PGECons
PostgreSQL Enterprise Consortium

2018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
  - 過去のWG1活動テーマ
  - 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
  2. Windows版PostgreSQL性能検証
  3. JITコンパイル検証
  4. パラレルクエリ性能検証
- 2018年度活動をふりかえって
- Appendix
  - 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

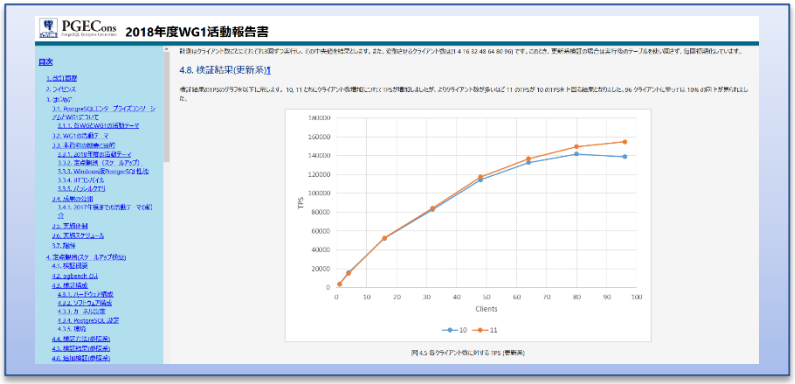
- 活動領域
  - 「大規模基幹業務に向けたPostgreSQLの適用領域の明確化」の中で特に「性能」について調査
  - 2016年度から「新技術検証WG」と名称を変え、PostgreSQLの新機能やトレンドを踏まえた活動テーマを選定
- 活動スタイル
  - 毎年のPostgreSQLの新バージョンにあわせて調査テーマを選定
    - PGEConsセミナーのアンケート結果も参考に
  - テーマごとに担当メンバを決めて初期検討・実機検証を実施
    - 実機検証は1か月程度
    - 測定データを共有の上で、PostgreSQLの内部構造を調査して課題・問題を洗い出す
  - 調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

過去のWG1活動テーマ

- 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

活動報告書のご紹介

- HTML, PDF で閲覧できます
  https://pgecons-sec-tech.github.io/tech-report/
- 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
- NECソリューションイノベータ株式会社
- NTTテクノクロス株式会社
- 日本電信電話株式会社
- 富士通株式会社

(企業名50音順・敬称略)

定点観測/スケールアップ検証成果紹介

検証概要

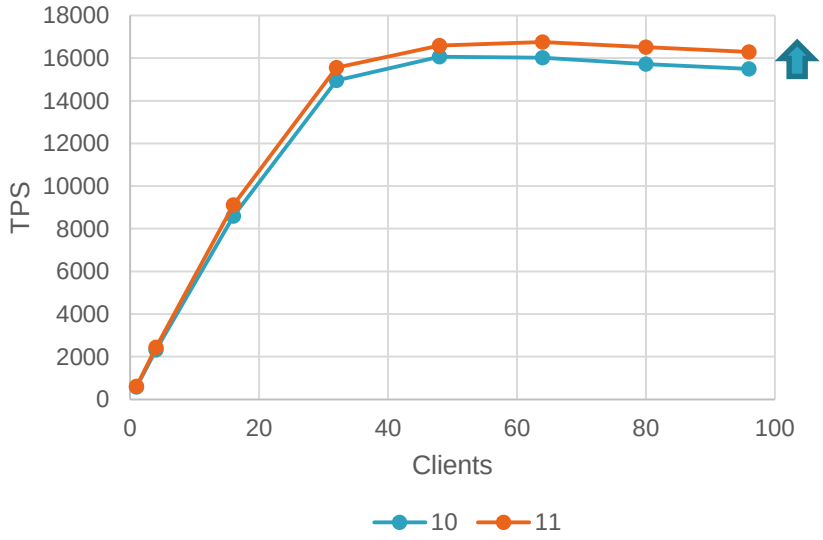
- 目的
  - メニーコアCPU上でのPostgreSQLのスケーラビリティを検証
  - 新バージョンのPostgreSQLの性能改善傾向を知る
    - PGECons発足当初（2012年度、PostgreSQL 9.2）から継続的に実施（定点観測）
      - 更新系性能に関する定点観測は2014年度から開始
    - バージョン間で性能差が現れたときはその要因も検証
  - 今年度はV10とV11の比較を実施

- 検証内容
  - 参照性能
  - 更新性能
  - （昨年度残課題）V10の更新性能劣化原因

検証結果

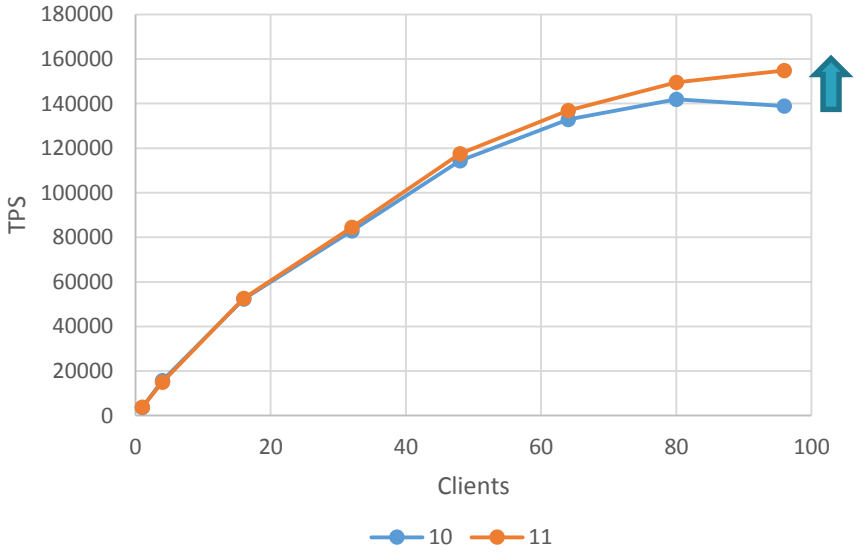
■ 参照系

□ 5%程度性能向上



■ 更新系

□ 10%程度性能向上

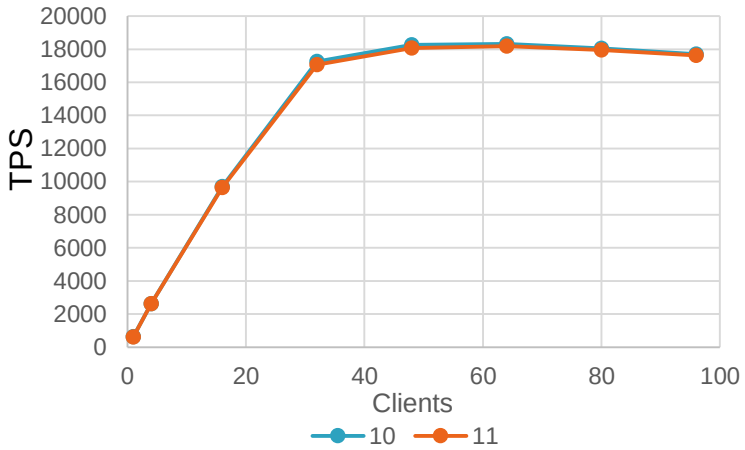


性能向上の要因は？

検証結果(性能向上要因追求)

- 参照系
  - 検証クエリに集約計算が含まれていた
    - V11は集約計算が改善されている
  - 集約計算を含まないクエリで再検証

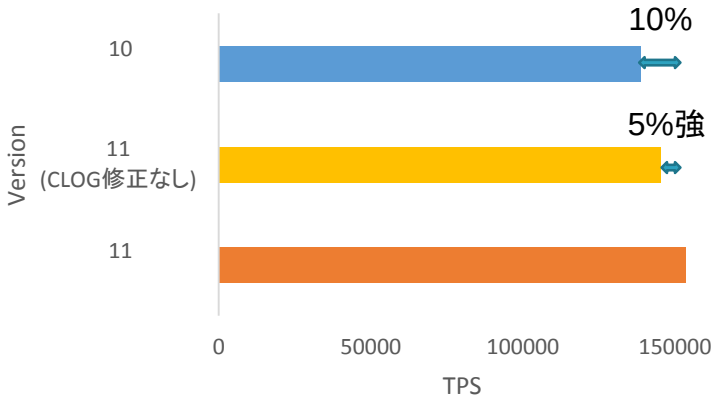
→参照性能変化なし



- 更新系
  - V11でCLOG処理の修正あり
  - 修正箇所のみ修正前の状態に戻したビルドで再検証

→V10/11間性能差の半分以上が現れた

→CLOG処理の修正が更新性能向上に大きく関わっていることを確認できた

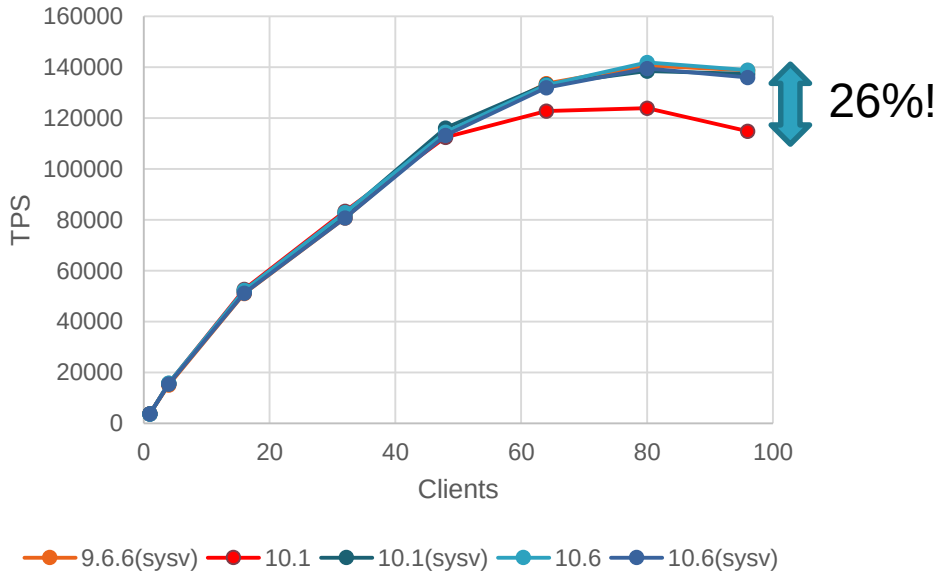


(96クライアントのみ)

(昨年度残課題) V10更新性能劣化原因検証

- V10の更新性能がV9.6に比べて劣化していた(2017年度検証)原因を調査
- デフォルトセマフォの変更(SYSV→POSIX)が原因と予想
  - セマフォの設定を変更したビルドで比較

  →証明できた
- (余談) V10.5でPOSIXセマフォでの性能劣化が修正された
- 以下のバージョンで比較
  - 9.6.6 (SYSV)
  - 10.1 (SYSV)
  - 10.1 (POSIX)
  - 10.6 (SYSV)
  - 10.6 (POSIX)



まとめ

- 参照性能
  - 一般的には性能変化なし
  - 集約計算のクエリに性能向上が見られる
- 更新性能
  - 10%程度の性能向上が見られた
  - CLOG に関する内部処理の修正が効果を発揮している
- (昨年度課題) V10の更新性能劣化原因
  - デフォルトセマフォの変更 (SYSV→POSIX) が原因だった
  - V10.5以降は変更後のセマフォでも性能が戻っている

V10を利用する場合、セマフォをSYSVに戻すかV10.5以降を使いましょう

Windows検証
成果紹介

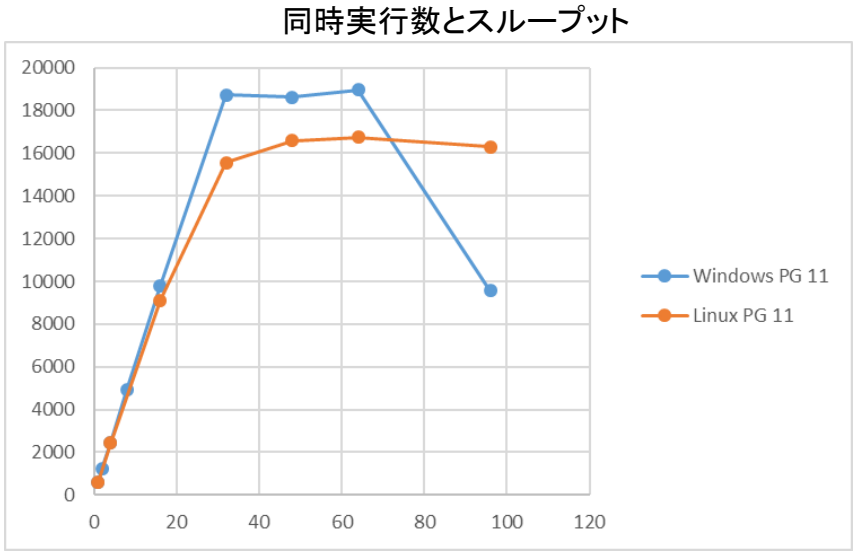
検証概要

- 目的
  - メニーコアCPU上でのWindows版PostgreSQLの性能傾向の調査
  - Linux版PostgreSQLとWindows版PostgreSQLを同一HW/同一ベンチマークを用いて比較する。
    - Linux版PostgreSQLの性能については「定点観測/スケールアップ検証」を参照。

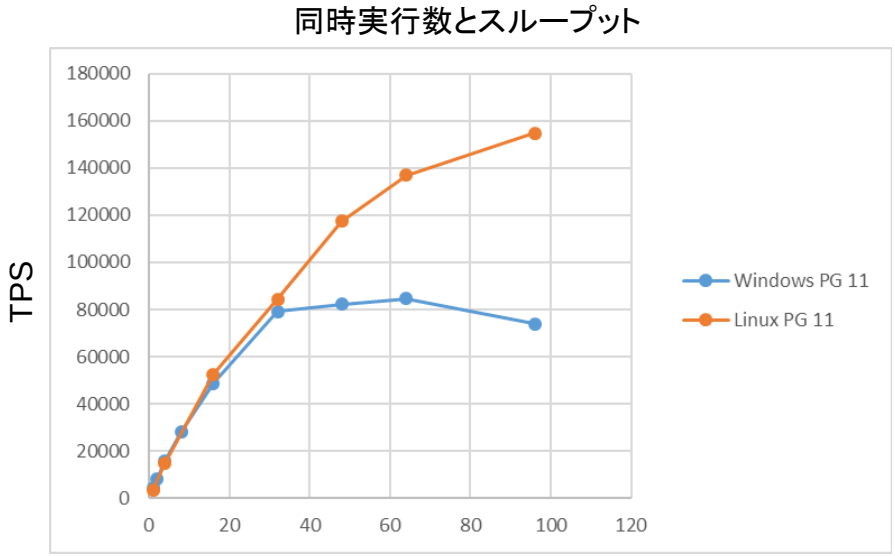
- 検証内容
  - 参照性能
  - 更新性能
  - 参考情報：EDB版インストーラとBigSQL版インストーラによる性能差

検証結果

- 参照系
  - 性能のピーク値はWindowsのほうがLinuxより高い。
  - 同時接続数が一定数を超えると急激に性能が低下



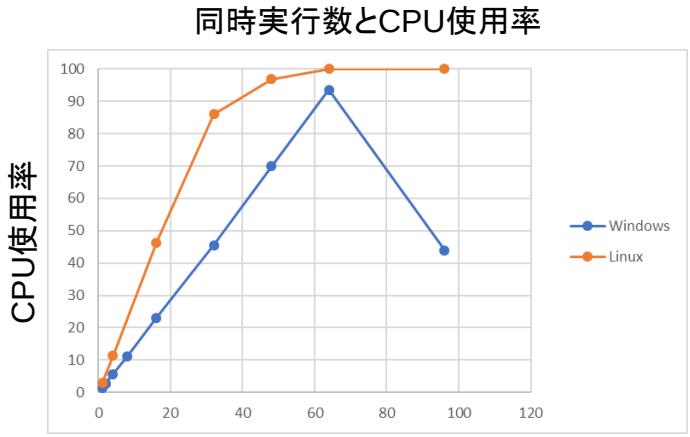
- 更新系
  - 同時接続数32までは性能は同等。
  - 同時接続数32を超えると性能は上がらなくなる。

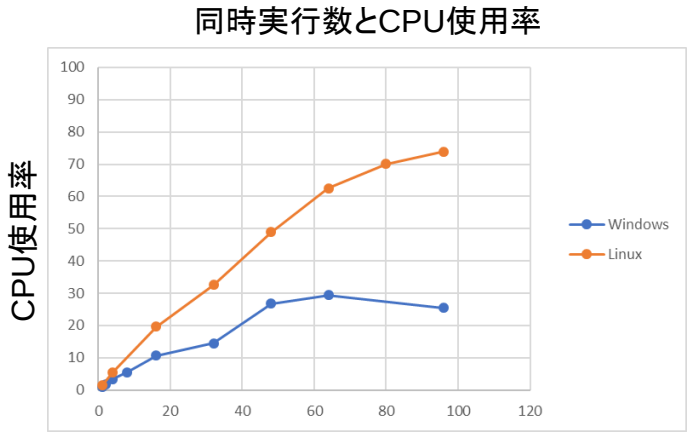


何が原因？

検証結果(リソース調査)

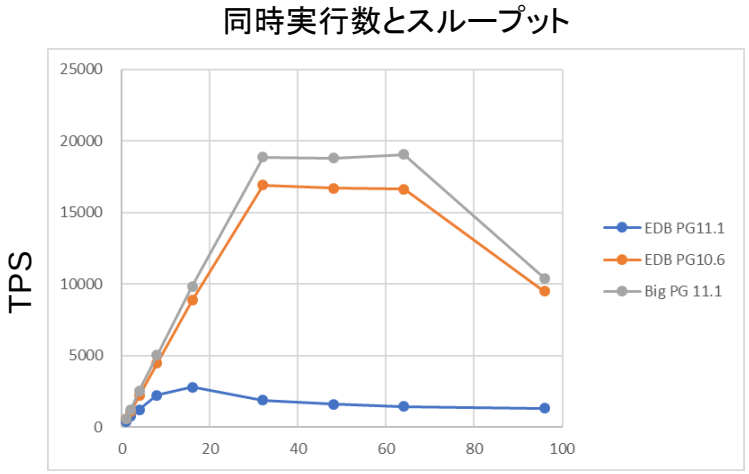
- 参照系
  - CPU使用率を見ると、同時接続数64までは上昇傾向。同時接続数64でほぼCPU使用率は上限に。
  - それ以上の同時接続数の場合、CPU使用率も低下。
  - プロセス排他待ち？
- 更新系
  - CPU使用率は全般にLinux版よりも低い傾向にある。
  - Windows OSの書き込みに何か原因があるのか？
  - ただ、今年度取得のOSリソース情報からは特定できず。





参考情報：EDB版インストーラとBigSQL版インストーラによる性能差

- 本測定前に、EDB版PostgreSQL 11.1のインストーラで検索系を測定したところ、PostgreSQL 10の結果と比較して性能がでないという問題を発見。
- BigSQL版PostgreSQL 11.1のインストーラを使用した場合には、この問題は発生せず。



- 本問題はPostgreSQL開発コミュニティMLに報告済み。
  - しかし、解決の報告はMLには流れず状況は不明のまま。

まとめ

- 参照性能
  - 同時接続数が64まではLinux版と同等の性能
  - 同時接続数が64を超えた場合には急激に性能が低下
- 更新性能
  - Linux版と異なり、同時接続数が32以上性能が伸びなくなる
  - CPU使用率はLinux版と比較すると全般に低めにとどまる
- 課題
  - 検索系
    - 同時接続数が過大になった場合に、CPU使用率が低下する原因の調査
  - 更新系
    - Linux版と比較すると全般に性能/CPU使用率が上がらない原因の調査
  - Windows OSリソースの取得内容/方式の検討(継続)
  - Windows版インストーラの差異による性能差

JITコンパイル検証
成果紹介

JITコンパイルとは

- 概要
  - クエリの実行時にコンパイルを実行し、高速化を図る機能
  - 実行計画のコストが、設定した閾値を超えた場合に採用
    - コンパイル時にオーバヘッドが発生
    - ある程度以上の規模のクエリでないと逆効果
  - CPUを多く使用する大規模クエリに有効
- 高速化の対象
  - 式評価
    - WHERE句
    - ターゲットリスト
    - 集約(sum, count, avg, …)

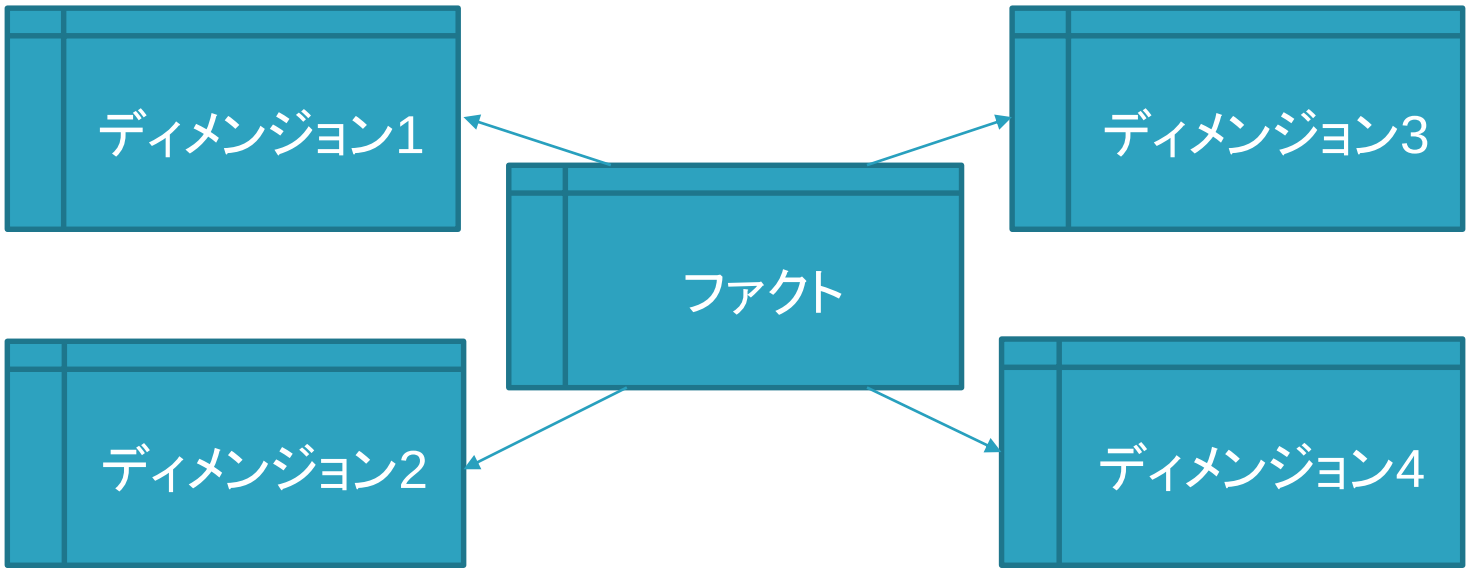
    など
  - タプル分解:ディスク上のタプルをメモリ上の表現に変換

検証概要

- 検証目的
  - 検証１：JITコンパイルが有効に働くクエリ種を調査する
    - 簡単なクエリで検証
      - 四則演算、集約関数を使用
      - 例：SELECT sum(col1) FROM …
  - 検証２：JITコンパイルによる実行時間の削減率を検証する
    - スタースキーマベンチマーク(SSB)で定義されたクエリで検証
      - OLAP系の大規模かつ実用的なクエリ
      - テーブルの構造については後述
- 検証方法
  - 事前にpg_prewarmを実行
  - クエリはpsqlコマンドによって実行

スタースキーマベンチマーク（SSB）とは？

- スタースキーマベンチマーク（SSB）のベースはTPC-H
  - ファクトテーブル：分析対象となる実績値が格納
  - ディメンジョンテーブル：ビジネス上の軸になる詳細なマスターデータが格納



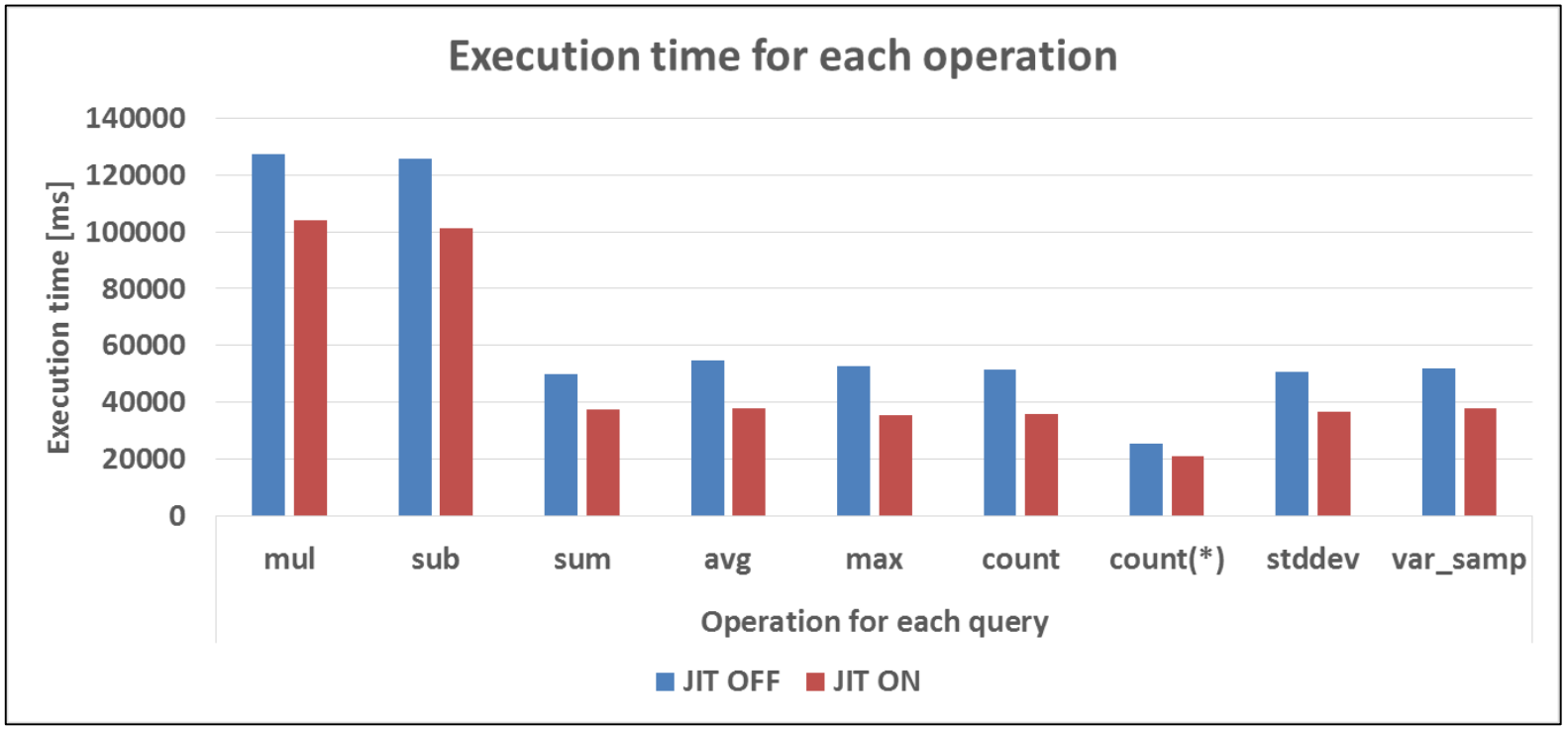
- テーブルサイズ
  - Scale Factor=100の時、全体で約60GB（テーブル物理サイズ）
    - 大部分がファクトテーブルに格納

スタースキーマベンチマーク（SSB）とは？

- SSBでのクエリは4パターン、計13本が定義されている。
  - q1（3本）
    ファクトテーブルとディメンジョンテーブル1つとの結合
    ファクトテーブルの選択率は0.008%（q1.3）～1.95%（q1.1）
  - q2（3本）
    ファクトテーブルとディメンジョンテーブル3つとの結合
    ファクトテーブルの選択率は0.02%（q2.3）～0.8%（q2.1）
    集約、ソートあり
  - q3（4本）
    ファクトテーブルとディメンジョンテーブル3つとの結合
    ファクトテーブルの選択率は0.0006%（q3.4）～3.4%（q3.1）
    集約、ソートあり
  - q4（3本）
    ファクトテーブルとディメンジョンテーブル4つとの結合
    ファクトテーブルの選択率は0.0009%（q4.3）～1.6%（q4.1）
    集約、ソートあり

検証１結果：クエリ種の調査

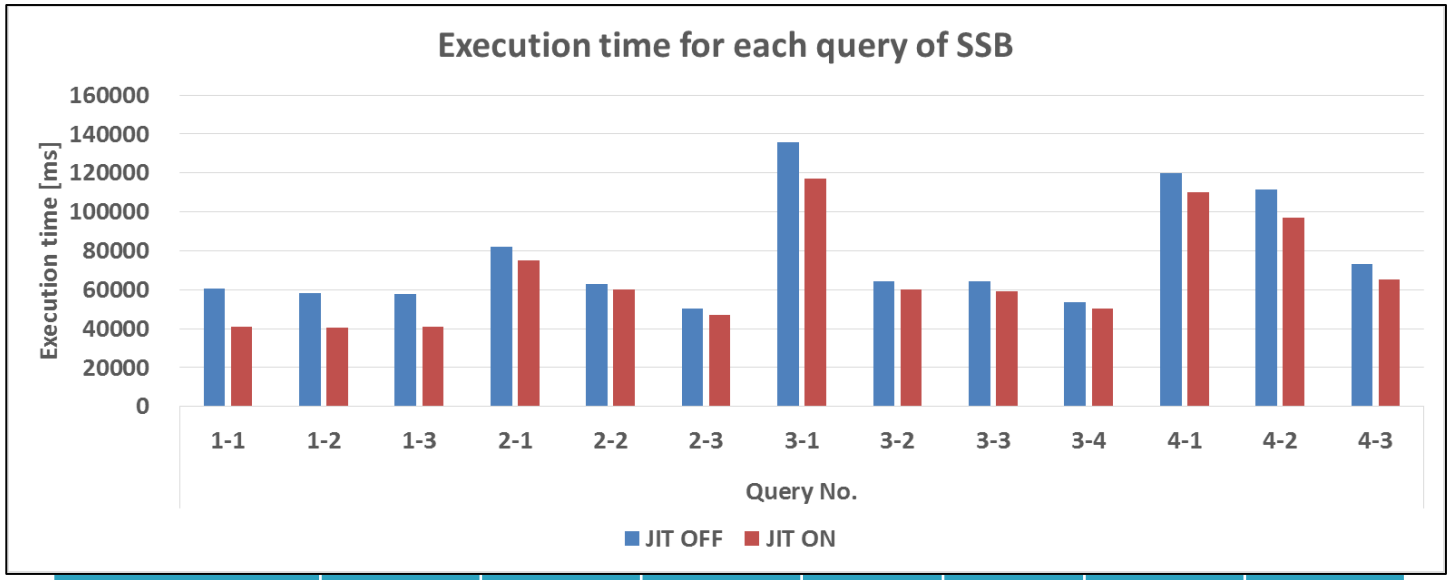
- すべての演算でJITコンパイルは有効に動作
  - 処理時間の大小にかかわらず、概ね2～3割の削減率



演算/関数	mul	sub	sum	avg	max	count	count (*)	stddev	var_samp
削減率 (%)	18.2	19.5	25.0	30.3	32.4	30.8	16.5	27.8	27.5

検証2結果：OLAP系クエリでの削減効果

- クエリによって削減効果のばらつきが見られた
  - 追加検証ではパラメータ(shared_buffersなど)を調整したが、改善せず → WHERE句などのクエリの構造が影響？



クエリNo.	1-1	1-2	1-3	2-1	2-2	2-3	
削減率(%)	32.1	30.5	29.5	8.5	4.1	6.6	
クエリNo.	3-1	3-2	3-3	3-4	4-1	4-2	4-3
削減率(%)	13.6	6.7	7.7	6.6	8.4	12.8	11.1

まとめ

- 四則演算、集約関数には効果あり
  - 実行時間が2～3割削減
- OLAP系の大規模クエリには効果のばらつきあり
  - クエリの構造（WHERE句など）が影響すると予想

- 今後、以下の検証も必要だと認識
  - 自作関数（PL/pgSQLなど）に効果があるか
  - インデックスを作成した場合の削減率はどのようなものか
- 新機能のため、今後の機能拡張に期待

パラレルクエリ検証
成果紹介

検証の目的

- PostgreSQLでのパラレルクエリとは
  - 1つのクエリを複数のプロセスで分担して並列処理すること
    - パラレルクエリで効率的に処理可能と判断した場合のみ採用
  - BIなどのOLAP用途で恩恵を受けやすい
  - PostgreSQL 9.6で採用され、10以降で継続的に強化されている

- PostgreSQLがOLAP用途で実用的に使用可能となったのかを検証するため、3種類の検証を試行
  - 検証A：パラレルクエリ関連のパラメータを、9.6は有効化、10以降はデフォルトのままで、パラレルクエリでの処理性能の改善状況を確認
  - 検証B：並列度を設定するパラメータを変更し、並列度の違いによる処理時間を比較
  - 検証C：B-treeインデックスのパラレル作成を行い、処理性能の改善状況を確認

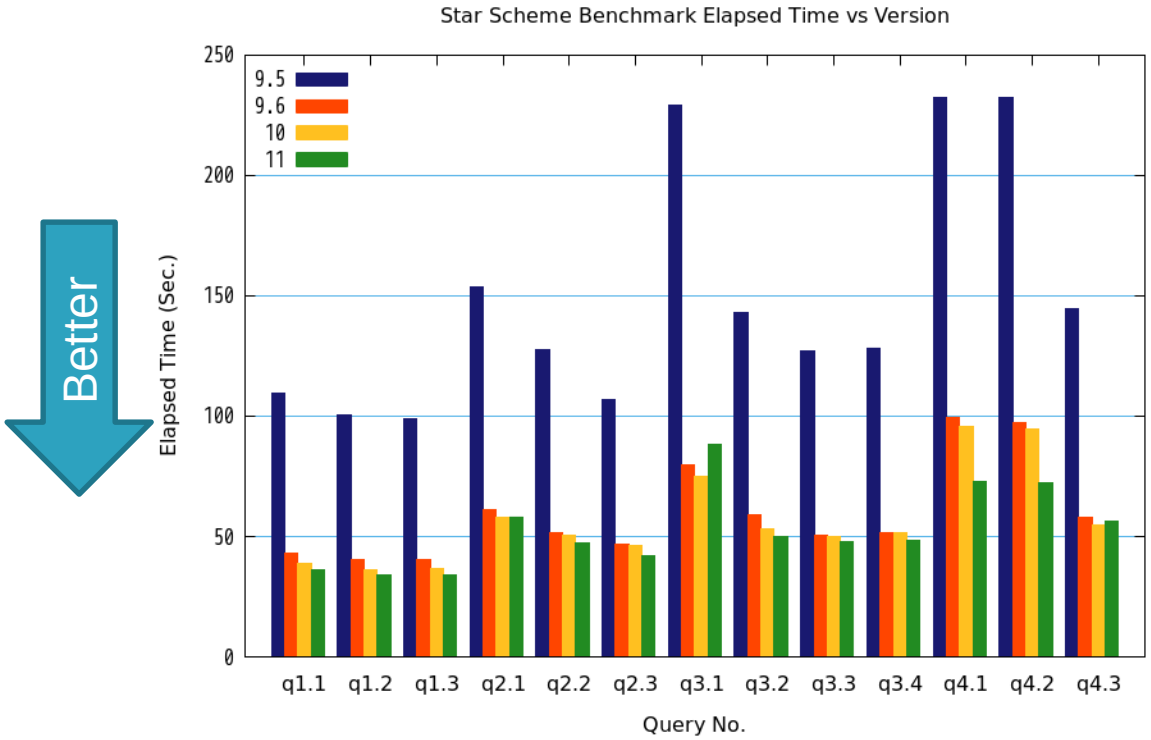
検証方法と環境

- 検証方法
  - 検証A・Bでは、Star Schema Benchmark (SSB) に定義されている クエリ 4パターン／13本のクエリを使用する
    (SSBに関してはJITコンパイル検証にて紹介済みのため割愛)
  - 検証Cではファクトテーブルに対して、インデックスの作成対象列を変化させたCREATE INDEX文を実行する
  - DBサーバとクライアントは同一で、psqlコマンドでクエリを実行する
  - Scale Factorは 100 とする
    - Scale Factor=1で概ね1GBのデータサイズが生成される
    - テーブルに格納すると、60GB程度の物理データサイズとなる

- 検証環境
  - 定点観測における、DBサーバと同一

検証A パラレルクエリでの処理性能の改善状況

- パラレルクエリの有効化によって、9.6以降の全バージョンにおいて、全クエリが9.5比で概ね2倍強～3倍程度の高速化



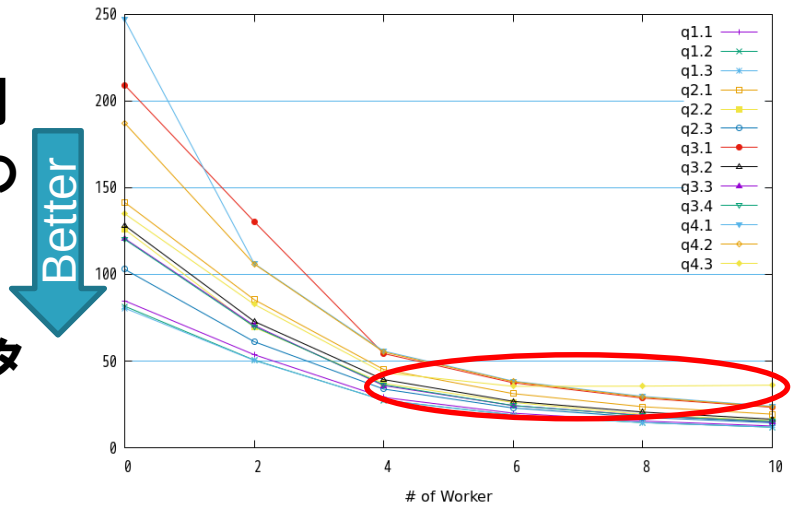
- パラレルクエリ無効の状態でも、v9.5比でv10が最大20%程度、v11が最大30%程度高速化していることが判明

検証B 並列度の違いによる処理時間の比較

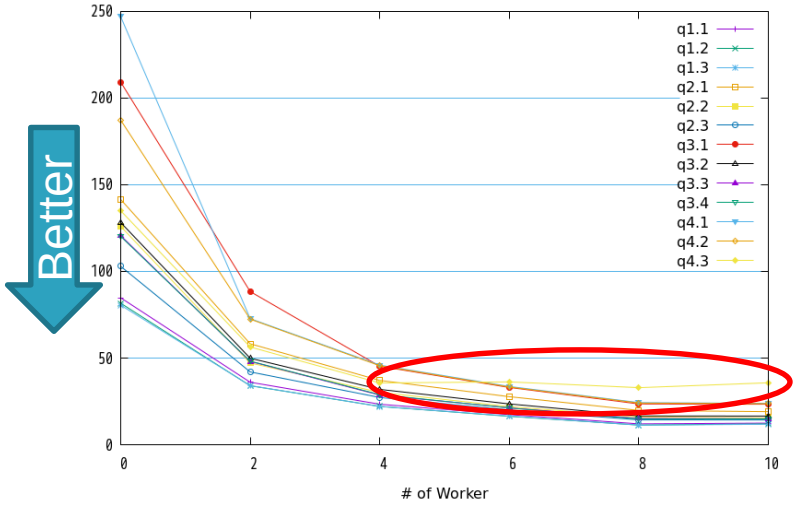
- 10、11ともに並列処理数の増加とともに処理時間が減少する傾向
  - グラフ形状からは並列処理数と処理時間の間に反比例に近い関係が示唆される
  - max_parallel_workersパラメータがmax_parallel_workers_per_gatherパラメータよりも大きい場合の方が、同一値としている場合よりも高速な傾向

- 11におけるq4.3は並列処理数4で性能が頭打ちに
  - v10とv11で結合順が変化、実行計画上のワーカー数がそれに応じて変化 (9→4)
  - 実行計画で決定されたワーカー数までしか実行時に使用しないため、性能が頭打ちに
  - ワーカー数を決定するロジックの改良を開発コミュニティに働きかける

処理時間とワーカー数の関係 (v11, Worker=Gather)



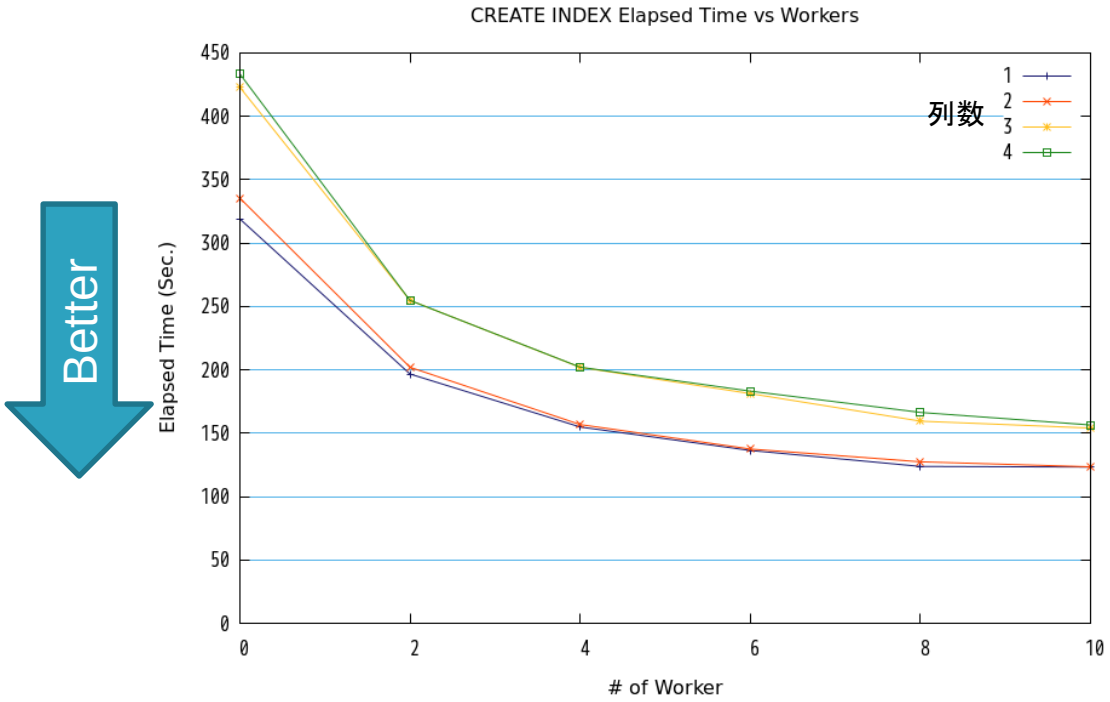
処理時間とワーカー数の関係 (v11, Worker=Gather×4)



※ 上下のグラフとも、Worker=0はパラレルクエリなしを示す。

検証C B-Treeインデックスのパラレル作成

- 並列処理数の増加とともに処理時間が減少する傾向
  - グラフ形状からは並列処理数と処理時間の間に反比例に近い関係が示唆される
  - 処理時間の大小は、作成されるインデックスの大きさ（一意となる組み合わせの数）に依存する



列数	対象列	組み合わせ数
1	lo_orderkey	150,000,000
2	lo_orderkey lo_custkey	150,000,000
3	lo_orderkey lo_custkey lo_partkey	600,036,916
4	lo_orderkey lo_custkey lo_partkey lo_suppkey	600,038,145

まとめ

- パラレルクエリの使用は注意点が多いが実用に耐えうる
  - 効果のあるクエリであれば、デフォルトパラメータでも2倍～3倍高速化
    - チューニングでさらに高速化する余地あり
  - 大量のデータ内から目的の情報を探索・導出するといった作業を効率的に行えるようになったものと確信
    - 検索、結合、ソート、集約演算
    - ハッシュテーブル共有化
  - パラレルクエリの効果が十分に得られるか、確認が必要
- 現状のパラレルクエリの実装においてOLAP用途で使用可能
  - バグと考えられる挙動も見られることから、常に最新バージョンでの使用を推奨
- パラレルクエリのさらなる進化、プランナ・エグゼキュータのさらなる深化を働きかける

2018年度活動をふりかえって

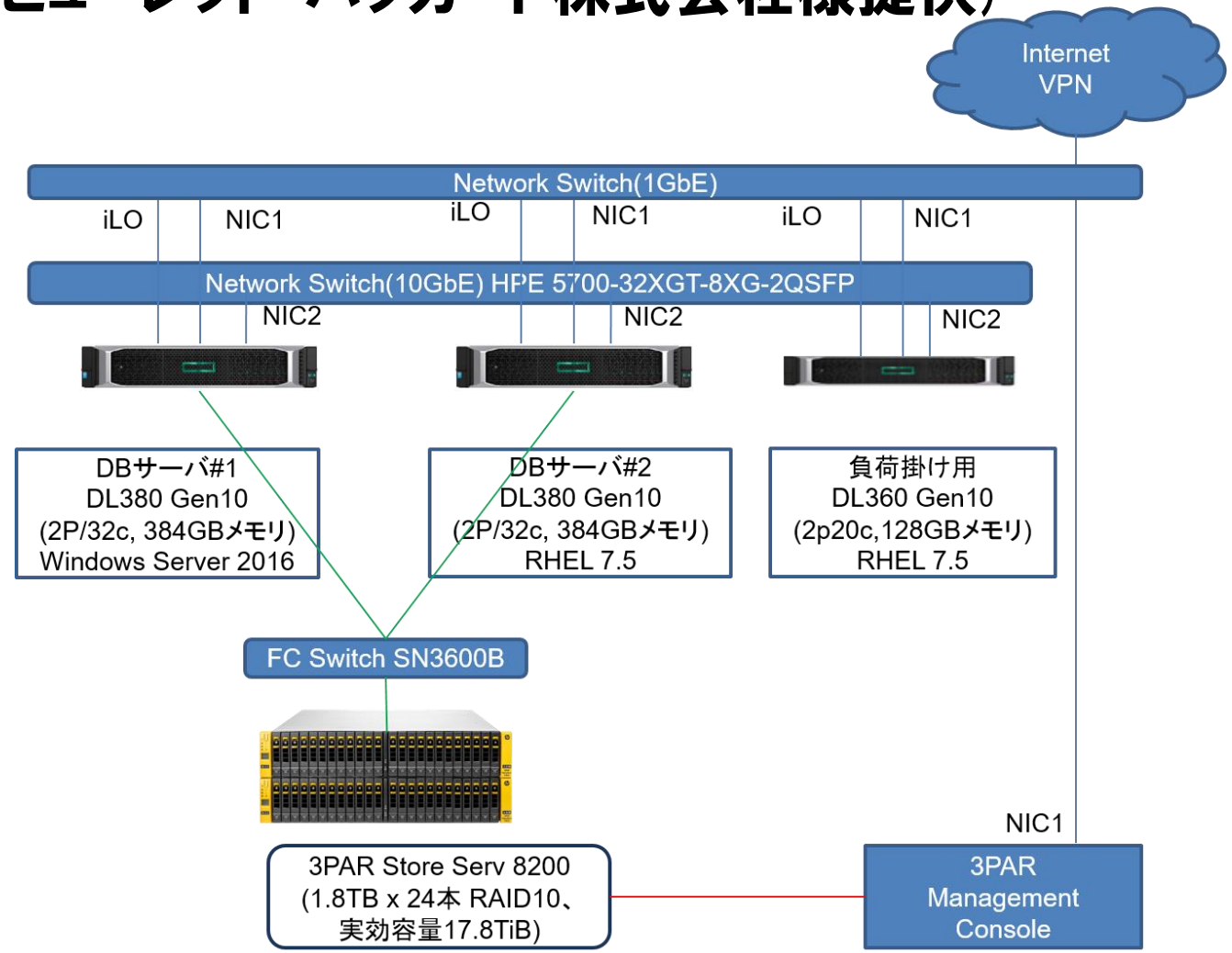
2018年度活動をふりかえって

- PostgreSQL 11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
    - 特にJITコンパイル機能
  - 検証テーマごとに測定パターンから議論
  - 測定結果からプロファイリングを含めた分析

- 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！

Appendix

スケールアップ検証，Windows検証，パラレルクエリ検証環境（日本ヒューレット・パッカード株式会社様提供）



JITコンパイル検証（富士通株式会社様提供）

機種	FUJITSU Server PRIMERGY RX2540 M4
CPU	インテル Xeon Gold 6140 CPU@2.3GHz（36Core）
メモリ	256GB
ストレージ	SSD 700GB
OS	RHEL 7.5

検証2：No.1-1の実行計画

- JIT:ON

```
                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
Aggregate  (cost=18719494.34..18719494.35 rows=1 width=8) (actual time=44864.530..44864.530 rows=1 loops=1)
  Buffers: shared hit=7758524
  ->  Hash Join  (cost=74.19..18663739.86 rows=11150896 width=8) (actual time=112.756..44343.859 rows=11913972 loops=1)
        Hash Cond: (lineorder.lo_orderdate = date.d_datekey)
        Buffers: shared hit=7758524
        ->  Seq Scan on lineorder  (cost=0.00..18259561.20 rows=78025470 width=12) (actual time=112.507..39017.385 rows=78550619 loops=1)
              Filter: ((lo_discount >= 1) AND (lo_discount <= 3) AND (lo_quantity < 25))
              Rows Removed by Filter: 521487283
              Buffers: shared hit=7758486
        ->  Hash  (cost=69.66..69.66 rows=362 width=4) (actual time=0.225..0.226 rows=365 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 21kB
              Buffers: shared hit=38
              ->  Seq Scan on date  (cost=0.00..69.66 rows=362 width=4) (actual time=0.043..0.188 rows=365 loops=1)
                    Filter: (d_year = 1993)
                    Rows Removed by Filter: 2191
                    Buffers: shared hit=38
Planning Time: 0.101 ms
JIT:
  Functions: 18
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 1.511 ms, Inlining 1.941 ms, Optimization 72.381 ms, Emission 38.003 ms, Total 113.835 ms
Execution Time: 44866.132 ms
```

高速化

JITによるオーバーヘッド

- JIT:OFF

```
                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
Aggregate  (cost=18719494.34..18719494.35 rows=1 width=8) (actual time=64400.234..64400.235 rows=1 loops=1)
  Buffers: shared hit=7758524
  ->  Hash Join  (cost=74.19..18663739.86 rows=11150896 width=8) (actual time=0.252..63787.222 rows=11913972 loops=1)
        Hash Cond: (lineorder.lo_orderdate = date.d_datekey)
        Buffers: shared hit=7758524
        ->  Seq Scan on lineorder  (cost=0.00..18259561.20 rows=78025470 width=12) (actual time=0.007..58197.223 rows=78550619 loops=1)
              Filter: ((lo_discount >= 1) AND (lo_discount <= 3) AND (lo_quantity < 25))
              Rows Removed by Filter: 521487283
              Buffers: shared hit=7758486
        ->  Hash  (cost=69.66..69.66 rows=362 width=4) (actual time=0.238..0.238 rows=365 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 21kB
              Buffers: shared hit=38
              ->  Seq Scan on date  (cost=0.00..69.66 rows=362 width=4) (actual time=0.032..0.206 rows=365 loops=1)
                    Filter: (d_year = 1993)
                    Rows Removed by Filter: 2191
                    Buffers: shared hit=38
Planning Time: 0.106 ms
Execution Time: 64400.272 ms
```

検証2：No.2-2の実行計画

- JIT:ON

```
                                                                         QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------------------
GroupAggregate  (cost=16129050.94..16131822.99 rows=6713 width=21) (actual time=88508.083..88720.456 rows=56 loops=1)
  Group Key: date.d_year, part.p_brand1
  Buffers: shared hit=7792538
  ->  Sort  (cost=16129050.94..16129727.17 rows=270492 width=17) (actual time=88505.310..88627.316 rows=961749 loops=1)
        Sort Key: date.d_year, part.p_brand1
        Sort Method: quicksort  Memory: 99713kB
        Buffers: shared hit=7792538
        ->  Hash Join  (cost=70125.25..16104645.50 rows=270492 width=17) (actual time=1050.155..87659.761 rows=961749 loops=1)
              Hash Cond: (lineorder.lo_orderdate = date.d_datekey)
              Buffers: shared hit=7792538
              ->  Hash Join  (cost=70030.26..16100831.24 rows=270492 width=17) (actual time=750.644..87139.416 rows=961749 loops=1)
                    Hash Cond: (lineorder.lo_suppkey = supplier.s_suppkey)
                    Buffers: shared hit=7792500
                    ->  Hash Join  (cost=41091.26..16064069.52 rows=1364746 width=21) (actual time=622.302..84999.357 rows=4813031 loops=1)
                          Hash Cond: (lineorder.lo_partkey = part.p_partkey)
                          Buffers: shared hit=7778538
                          ->  Seq Scan on lineorder  (cost=0.00..13759100.40 rows=600061440 width=16) (actual time=0.014..30166.398 rows=600037902 loops=1)
                                Buffers: shared hit=7758486
                          ->  Hash  (cost=41051.46..41051.46 rows=3184 width=13) (actual time=622.161..622.161 rows=11270 loops=1)
                                Buckets: 16384 (originally 4096)  Batches: 1 (originally 1)  Memory Usage: 657kB
                                Buffers: shared hit=20052
                                ->  Seq Scan on part  (cost=0.00..41051.46 rows=3184 width=13) (actual time=0.051..620.122 rows=11270 loops=1)
                                      Filter: (((p_brand1)::text >= 'MFGR#2221'::text) AND ((p_brand1)::text <= 'MFGR#2228'::text))
                                      Rows Removed by Filter: 1388730
                                      Buffers: shared hit=20052
                    ->  Hash  (cost=26461.59..26461.59 rows=198193 width=4) (actual time=127.785..127.785 rows=199889 loops=1)
                          Buckets: 262144  Batches: 1  Memory Usage: 9076kB
                          Buffers: shared hit=13962
                          ->  Seq Scan on supplier  (cost=0.00..26461.59 rows=198193 width=4) (actual time=0.016..102.187 rows=199889 loops=1)
                                Filter: ((s_region)::text = 'ASIA'::text)
                                Rows Removed by Filter: 800111
                                Buffers: shared hit=13962
              ->  Hash  (cost=63.33..63.33 rows=2533 width=8) (actual time=299.477..299.477 rows=2556 loops=1)
                    Buckets: 4096  Batches: 1  Memory Usage: 132kB
                    Buffers: shared hit=38
                    ->  Seq Scan on date  (cost=0.00..63.33 rows=2533 width=8) (actual time=298.936..299.241 rows=2556 loops=1)
                          Buffers: shared hit=38
Planning Time: 0.190 ms
JIT:
  Functions: 46
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 2.996 ms, Inlining 3.770 ms, Optimization 194.606 ms, Emission 100.255 ms, Total 301.627 ms
Execution Time: 88729.503 ms
```

高速化

JITによるオーバーヘッド

検証2：No.2-2の実行計画

- JIT:OFF

```
                                                                                 QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
GroupAggregate  (cost=16129050.94..16131822.99 rows=6713 width=21) (actual time=94027.557..94265.541 rows=56 loops=1)
  Group Key: date.d_year, part.p_brand1
  Buffers: shared hit=7792538
  ->  Sort  (cost=16129050.94..16129727.17 rows=270492 width=17) (actual time=94024.314..94150.612 rows=961749 loops=1)
        Sort Key: date.d_year, part.p_brand1
        Sort Method: quicksort  Memory: 99713kB
        Buffers: shared hit=7792538
        ->  Hash Join  (cost=70125.25..16104645.50 rows=270492 width=17) (actual time=779.793..93198.298 rows=961749 loops=1)
              Hash Cond: (lineorder.lo_orderdate = date.d_datekey)
              Buffers: shared hit=7792538
              ->  Hash Join  (cost=70030.26..16100831.24 rows=270492 width=17) (actual time=779.182..92970.569 rows=961749 loops=1)
                    Hash Cond: (lineorder.lo_suppkey = supplier.s_suppkey)
                    Buffers: shared hit=7792500
                    ->  Hash Join  (cost=41091.26..16064069.52 rows=1364746 width=21) (actual time=636.921..90822.476 rows=4813031 loops=1)
                          Hash Cond: (lineorder.lo_partkey = part.p_partkey)
                          Buffers: shared hit=7778538
                          ->  Seq Scan on lineorder  (cost=0.00..13759100.40 rows=600061440 width=16) (actual time=0.006..31115.221 rows=600037902 loops=1)
                                Buffers: shared hit=7758486
                          ->  Hash  (cost=41051.46..41051.46 rows=3184 width=13) (actual time=636.823..636.824 rows=11270 loops=1)
                                Buckets: 16384 (originally 4096)  Batches: 1 (originally 1)  Memory Usage: 657kB
                                Buffers: shared hit=20052
                                ->  Seq Scan on part  (cost=0.00..41051.46 rows=3184 width=13) (actual time=0.020..634.619 rows=11270 loops=1)
                                      Filter: (((p_brand1)::text >= 'MFGR#2221'::text) AND ((p_brand1)::text <= 'MFGR#2228'::text))
                                      Rows Removed by Filter: 1388730
                                      Buffers: shared hit=20052
                    ->  Hash  (cost=26461.59..26461.59 rows=198193 width=4) (actual time=141.636..141.636 rows=199889 loops=1)
                          Buckets: 262144  Batches: 1  Memory Usage: 9076kB
                          Buffers: shared hit=13962
                          ->  Seq Scan on supplier  (cost=0.00..26461.59 rows=198193 width=4) (actual time=0.010..116.053 rows=199889 loops=1)
                                Filter: ((s_region)::text = 'ASIA'::text)
                                Rows Removed by Filter: 800111
                                Buffers: shared hit=13962
              ->  Hash  (cost=63.33..63.33 rows=2533 width=8) (actual time=0.592..0.592 rows=2556 loops=1)
                    Buckets: 4096  Batches: 1  Memory Usage: 132kB
                    Buffers: shared hit=38
                    ->  Seq Scan on date  (cost=0.00..63.33 rows=2533 width=8) (actual time=0.008..0.335 rows=2556 loops=1)
                          Buffers: shared hit=38
Planning Time: 0.197 ms
Execution Time: 94271.519 ms
```

PGECons

PostgreSQL Enterprise Consortium