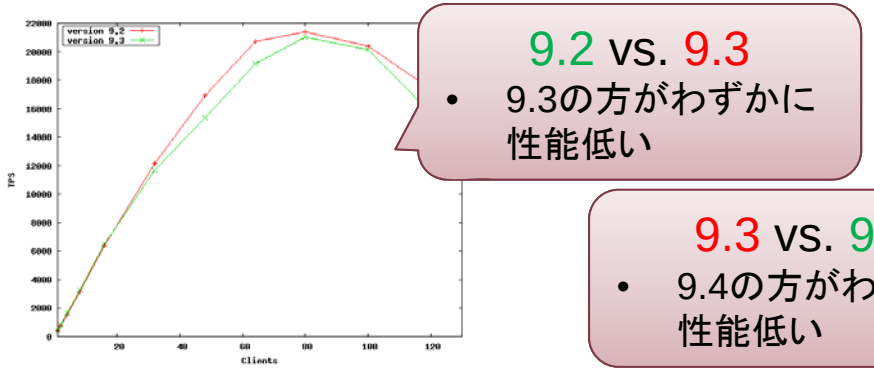
(企業名50音順・敬称略)

活動報告1

# 定点観測/
# スケールアップ検証[参照系]

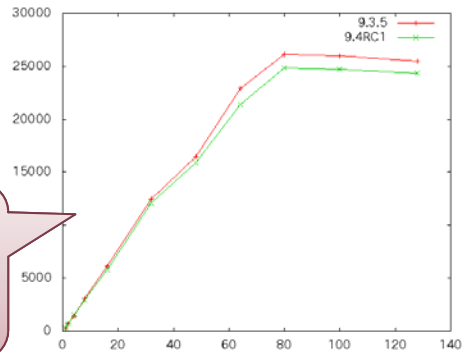
# スケールアップ検証 [参照系] (定点観測)

- 目的
  - 多コアCPU上での PostgreSQL 9.5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 9.4 との比較を実施
- 過去の結果
  - 過去検証では、9.2当時からスケール性は確認できたものの9.3, 9.4においてバージョンアップによる性能向上は見られなかった



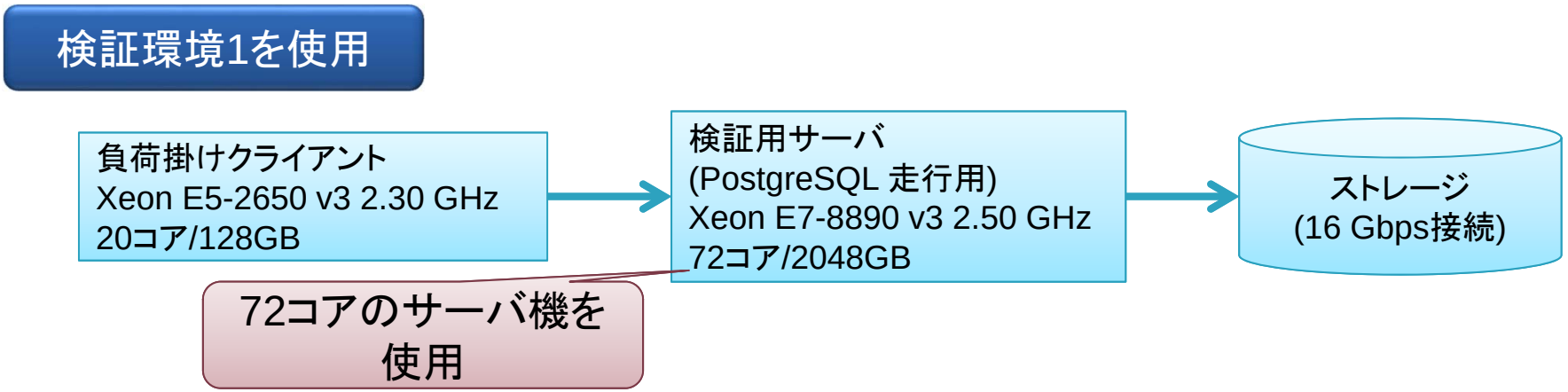

9.2 vs. 9.3
- 9.3の方がわずかに性能低い

9.3 vs. 9.4
- 9.4の方がわずかに性能低い



# 検証方法

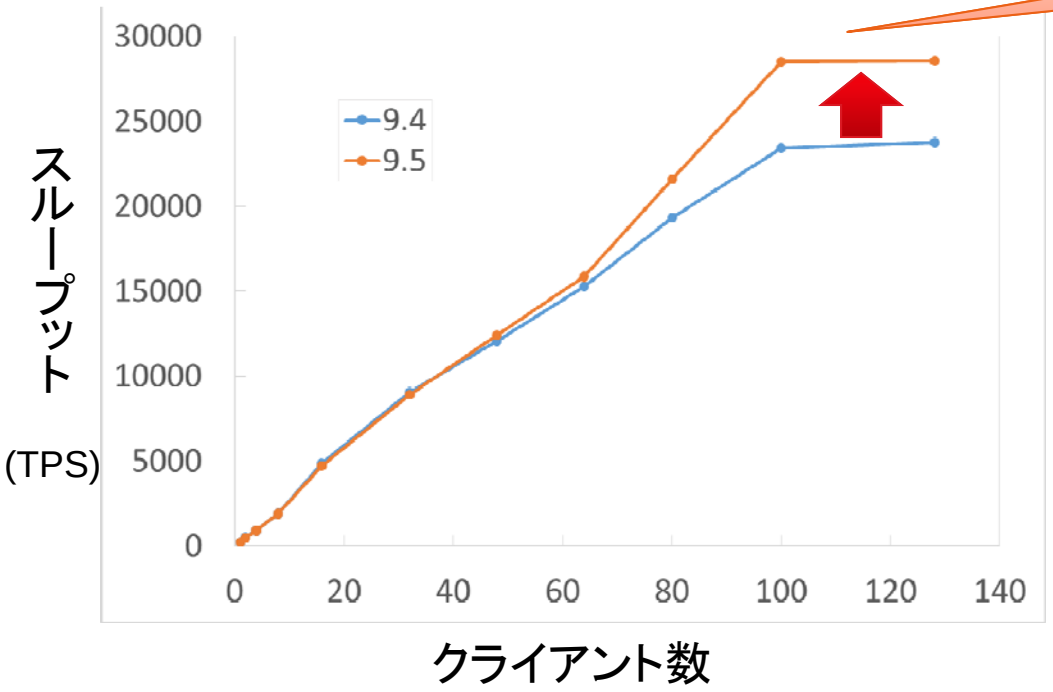
- CPUに十分な負荷が掛かる条件でpgbenchを実行
  - オンメモリでの走行
    - スケールファクタ1,000（DBサイズは15GB）
    - 事前に pg_prewarm を実行した
  - SQL1文の負荷を大きく
    - 1回の実行で1万行取得するカスタムスクリプトを使用
  - 多数のクライアントから同時にリクエスト
    - クライアント数を変化（1～128）させスループットを測定

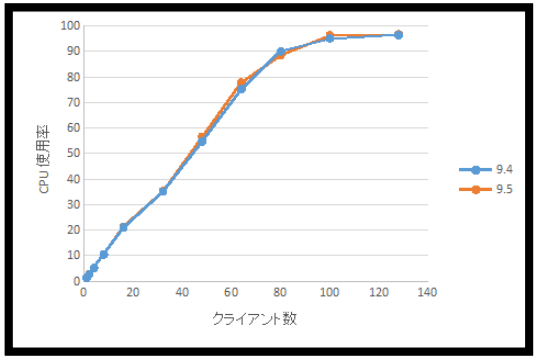


# 測定結果

- 9.5はクライアント数100で最高性能28,500TPS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
  - 9.4も同条件で最高性能である23,400TPSに達した
    - 両者クライアント数100でピーク性能に達する
  - 9.5は対9.4で22%の性能向上

9.5 vs. 9.4にて
22%の性能向上を確認



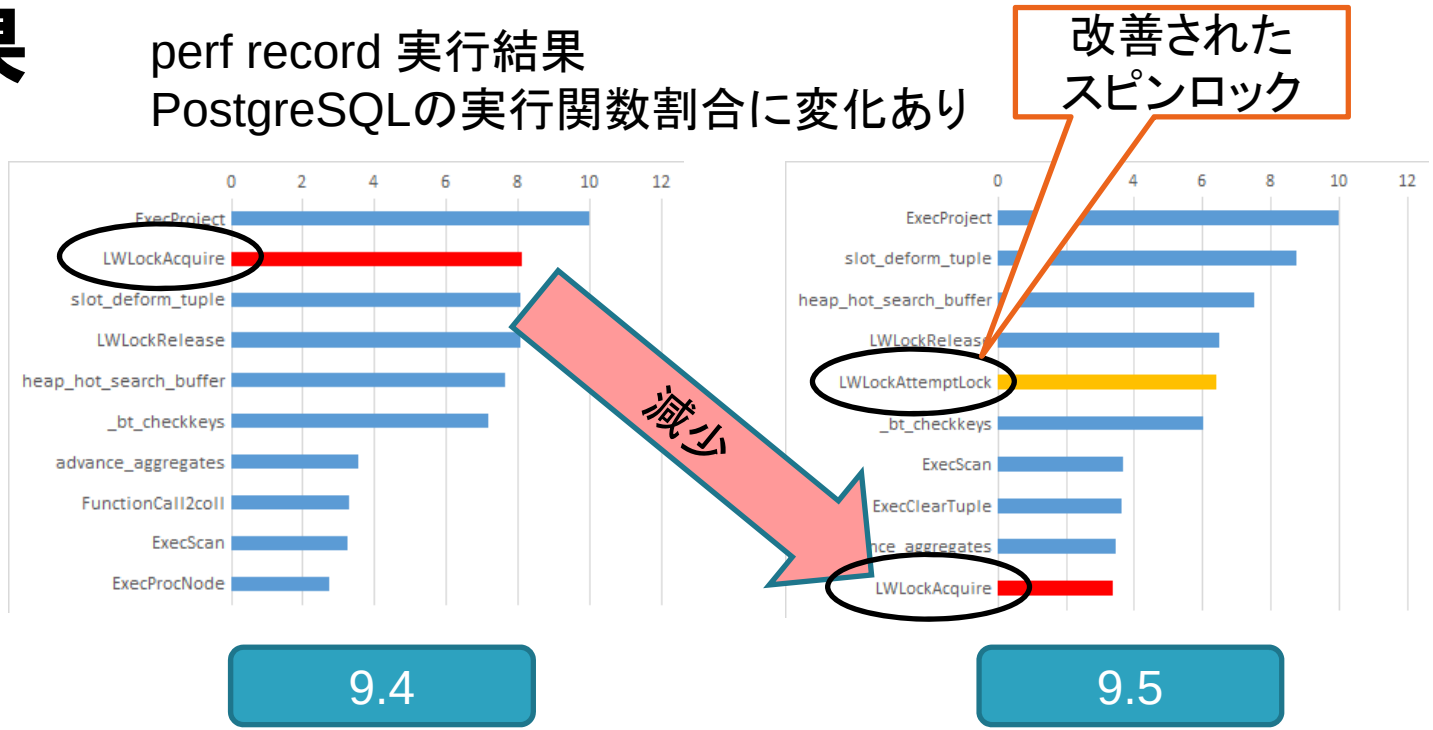

CPU利用率は9.5, 9.4ともほぼ100%に達しており、有効に使用している

# 測定結果の考察～改善した理由～

- スケーラビリティが向上した原因をプロファイラで調査
- 9.5のロック機構改善が寄与していることを確認
  - LWLockAcquire（ロック取得関数）の実行割合が下がった
  - 分岐の頻度が下がり、IPCが向上した

## 結果

perf record 実行結果
PostgreSQLの実行関数割合に変化あり



# 結果まとめ

- 9.4と比較し、9.5はより優れたスケーラビリティ性能が確認できた
  - 約22％の向上を確認
- 9.5の修正による動作の変更を実際に確認できた
  - LWLockAquire 内スピンロックのボトルネックが解消された
  - 本修正が今回のスケーラビリティ性能向上に寄与していると思われる

活動報告2

# 定点観測/
# スケールアップ検証［更新系］

# 定点観測(スケールアップ検証[更新系])概要

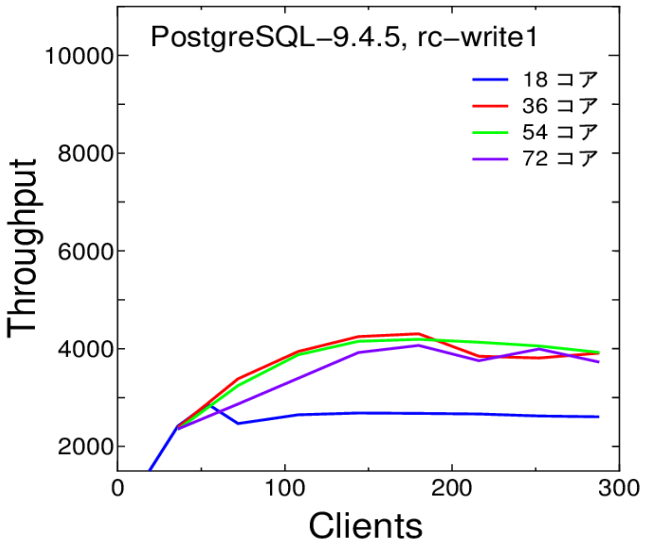
- PostgreSQL 9.4 (2014年度) から更新系のスケールアップ検証を実施
  - 今年度は定点観測として継続実施
  - 参照系のスケーラビリティ検証はPGECons発足当初 (2012年度) から定点観測を継続中
  - 測定したバージョン: 9.4.5 と 9.5beta2 を比較
- PostgreSQL 9.5で実装された更新系性能に関係する改善点
  - 共有ロックのスケーラビリティ改善
  - 共有バッファ・ロックのパーティション数増加(16->128)
- 改善の効果を3つの面から確認することを目的とする
  1. PostgreSQL 9.5beta2 のPostgreSQL 9.4.5 からの更新性能改善
  2. PostgreSQLの更新系のCPUスケーラビリティの達成状況
  3. PostgreSQL 9.5の性能を更に引き出す手法の検討

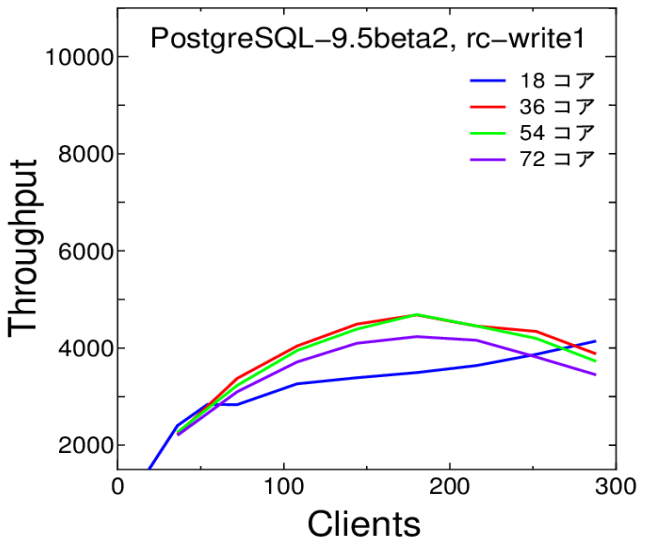
# 測定条件・環境

- マシンは検証環境1（付録参照）を使用
- サーバ側のコア数（18, 36, 54, 72コア）とpgbenchの接続数を変更して測定
- 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
  - 各ボリュームは36台RAID10相当を4つに論理分割したもの
- 測定ツールはpgbench（独自トランザクション）を利用
  - DBサイズはSF=7,000（約100GB）とし、オンメモリで動作
  - トランザクションは独自に1つの大きいテーブル（pgbench_accounts）に対するUPDATEのみと設定し、ブロック排他の集中を避ける
  - FILLFACTOR=80で定義し、HOTの効果を期待
- 測定中にCHECKPOINTは発生させないように設定
- 3回の走行の測定値の中央値を測定結果とする

# 測定結果概要

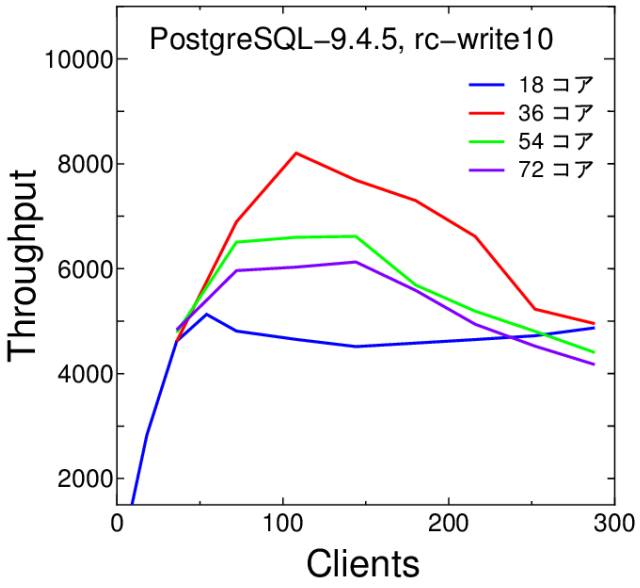
1. PostgreSQL 9.5beta2は9.4.5よりピーク性能が向上
   - 参照系とは異なり、コア数の2倍以上の接続数でピーク性能となる
     - 36コア以上では、いずれも接続数150～200辺り
2. PostgreSQL 9.4.5、9.5beta2のCPUスケーラビリティ
   - 両者、コア数36で最も高いスループットとなり、9.5beta2の方がやや高い
   - 18コアでは9.5beta2の方が高負荷時にスループット高い
   - 36コア以上の高負荷領域では、9.4.5よりも下がっているケースもあった

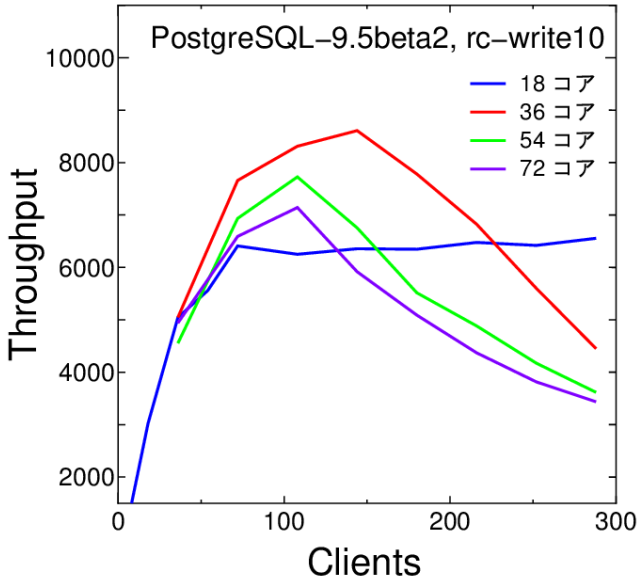


# ロック競合を減らした場合の性能状況

- 1トランザクションあたり1更新→10更新としロック競合を緩和
  - ProcLockArray*の競合の緩和を狙う
- 性能状況
  - スループットは 1トランザクションあたり1更新 の約2倍
  - 36コアが最も高いスループット、54, 72コアとの差が明瞭になった
  - 18コア：クライアント数を増やしても性能が落ちない

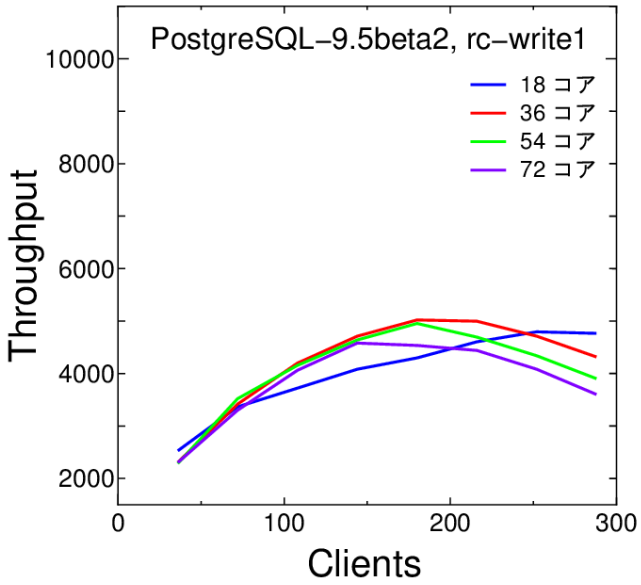


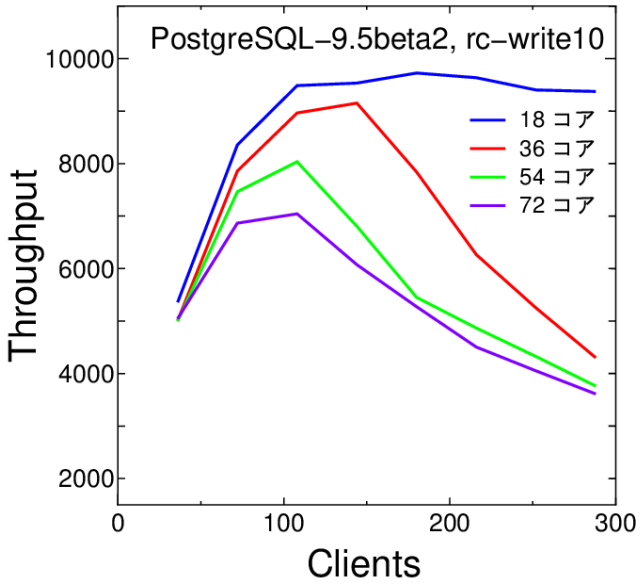

＊：昨年度の性能検証で ProcLockArray がボトルネックであろうと示唆された

# CPU負荷削減時の性能状況

- CPU負荷削減時の性能の検証
  - Huge pages *1を用いてTLBミスの発生頻度低減→CPU負荷削減
- 更新性能に対する効果
  - 10更新/トランザクションのケースで有意な性能向上が観測された
    - 特に18コアのケースで性能向上の効果が顕著




(1更新/1Trx :前々slide 右グラフとの比較)




(10更新/1Trx :前slide 右グラフとの比較)

*1: Huge pages は、transparent huge pages とは異なり、そのためのメモリ領域を予め確保しておく方法。

# 結果まとめ

1. 9.5beta2は9.4.5と比較して更新性能が改善されることが確認できた
2. PostgreSQLの更新系のCPUスケーラビリティ
   - 18/36/54/72コアの測定単位では、単純な更新のみのワークロードでは、9.4.5、9.5beta2共CPUスケーラビリティについて確認できなかった（昨年度と同様の結果）
     - 36コアが最も高いスループットとなった
   - ただし、参照系でのCPUスケーラビリティが確認できているため、採用システムのワークロード（更新・参照の割合等）に応じたCPU設計が必要であると判断

- 性能向上策（ロック競合の削減）の効果を確認
  - 複数の更新を1トランザクションで実施（ProcArrayLock競合削減）
  - Huge pages利用（TLBミス削減）、Prepare文利用（SQLのparse処理削減）
  - Isolation level変更（ProcArrayLock競合削減）
  - 各対策のトレードオフについては活動報告に記載

詳細は、2015年度WG1活動報告をご参照ください

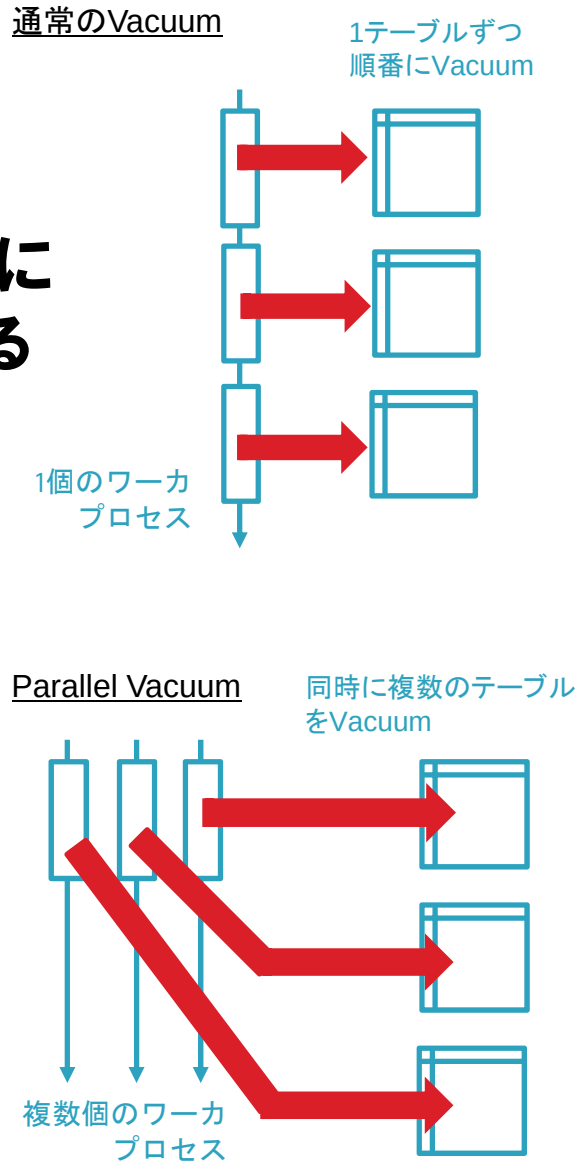
活動報告3

# PARALLEL VACUUM 検証

# 検証目的

- Parallel Vacuumとは
  - 複数のワーカプロセスで複数のテーブルに対して同時にVacuumやAnalyzeが行える
  - テーブル数が多い場合に適用できる
    - 大規模なパーティショニングされた複数のテーブルがあるようなケースで効果が期待できると思われる

- Vacuumを行うワーカ数を増やすと、どの程度が処理時間の短縮が期待できるかを確認



# 測定環境

- 物理サーバ1台とDB用外部ストレージで構成
  - 検証環境1を使用

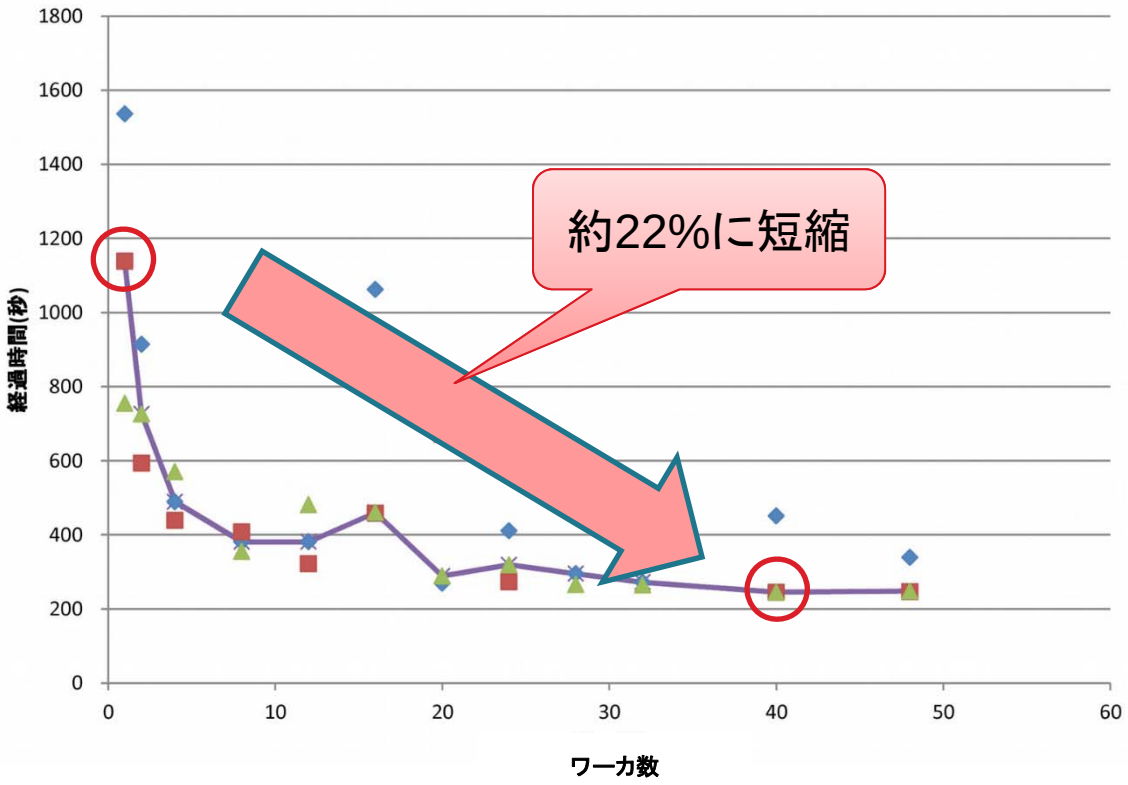
| 項目 | 仕様 |
| --- | --- |
| CPU | XeonプロセッサE5-2690v3@2.60GHz<br>12コア×2ソケット = 24コア |
| 搭載メモリ | 256GB |
| DB格納用ストレージ | Fiber Channel接続(16Gbps) SAN<br>DBとWALで格納領域を分ける<br>領域1：PostgreSQLのDB領域<br>領域2：WAL領域<br>※それぞれRAID10、実効容量約3TB |
| OS | Red Hat Enterprise Linux 7.2 |
| DBMS | PostgreSQL 9.5 beta2 |

# 検証方法・測定条件

- Parallel Vacuumが有効となりそうな比較的件数の多いテーブルを複数用意
  - pgbenchのpgbench_accountsテーブルを利用し、2000万件、計48個のテーブル
  - HOTが働くようにFILLFACTOR=80で定義
  - 一定時間ランダムな更新処理をかけて、ガベージを作成
- autovacuumは停止しておく
- ワーカ数を1～48の中で変えて、Vacuum Freezeを実行し、完了までの処理時間を測定する
  - 3回測定し、その中央値を測定結果とする

# 測定結果

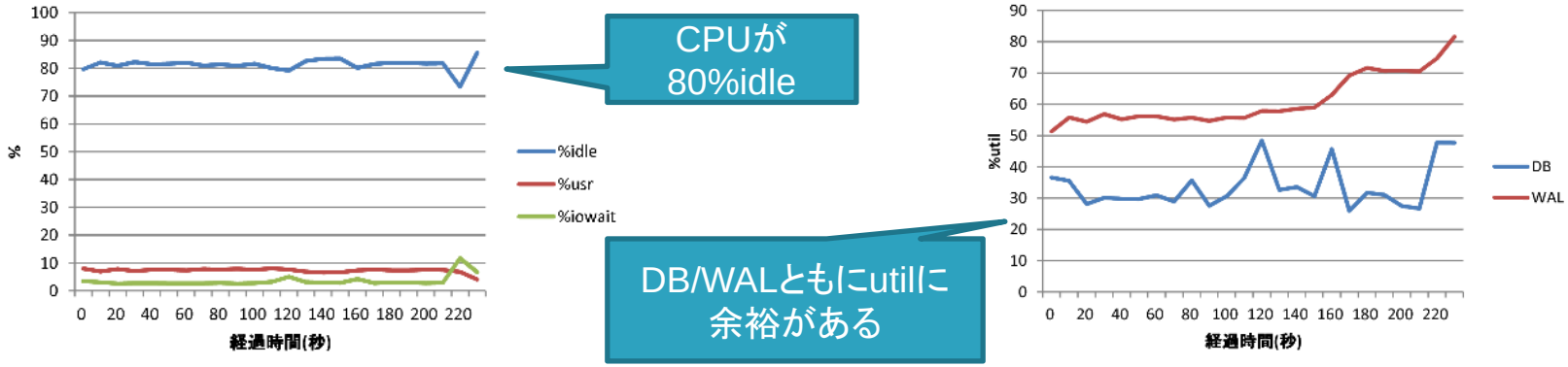
- 処理時間が最短だったのはワーカ数=40で、ワーカ数=1に対して22%（1/4.6）
- ワーカ数=20以降はほとんど変わらない




| ワーカ数 | 処理時間[秒] | ワーカ数=1に対す処理時間 |
|---|---|---|
| 1 | 1138 | 100% |
| 2 | 726 | 64% |
| 4 | 489 | 43% |
| 8 | 381 | 33% |
| 12 | 381 | 33% |
| 16 | 460 | 40% |
| 20 | 289 | 25% |
| 24 | 319 | 28% |
| 28 | 295 | 26% |
| 32 | 272 | 24% |
| 40 | 245 | 22% |
| 48 | 248 | 22% |

# 結果まとめ

- Parallel Vacuumで処理時間短縮が期待できる
  - 40ワーカの時に22%（約1/4.6）
  - ある程度性能の高いストレージが時間短縮の条件
    - I/O分散を意識していないシンプルな構成では約83%に留まる（詳細は2015年度WG1活動報告を参照）
- 参考情報：CPUやストレージの利用率が低い
  - バックグラウンドライタやWALライタがボトルネックと想定（下図、詳細は未調査）



活動報告5

# BRIN INDEX 検証

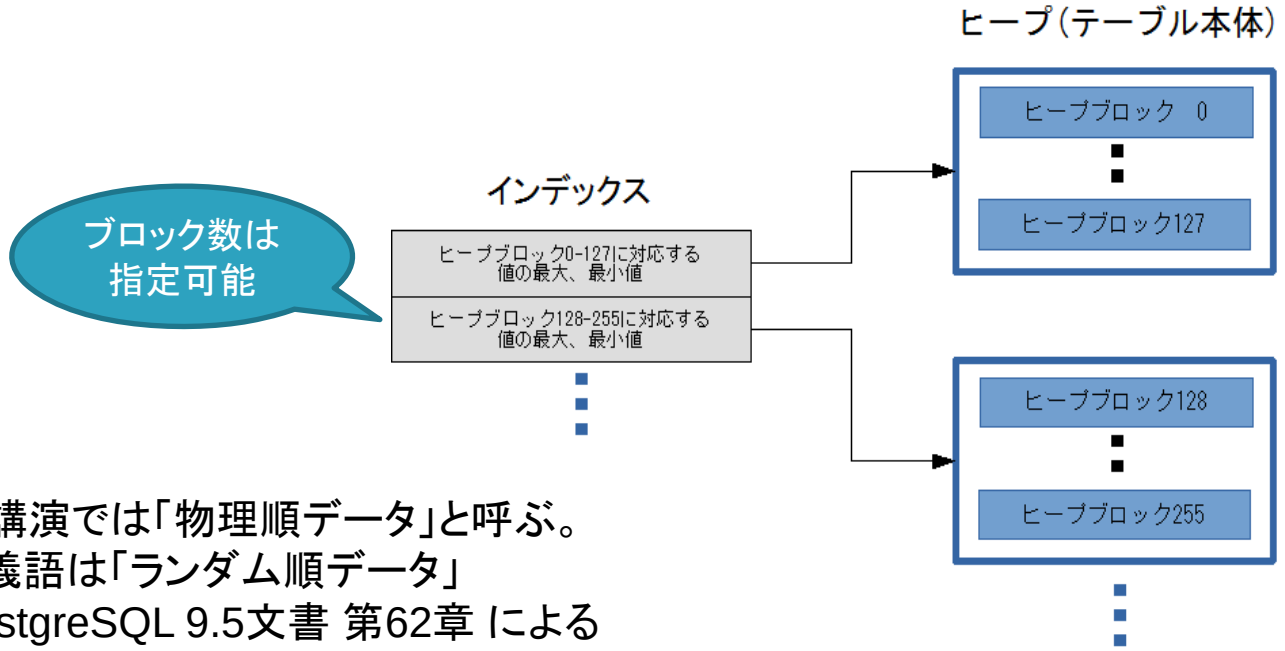
# BRIN の検証概要

- BRIN方式が有用と考えられる大規模データに対するデータ参照性能とインデックスの作成時間・サイズを測定
  - btreeおよびパーティションと比較
  - テーブルのデータ件数は 1億2千万件 (約 7.6GB)
  - データの特長として、物理順データ、ランダム順データについて測定
- データ参照性能の検証
  - BRIN, btree, インデックス無の3者比較
  - 1件検索
    - SELECT name FROM tbl WHERE id = ?
  - 範囲検索
    - SELECT name FROM tbl WHERE id BETWEEN ? AND ?

# BRIN (Block Range INdex) の概要

- BRINは一定数のヒープブロック内の値の最大値と最小値を保存し、参照時はその範囲をたどる
- 格納行の物理位置と相関関係を持ったデータ*1を持つ大規模テーブル向けとされている*2
  - 実際のデータで物理順となるのは(なりやすいのは) serial型カラム、時系列データの時刻カラムなど

『物理順データ』

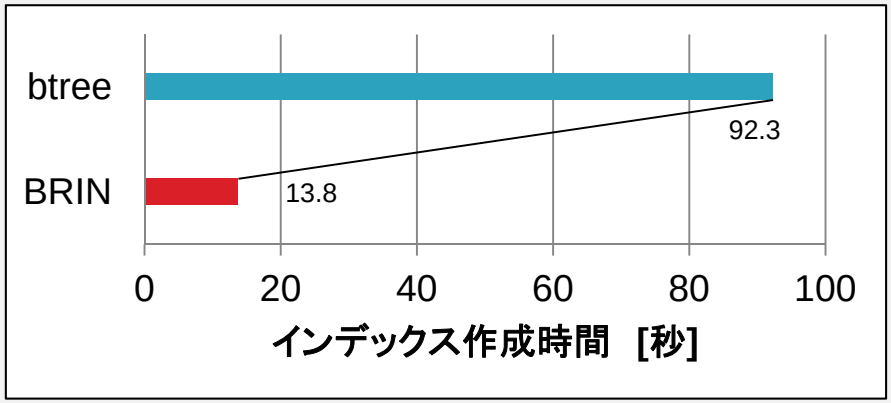



*1:本講演では「物理順データ」と呼ぶ。
対義語は「ランダム順データ」
*2:PostgreSQL 9.5文書 第62章 による

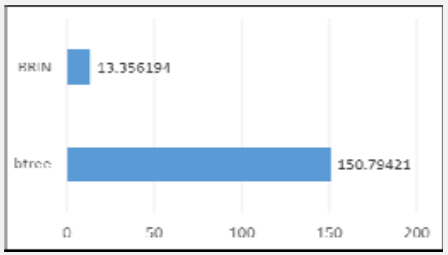
# 測定結果1：対btree比較～インデックス生成時間とサイズ～

- btreeに比べ、作成時間・サイズともに小さい
  - インデックス生成時間は1/7以下
  - インデックスサイズは1/10000以下
    - いずれも、BRINでは128ページを1ブロックとした場合

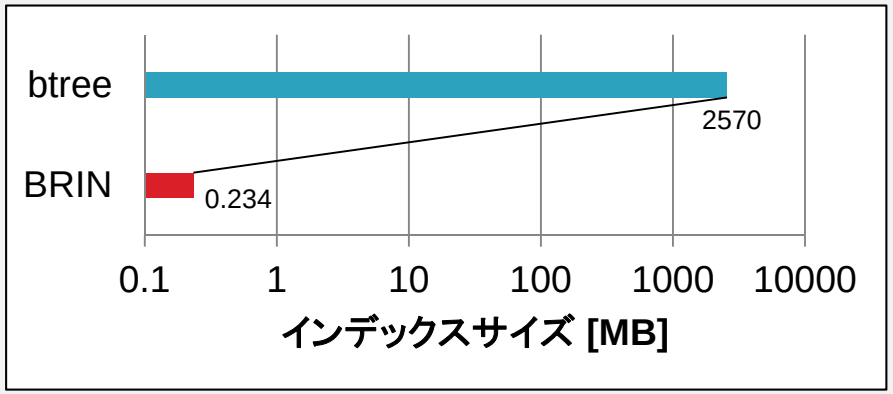
## インデックス作成時間



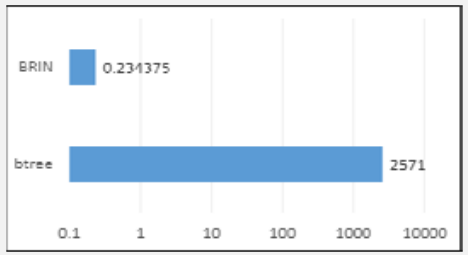

ランダムデータでも同傾向




## インデックスサイズ



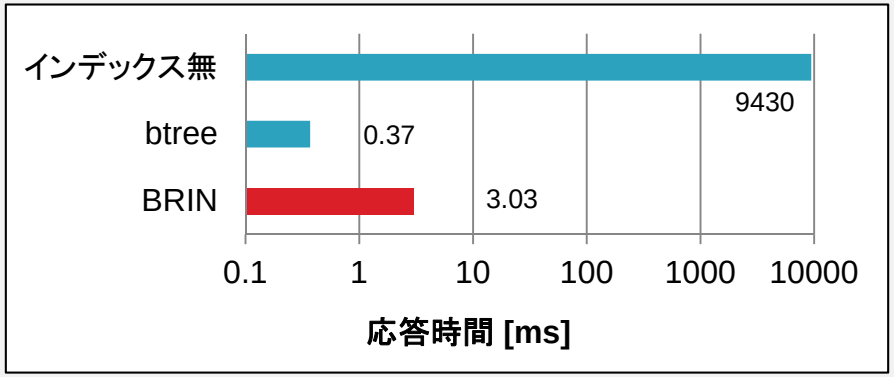

ランダムデータでも同傾向



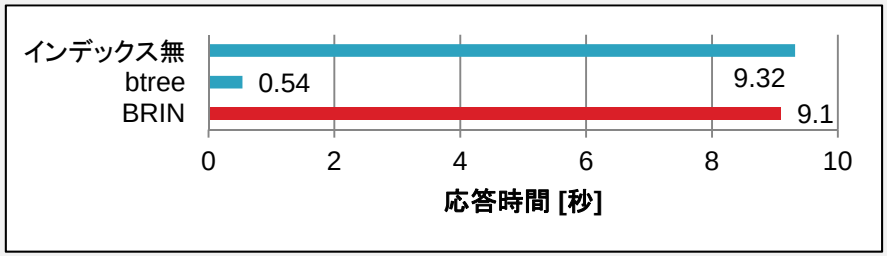
# 測定結果2：対btree比較～データ参照性能～

- 1件検索では、BRINはbtreeよりも遅いがインデックス無よりは大幅に高速
  - btreeに比べ約8.2倍、インデックス無に比べ約1/3100の応答時間
- 範囲検索では、btreeよりも高速

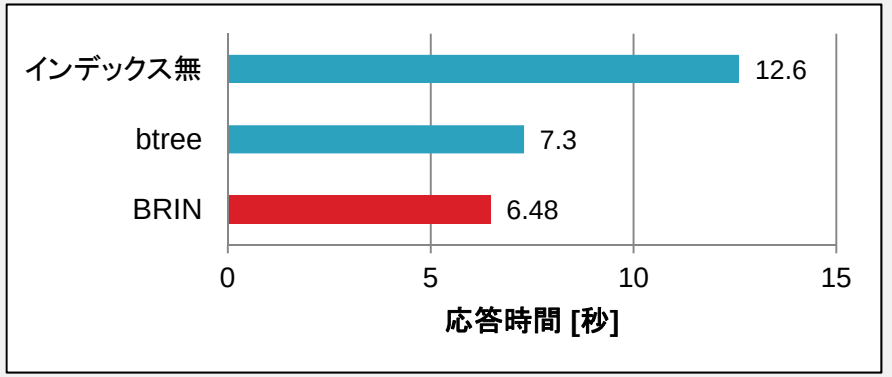
## 1件検索の応答時間



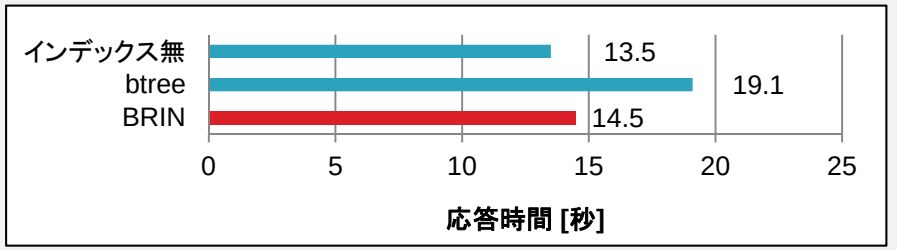

ランダムデータではインデックス無と同程度




## 範囲検索の応答時間




ランダムデータではインデックス無と同程度



# パーティショニングとの比較検証

- 大規模データの処理に利用されるケースの多いパーティショニングと比べたBRINの有用性を比較検証する
  - パーティショニング・単表・BRINの3者比較を実施
  - パーティショニングの評価には2013年度検証での知見を活用
- データ挿入性能（初期データ投入）
  - パーティショニングはトリガ利用でオーバーヘッドが大きいが、BRINはどうか
- データ参照性能
  - BRINを作成するだけでどこまでパーティショニングに迫れるか
  - 目的の値が存在しない集合を読み飛ばす絞り込みを行う両者に差はあるか

# 測定条件

## データ挿入、データ参照性能についてそれぞれ以下の条件で比較

1. パーティショニング(インデックスなし)
2. パーティショニング(btree)
3. 単表(BRIN)
4. 単表(btree)
5. 単表(インデックス無)

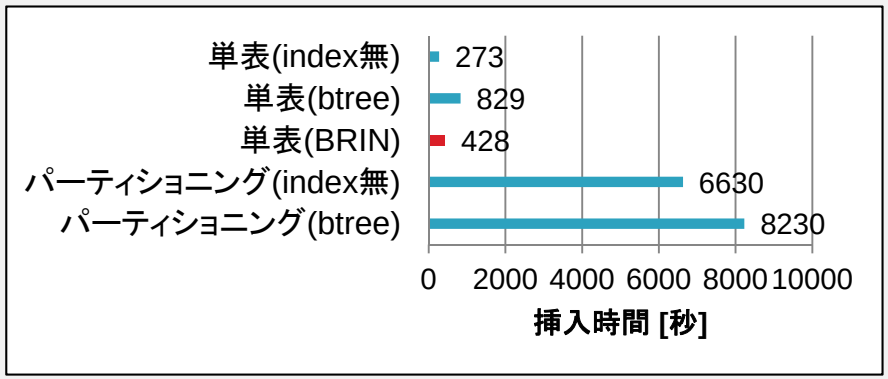
※単表とは非パーティショニングテーブルを示す。

- 検証方法は2013年度のパーティショニング検証にならう
  - 日々のログデータを1か月分保存するテーブルを想定
    - 1ヶ月分のデータ(15GB)を1日の子テーブルに分割
    - 1日分のデータは768万件(500MB)
  - パーティションに用いるトリガは速度の面からC言語関数トリガ
- データ挿入はCOPYコマンドで実施

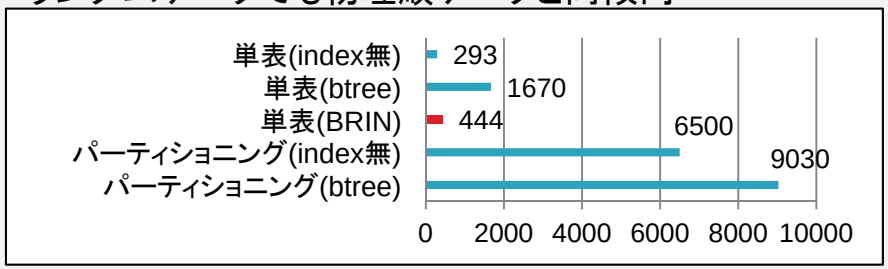
# 測定結果3：対パーティショニング比較

- 挿入性能（時間）はパーティショニングの1/15～1/20
  - 物理順、ランダム順とも同傾向
- 参照性能は、物理順データに関してはパーティショニングに匹敵
  - ランダム順の場合はインデックス無と同程度

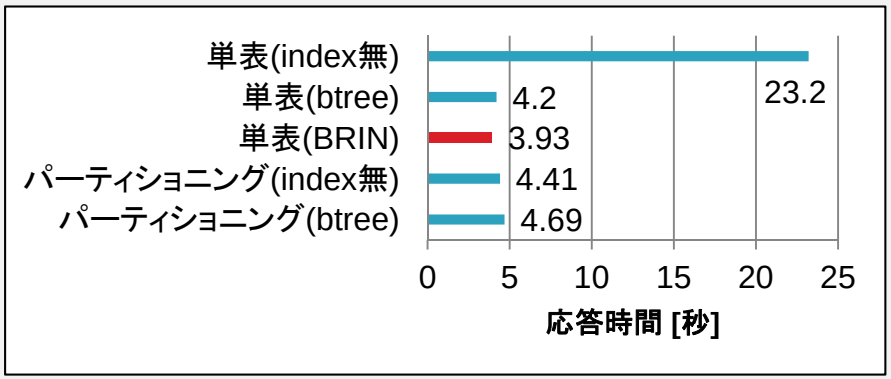
## 挿入性能



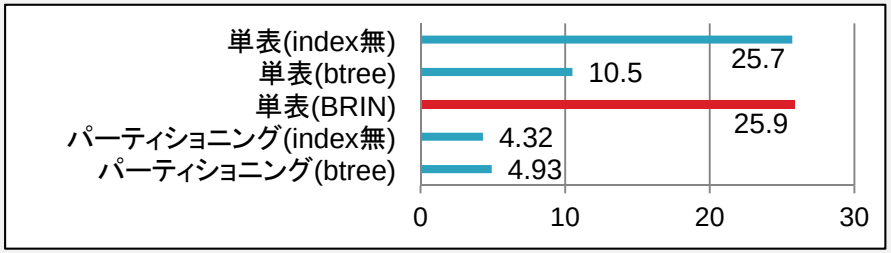

ランダムデータでも物理順データと同傾向




## 参照性能




ランダムデータではインデックス無と同程度



# 結果まとめ

- BRINは物理順データに用いれば効果大
  - 物理順データもしくはそれに近いデータに用いる
    ⇔データに依存せず性能を安定させたいなら使うべきでない
  - 最初は物理順でも更新するデータには不適（物理順が崩れるため）
- btreeに比べた利点
  - btree よりもインデックスのサイズが小さく、作成も速い
  - 範囲検索では btree 以上の性能
- パーティショニングに比べた利点
  - データ挿入時のトリガに依るオーバヘッドがない
  - パーティショニング作成に関する作業（テーブルの継承、トリガ作成）が不要で、簡易にパフォーマンスを出せる

活動報告5

# OS比較検証

# OS比較検証の背景・目的

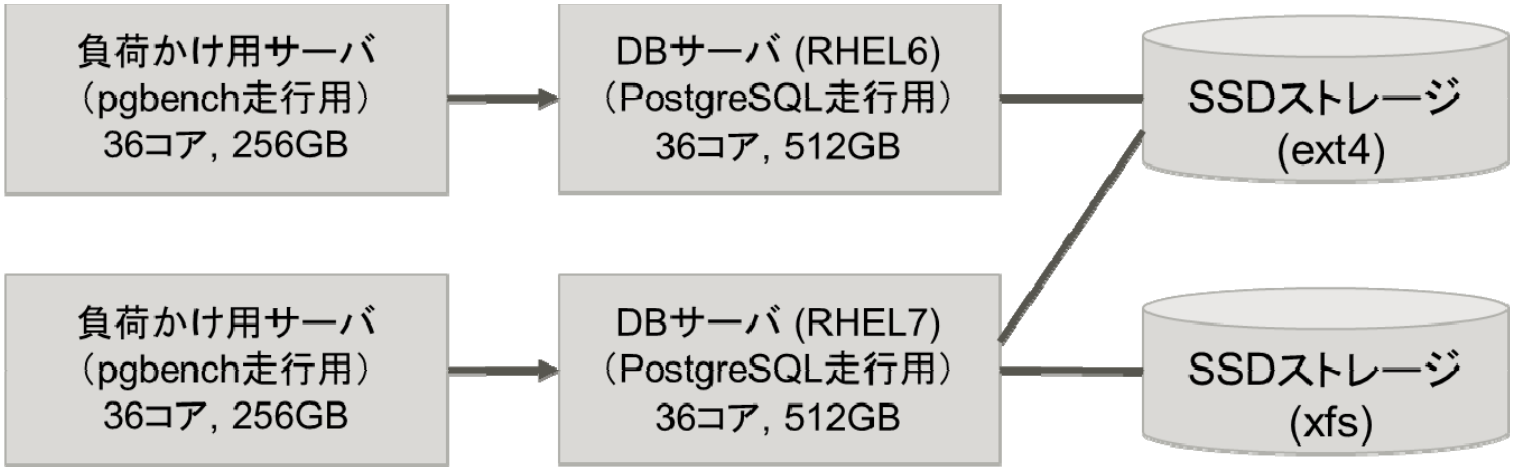
- Red Hat Enterprise Linux 7 (RHEL7) は、RHEL6から大幅な変更が加えられている

| 最新版 (2016年1月現在) | RHEL6.7 | RHEL7.2 |
|---|---|---|
| カーネルバージョン | 2.6.32 | 3.10.0 |
| ファイルシステム | ext4 (デフォルト) | xfs (デフォルト)<br>ext4 |
| I/Oスケジューラ | deadline<br>cfq (デフォルト)<br>noop | deadline (デフォルト)<br>cfq<br>noop |

- これらの差によりPostgreSQLの性能に差が見られるか検証
  1. OS (カーネルバージョン) の違いによる影響
  2. ファイルシステムの違いによる影響
  3. I/Oスケジューラの違いによる影響
- OS, ファイルシステム, I/Oスケジューラの各組合せで性能比較

# 測定条件・環境

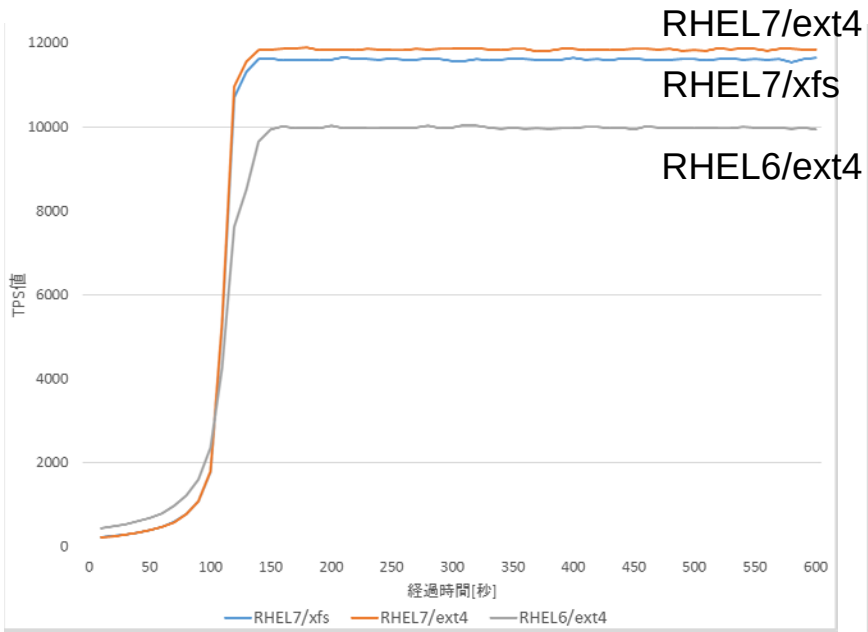
- マシンは以下の環境（検証環境２）を使用



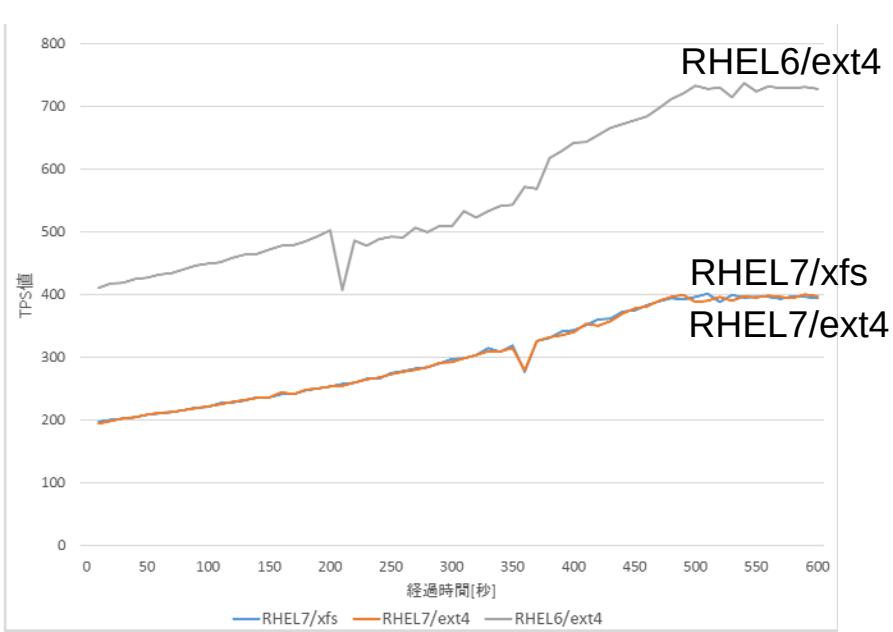

- PostgreSQLは9.5を使用
- 測定ツールはpgbenchを利用し、２種類のデータを作成
  - SF=6,000（約75GB）・・・ メモリに乗り切る
  - SF=60,000（約750GB）・・・ メモリに乗り切らない
- 測定方法
  - 10秒間の平均TPS値を連続的に測定し、その推移を観測
  - 参照系、更新系共に測定を行う

# 測定結果：OS/ファイルシステム比較（参照系）

- OSの差異による性能影響が大きいと考えられるが、データベースのサイズがオンメモリか否かによって傾向が逆転
  - SF=6,000の場合：RHEL7 > RHEL6… RHEL7/ext4 > RHEL7/xfs > RHEL6/ext4
  - SF=60,000の場合：RHEL6 > RHEL7 … RHEL6/ext4 > RHEL7/xfs > RHEL7/ext4



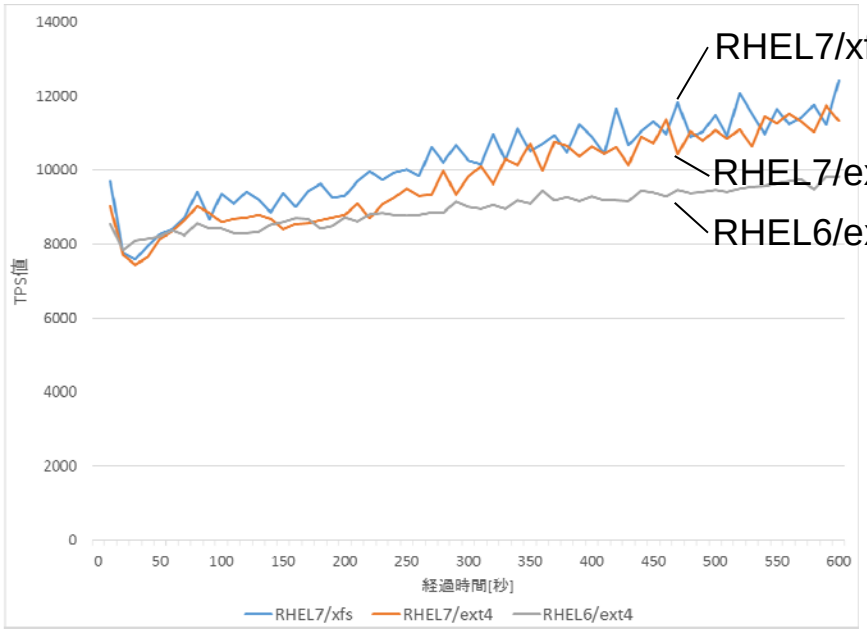

SF=6,000



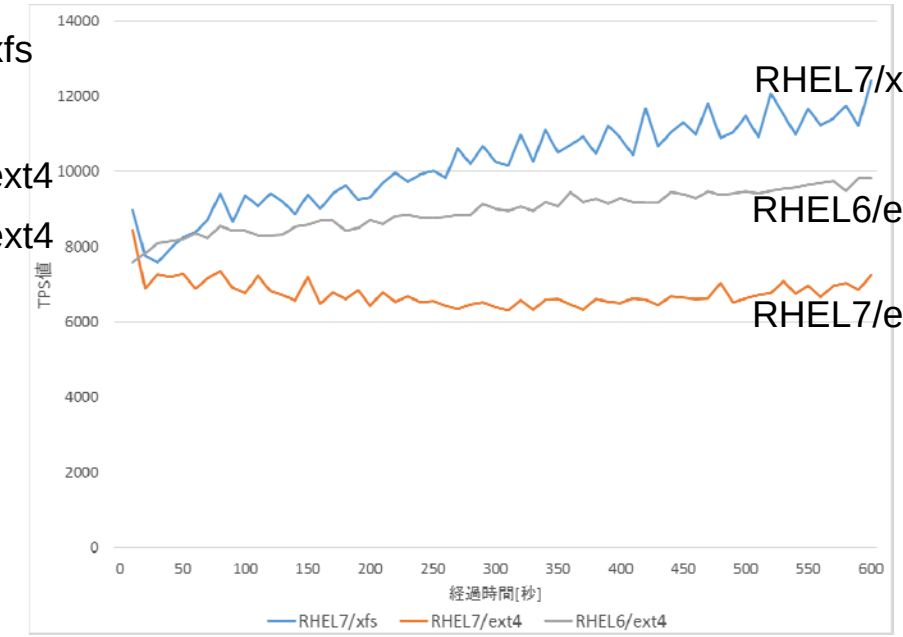

SF=60,000

# 測定結果：OS/ファイルシステム比較（更新系）

- ファイルシステムの差異による性能影響が大きいと考えられる
  - SF=6,000の場合： xfs > ext4 ・・・ RHEL7/xfs > RHEL7/ext4 > RHEL6/ext4
  - SF=60,000の場合： xfs > ext4 ・・・ RHEL7/xfs > RHEL6/ext4 > RHEL7/ext4
- I/O待ち時間（%iowait）はどの組合せでも90～95%で推移している




SF=6,000

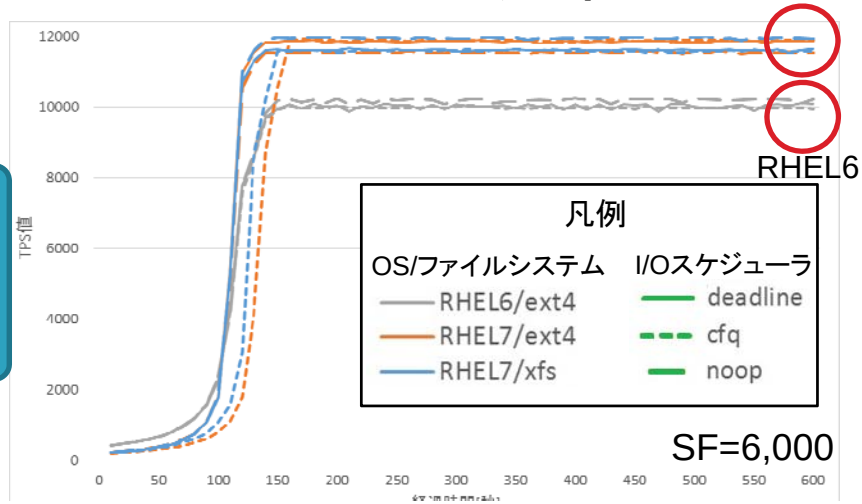


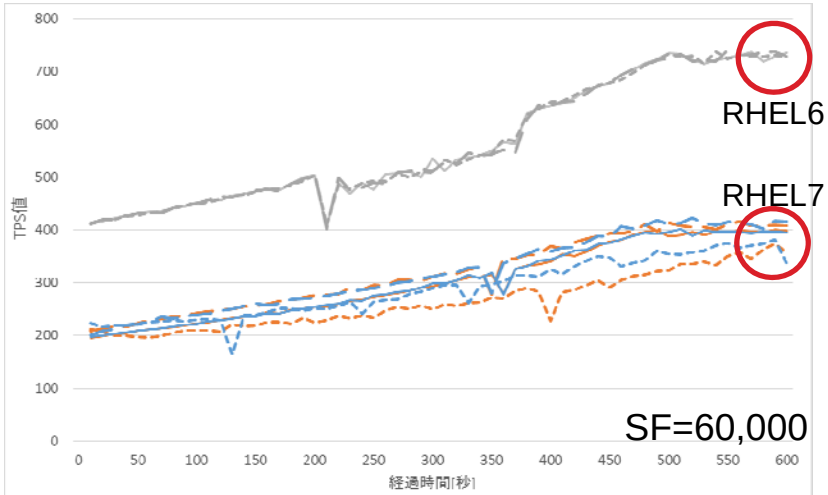

SF=60,000

# 測定結果：OS/ファイルシステム比較（更新系）

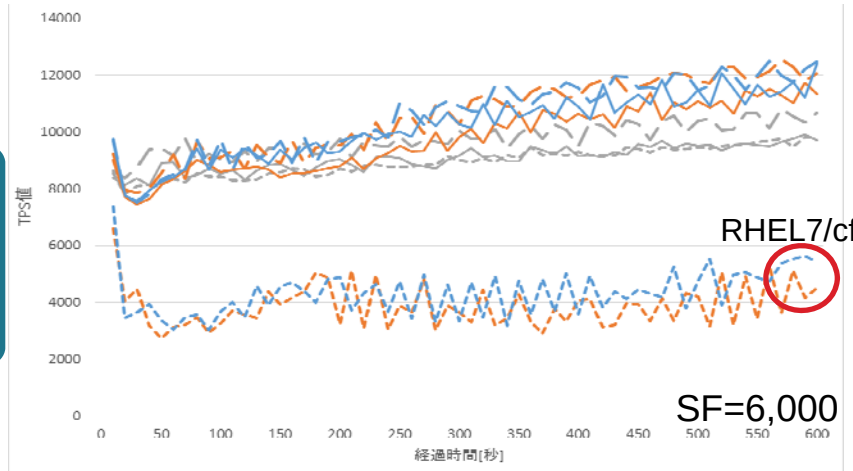
- 参照系の場合、I/Oスケジューラの変更によるTPS値の変化はほぼ見られない
- 更新系の場合、RHEL7/cfqの組合せのTPS値が圧倒的に低い

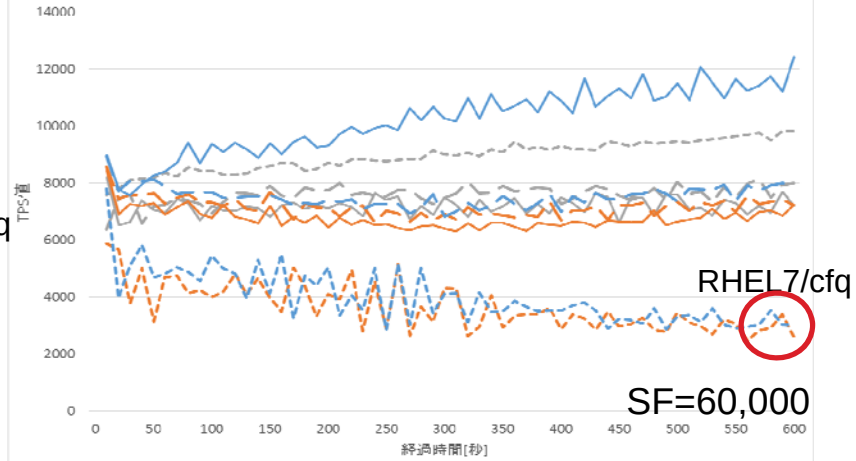
参照系




更新系




グラフの詳細は、2015年度WG1活動報告書をご参照ください

# 結果まとめ

- OS, ファイルシステム, I/Oスケジューラの組合せ (全9パターン) について性能測定を行った
- 参照系
  - OSの差異による性能影響が大きいと考えられる
    - このことは、RHEL6のファイルシステム、I/OスケジューラをRHEL7と同様の設定に変更してもTPS値の傾向が同等とならなかったことからも分かる
  - ただし、オンメモリか否かによって性能の順位は逆転
    - オールオンメモリ:RHEL7 > RHEL6, 左記以外:RHEL6 > RHEL7
- 更新系
  - RHEL7/xfsの組合せが最も高い性能が得られた
  - OSバージョンごとにI/Oスケジューラとの組合せ傾向が異なるため、最適な組合せ検証が必要
    - 今回の測定ではRHEL7/cfqの組合せが他に比べ性能が低くなることを確認
- 性能差の原因については特定できていない部分があるため、来年度以降の課題としたい

活動報告6

# 2015年度活動をふりかえって

# 2015年度活動をふりかえって

- PostgreSQL 9.5の検証を12月に計画したが、リリースが遅れて一部の検証でbeta版を使用しました
  - OS比較検証は9.5.0を使用
  - 9.6はリリースを早めるとの情報も
- 9.5新機能・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
- 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。

# 2015年度活動をふりかえって

- 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！

付録

# 今回使用した検証環境について

# 検証環境1 （提供：日本ヒューレット・パッカード株式会社）

## 定点観測（スケールアップ）参照系・更新系/Parallel VACUUM/BRIN Index検証での使用機器/設備

### 日本ヒューレット・パッカード ソリューションセンタ

**HPE ProLiant DL360 Gen9 (4台)**
CPU: Xeon E5-2650v3 (10Core) × 2 [20Core]
Memory: 128GB / HDD: 1.2TB 12G SAS 10k x 4

**HPE ProLiant BL460c Gen9 (3台)**
CPU: Xeon E5-2690v3 (12Core) × 2 [24Core]
Memory: 256GB / HDD: 1.8TB 12G SAS 10k x 2

**HPE ProLiant DL580 Gen9**
CPU: Xeon E7-8890v3 (18Core) × 4 [72Core]
Memory: 2TB / HDD: 1.2TB 12G SAS 10k x 10

10GbE Network Switch

**HPE 5700AF-32XGT**

16Gb FC Switch

16Gb FC Switch

VPN

**インターネット経由 リモートアクセス**

**HPE 3PAR StoreServ 8400**
HDD: 1.8TB 6G SAS 10K 2.5inch x 72

RAID10 for DL580 [定点観測/参照系・更新系] (3TB) x 4
RAID10 for BL460c#1 [BRIN Index] (3TB) x 4
RAID10 for BL460c#2 [PARTITION (BRIN対比)] (3TB) x 4
RAID10 for BL460c#3 [Parallel VACUUM] (3TB) x 4

# 検証環境2 （提供：富士通株式会社）

## ■ OS比較検証での使用機器

### 富士通トラステッド・クラウド・スクエア

メインフレームクラスの圧倒的な信頼性・可用性を備えた基幹IAサーバ

**FUJITSU Server PRIMEQUEST 2800E2**

| | |
|---|---|
| システムボード: | 4枚 |
| CPU: | Intel Xeon E7-8890v3 (2.50GHz, 18コア)×8 [144コア] |
| メモリ: | 1.5TB(512GB×2＋256GB×2) |
| 内蔵HDD: | 300GB 12G SAS 15k ×8 |

I/Oボトルネックを解消し、迅速かつ安定した処理を実現する
フラッシュメモリ搭載ストレージ

**FUJITSU Storage ETERNUS DX200F**

| | |
|---|---|
| コントローラー数: | 2 |
| SSD: | 1.6TB×24 SAS 12Gbits/s |
| キャッシュ: | 16GB |
| インターフェース: | FC接続 16Gbit/s |

# 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、富士通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