



PGECons
PostgreSQL Enterprise Consortium

2017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. パラレルクエリ性能検証
4. 全文検索検証
5. レプリケーション検証
6. FDW検証

■ 2017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

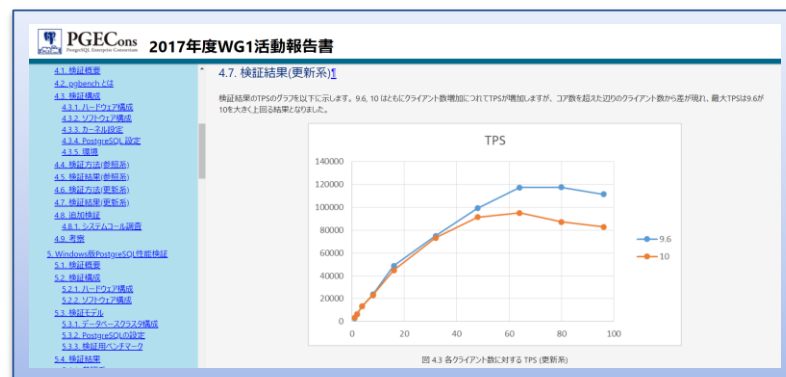
- postgresql.confやベンチマークプログラムのパラメータなど、性能改善のヒントとなるノウハウも豊富

□ 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
- 株式会社クリアコード
- 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

定点観測 / スケールアップ検証

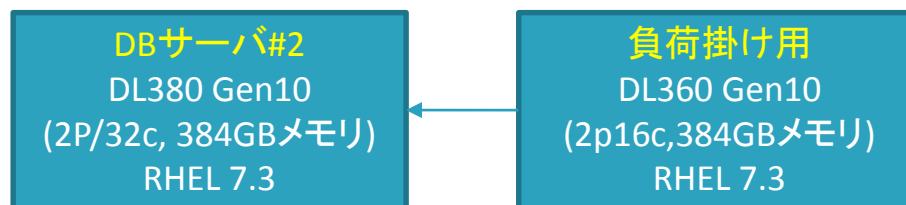
■ 目的

- メニーコアCPU上でのPostgreSQLのスケラビリティを検証
- 新バージョンのPostgreSQLの性能改善傾向を知る
 - PGECcons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
- 今年度はバージョン9.6とバージョン10の比較を実施

検証方法

■ pgbenchを用いて検証

- オンメモリでの走行
 - スケールファクタ2000(DBサイズは30GB)
 - 更新系はフィルファクタ80
 - 事前に pg_prewarm を実行
- カスタムスクリプト
 - 参照系、更新系それぞれで適度な負荷がかかるスクリプトを使用
- 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(1~96)させスループットを測定

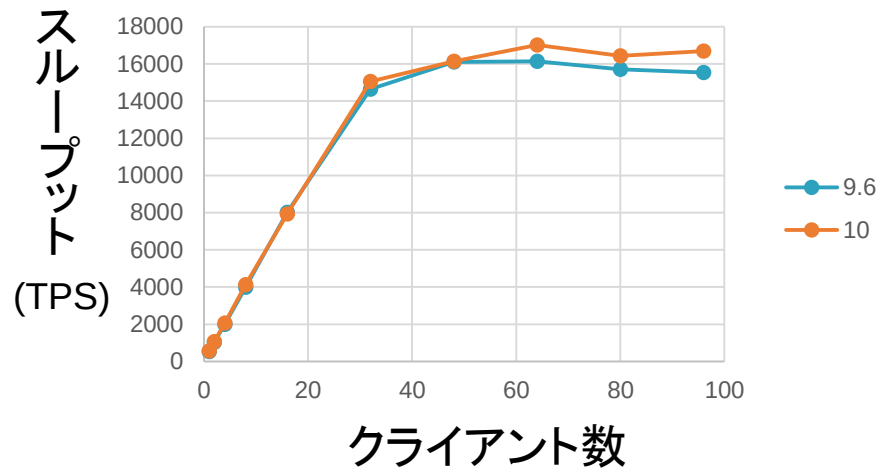


32コアのサーバ機を使用

測定結果

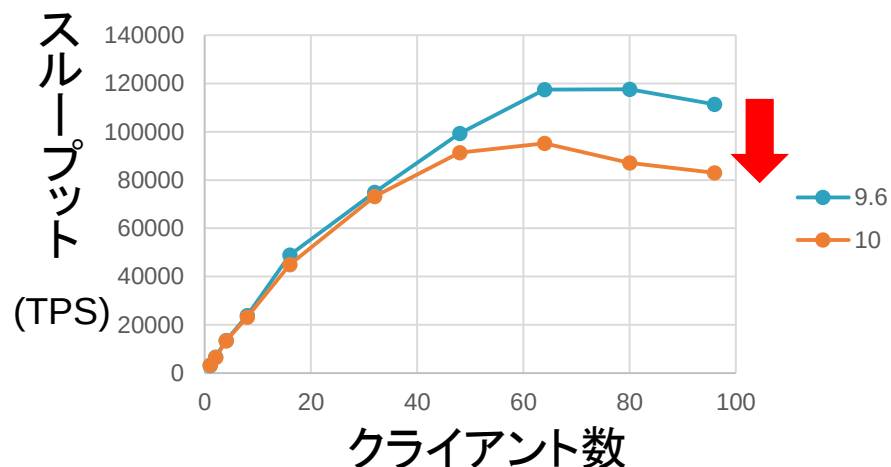
■ [参照系]

- 9.6と10では参照性能差は見られなかった
- ピーク値にわずかに差がある程度



■ [更新系]

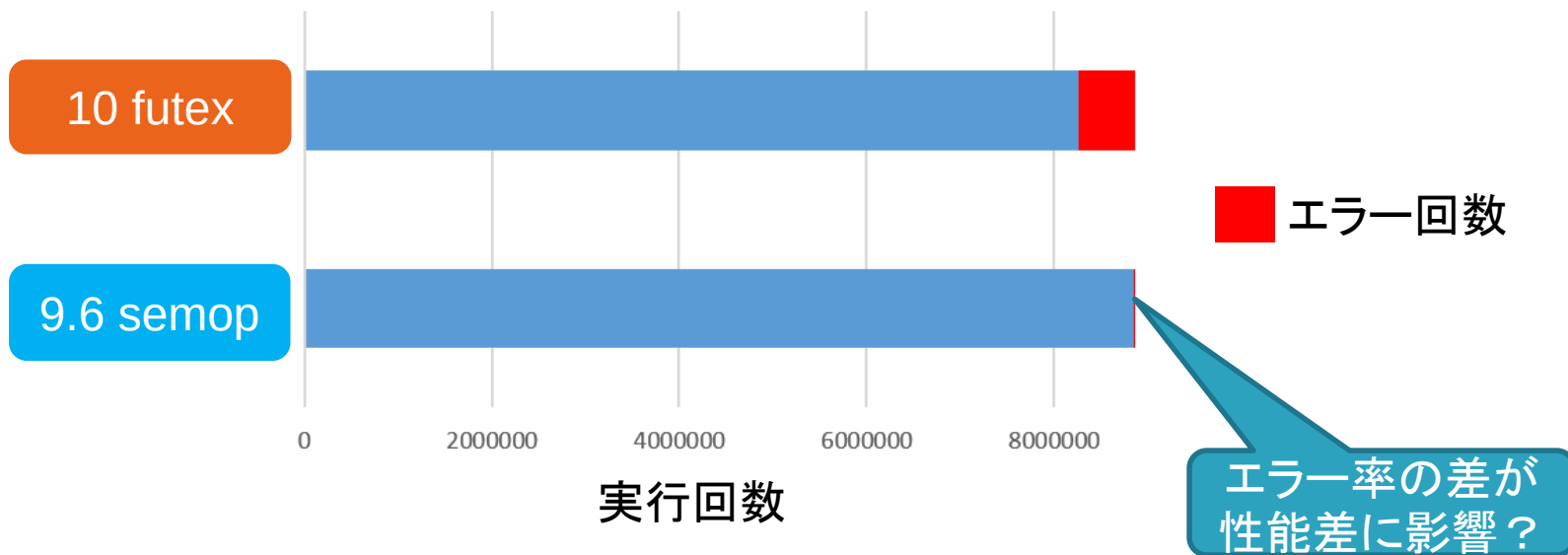
- 9.6に比べて10は明らかな更新性能低下が見られた
- 最大で26%の性能低下



原因調査の追加検証実施

[更新系] 追加検証

- 9.6と10で使用されているシステムコールの違いを確認 (straceコマンド)
 - 実行時間割合が一番高いシステムコールが異なっていた
→9.6: semop, 10: futex
 - 実行回数とエラー回数を取得すると、ほぼ同数の実行回数に対してエラー率の違いが見られる



[更新系] 追加検証

■ そもそも各システムコールはどんな関数から呼ばれているのか

```
$ strace ... -k -e semop  
semop(5472262, [{3, -1, 0}], 1) = 0
```

9.6

```
> /usr/lib64/libc-2.17.so(semop+0x7) [0xf8f37]  
> /usr/local/pgsql/9.6.6/bin/postgres(PGSemaphoreLock+0x41) [0x24f441]  
>...
```

```
semop(5505031, [{1, 1, 0}], 1) = 0  
> /usr/lib64/libc-2.17.so(semop+0x7) [0xf8f37]  
> /usr/local/pgsql/9.6.6/bin/postgres(PGSemaphoreUnlock+0x41) [0x24f4c1]  
>...
```

```
$ strace ... -k -e futex
```

10

```
futex(0x7f7f17ddfc98, FUTEX_WAIT, 0, NULL) = 0  
> /usr/lib64/libpthread-2.17.so(do_futex_wait.constprop.1+0x2b) [0xd79b]  
> /usr/lib64/libpthread-2.17.so(__new_sem_wait_slow.constprop.0+0x4f) [0xd82f]  
> /usr/lib64/libpthread-2.17.so(sem_wait+0x2b) [0xd8cb]  
> /usr/local/pgsql/10.1/bin/postgres(PGSemaphoreLock+0x22) [0x284e92]  
>...  
futex(0x7f7f17ddfe58, FUTEX_WAKE, 1) = 1  
> /usr/lib64/libpthread-2.17.so(sem_post+0x3d) [0xdb7d]  
> /usr/local/pgsql/10.1/bin/postgres(PGSemaphoreUnlock+0x22) [0x284ef2]  
>...
```

□ いずれも同じセマフォ操作の関数から呼ばれている

■ PGSemaphoreLock, PGSemaphoreUnLock

□ ここで、10のリリースノートをおさらいすると次の変更点が

■ “Use POSIX semaphores rather than SysV semaphores on Linux and FreeBSD” 「Linux と FreeBSD において SysVセマフォではなく POSIXセマフォを使用するようになりました。」

まとめ

■ [参照系]

- 9.6と比較し、10は特に大きな性能向上は見られなかった

■ [更新系]

- 9.6と比較し、10では大きな性能**低下**が見られた
- 実行時間が最も長いシステムコールが異なり、実行回数はほぼ同等だが、エラー率の差があった
- 異なるシステムコールはいずれも同じセマフォ操作の関数から呼び出されていた
- 10の変更点「Linux 上で利用されるセマフォが System V セマフォから POSIX セマフォに変更された」ことが更新性能に影響していたと推測される



Windows検証 成果紹介

Windows検証

■ 検証理由

- Windows版PostgreSQLの利用者は増えているが、Windows版PostgreSQLの性能に関する検証を実施していなかった。

■ 検証概要

- 同一モデルでの性能測定を行い、Windows版PostgreSQLとLinux版のPostgreSQLとの性能傾向を比較する

■ 検証環境

- 定点観測で用いた構成を使用する
- DBサーバのOSとして、Windows OSを用いる

■ 検証方法

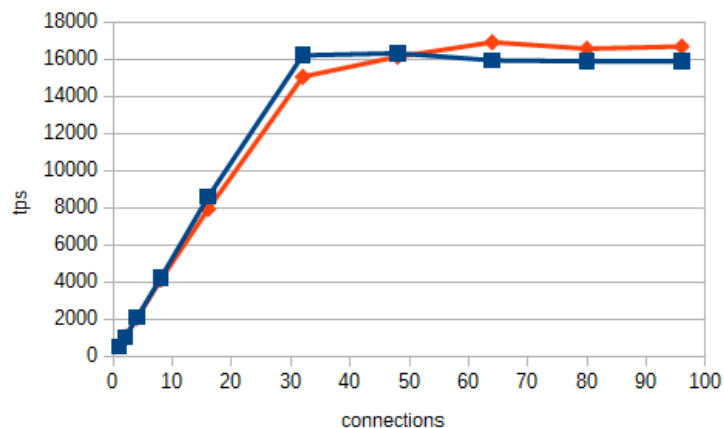
- 定点観測で実施していた測定モデルをWindows版PostgreSQL上に構築し、性能を測定した
(詳細は「定点観測」参照)

検証結果

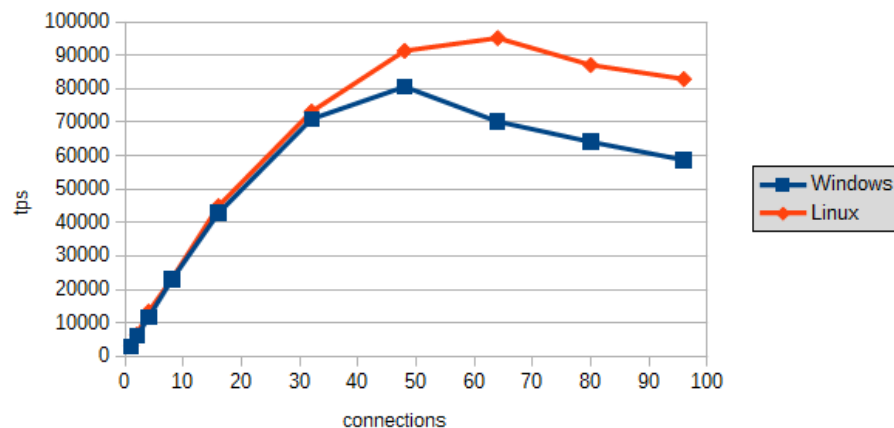
■ 今回の検証の要約を以下に示す

1. 参照モデルの場合はWindows版/Linux版での性能差はなかった
2. 更新モデルの場合も高負荷でなければWindows版/Linux版での性能差はなかった
3. 書き込みの負荷が非常に高い場合は、Windows版で性能低下の可能性はある

Winows/Linux PostgreSQL 10.1 Readonly tps



Winows/Linux PostgreSQL 10 Write tps



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で強化された

■ 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、2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（デフォルト無効）を有効化し、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
- パラレルクエリによらない改善の効果を確認するため、パラレルクエリを無効化した状態でも測定

検証方法

- TPC-DSに定義されているクエリ 99種類/103本のクエリのうち、以下の2種類/2本を使用する
 - No.10
 - count () が6列分、group byとorder byの対象8列
 - 合計7テーブルの結合処理
 - exists句が入れ子状に3か所出現
 - No.11
 - 2つのサブクエリの結果のunion all
 - 両サブクエリともに3テーブルの結合処理、sum () が1列分、group byの対象8列
 - 上記結果を3回自己結合
- 上記2種類/2本は事前検証の結果、下記の理由で選定
 - No.10 : パラレルクエリ実行を行うことで劇的な性能改善が見られた
 - No.11 : v10でのみパラレルクエリ実行が選択され、v10で新規対応したパターンにマッチしたものと判断した
- 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 2GB とする

検証環境

■ 環境1

- 定点観測における、DBサーバと同一

■ 環境2

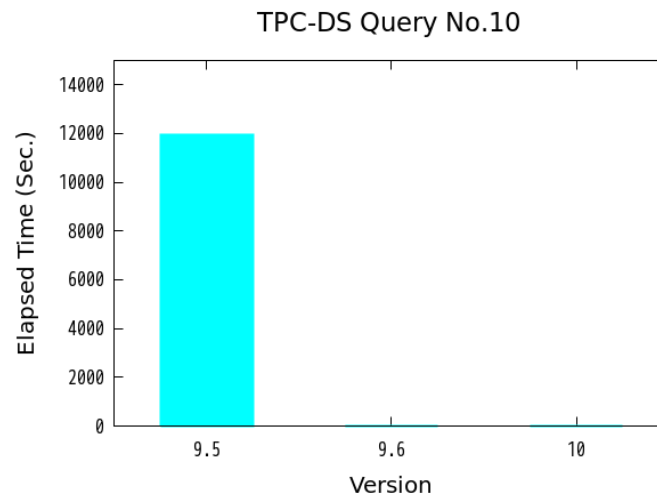
機種	NEC Express5800/R120f-1E
CPU	インテル Xeon E5-2630 v4 @ 2.20GHz
メモリ	256GB
ストレージ	SSD 400GB × 8 (RAID 5)
OS	Red Hat Enterprise Linux 7.2

■ 時間の都合上、ここでは環境2での結果のみを紹介

検証A パラレルクエリでの処理性能の改善状況

■ No.10

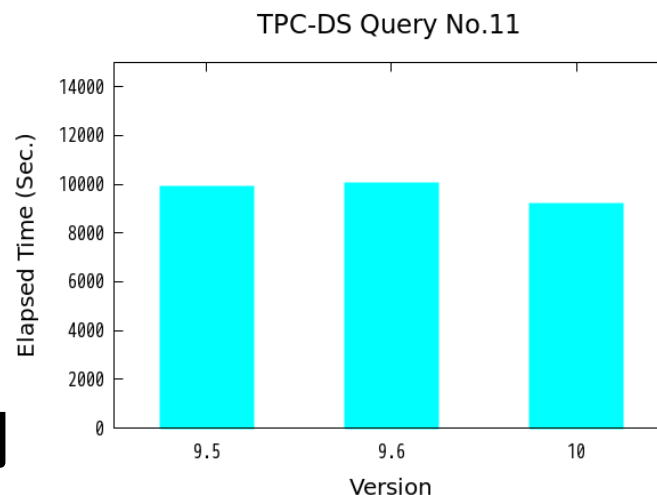
- パラレルクエリの有効化によって
劇的 (2000倍以上) に高速化
 - ただし、並列処理で速くなったのではなく、実行計画のうち、実際に実行された部分が変わった結果、高速化。



■ No.11

- v9.6ではパラレルクエリを選択せず
- v10はパラレルクエリを選択し、
8%程度の高速化
 - 「非相関サブクエリのパラレル実行」
(v10で新規対応) に該当したか？

- パラレルクエリ無効の状態でも、
v10がv9.5比で、両クエリともに
20%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

■ No.10

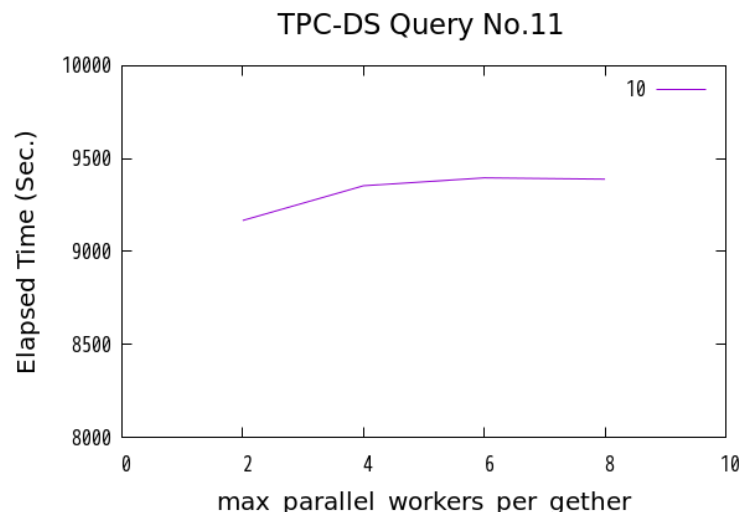
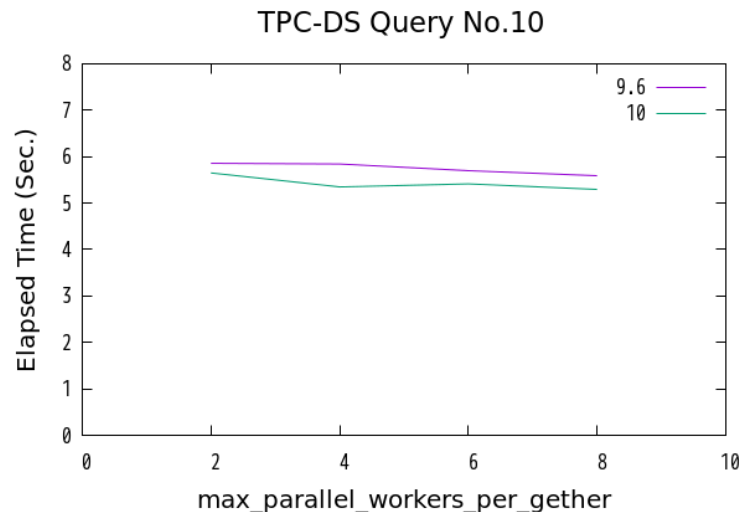
- 並列処理数の増加にともなって、全体処理時間が減少する傾向

■ No.11

- 並列処理数の増加にともなって、全体処理時間が増加する傾向

- ただし、パラレルクエリの処理時間は、両クエリともに、全体の処理時間を左右するような割合にはなっていない。

- No.10で全体の20%弱、No.11で全体の0.25%以下
- パラレルクエリ部分のみでは、並列処理数の増加にともなって、処理時間が減少する傾向が認められた。
- 上記の全体処理時間の傾向は、パラレルクエリ外の要因が大きい。



まとめ

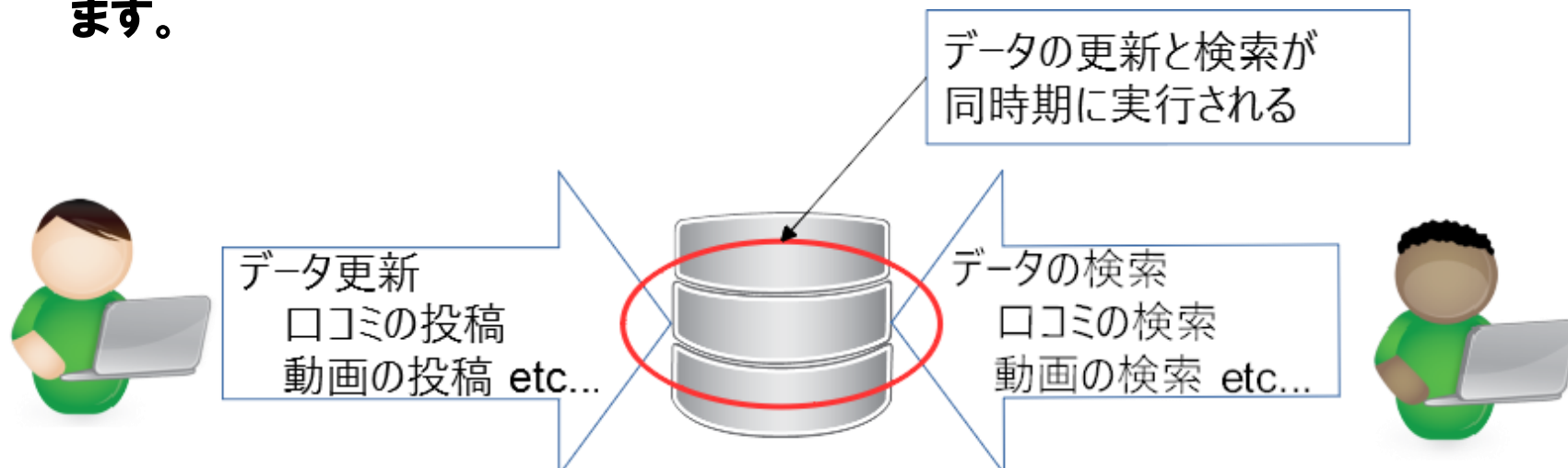
- **パラレルクエリの使用には細心の注意が必要**
 - **パラレルクエリ実行により、実行計画の取得だけでは判明しない差異が存在**
 - **EXPLAINではわからない。EXPLAIN ANALYZEが必要。**
 - **パラレルクエリの効果が十分に得られるか、確認が必要**
 - **パラレルクエリ実行の処理時間と、オーバヘッドやパラレルクエリ部分以外の処理時間の割合を算出する、など**
- **現状のパラレルクエリの実装においては、OLAP用途での使用には不安が残る**
- **より不安なくパラレルクエリを使用できるよう、今後のパラレルクエリのさらなる進化、プランナ・エグゼキュータのさらなる深化に期待**



全文検索 成果紹介

背景・概要

- 近年の全文検索のユースケースでは、静的なデータに対する検索だけではなく、継続的に更新されているデータに対して検索を行う事が増えてきています。



- 上記のようなケースでは、更新と検索を1つのデータベースで担うため、静的データに対する検索と比較してパフォーマンスが劣化すると考えられます。
- そこで、本検証では、PostgreSQL 10から導入されたロジカルレプリケーションを用いた負荷分散の方法を検討し、動的に変化するデータに対する検索でもパフォーマンスを落とさずにデータの更新・検索が可能かを検証しました。

検証方法

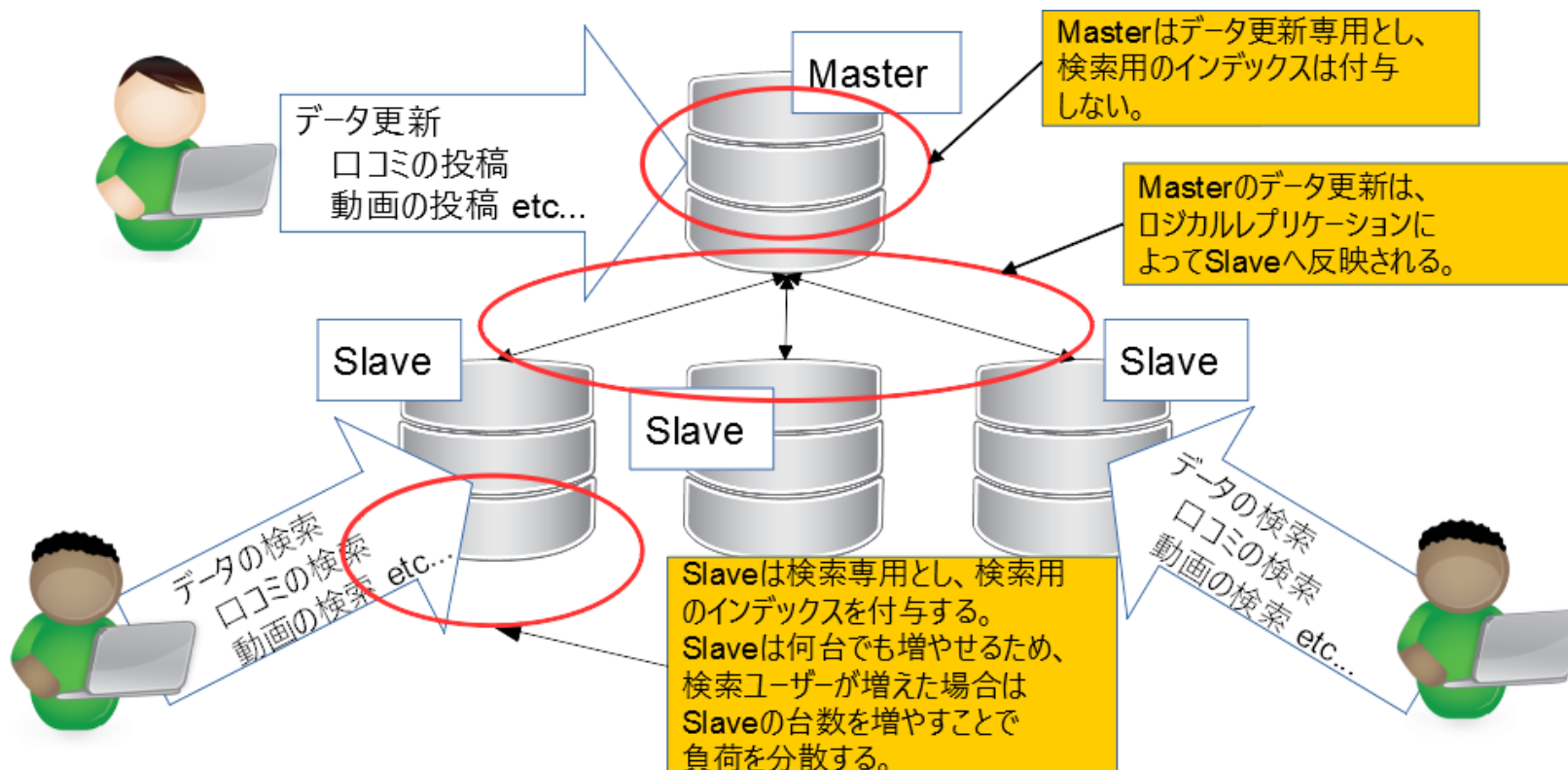
- 本検証では、特に国内においてニーズの高い「日本語検索」に対応した、以下の全文検索モジュールを使用しました。
 1. pg_bigm (ピージー・バイグラム)
 2. PGroonga (ピージールンガ)
- 検索対象は、国立情報学研究所のダウンロードサービスにより株式会社ドワンゴから提供を受けた「ニコニコ動画コメント等データ」内の動画のメタデータをPostgreSQLへjsonb型として格納した以下のようなデータを使用しています。
 - 今回は、以下のデータから、よく検索対象になる、動画の概要 (discription) に対して検索を行いました。

id	record (jsonb型)
1	{"tags": ["料理", "カレー"], "title": "テスト", "length": 578, "category": "料理", "size_low": 29606343, "video_id": "smXXXXXXXX", "file_type": "flv", "size_high": 31095758, "watch_num": XXXX, "mylist_num": 22, "comment_num": 290, "description": "テスト", "upload_time": XXXXXXXXXXXXX}
...	...

検証構成

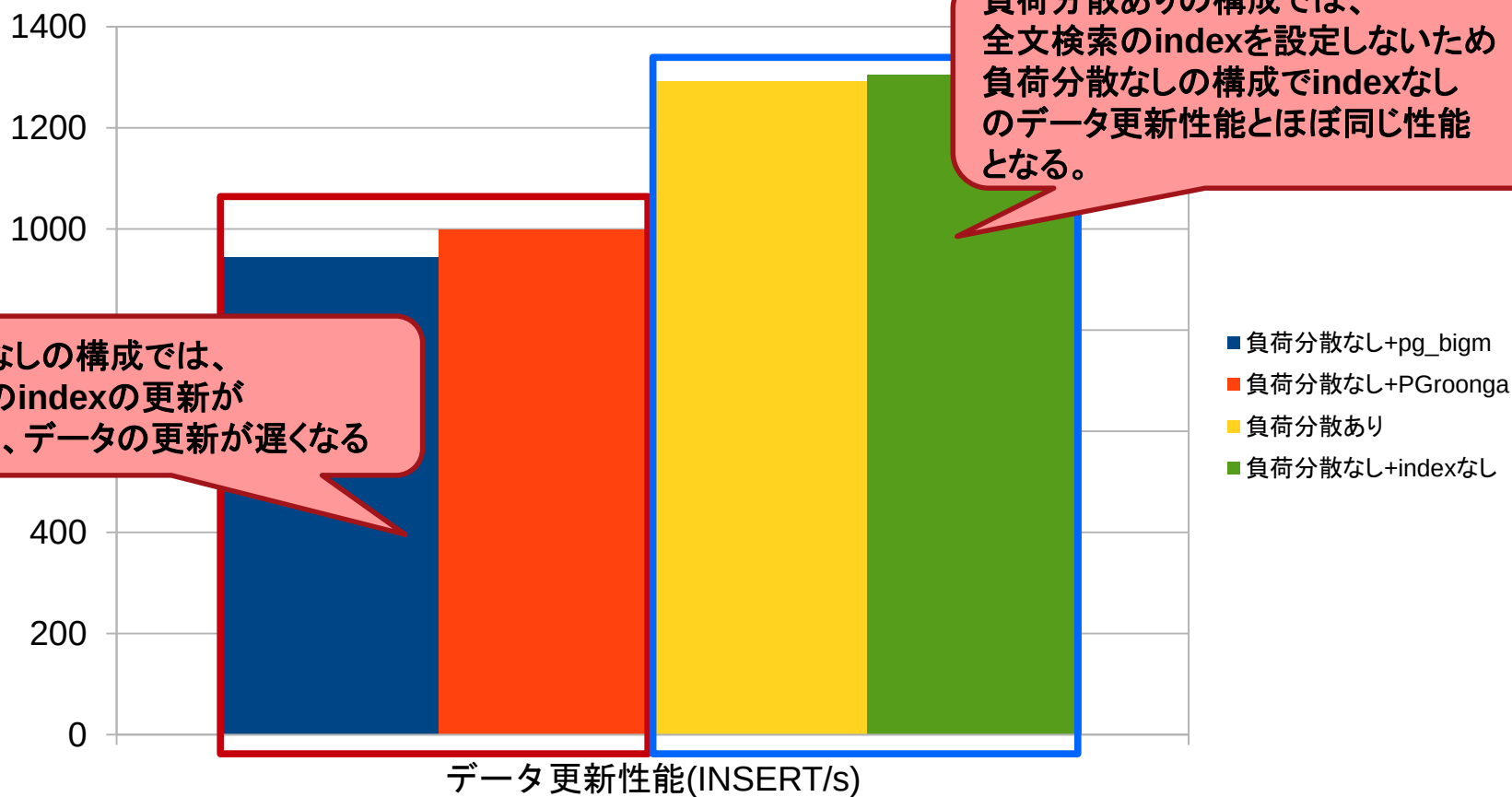
- ロジカルレプリケーション（非同期）を用いた負荷分散の構成は以下の通りです。以下の構成と、単一のデータベースを用いた構成とでデータの更新性能と検索性能を測定しました。

□ ロジカルレプリケーションを用いた負荷分散構成



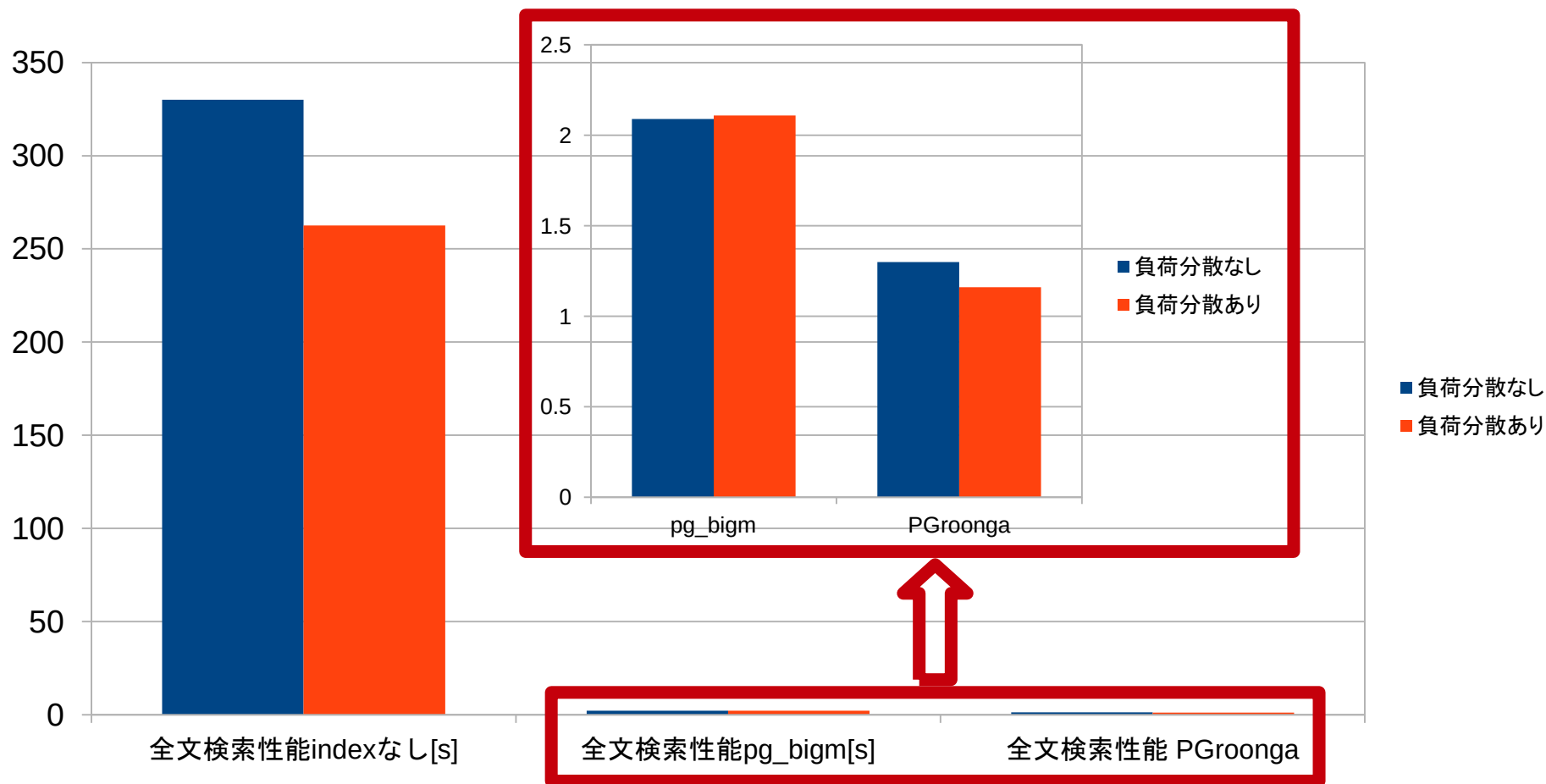
測定結果：データ更新性能の比較

- 負荷分散構成あり、なしでのデータ更新性能を比較しました。
- データ更新性能とは、1秒間あたりにINSERTしたレコード数で、数値が大きいほど、高速にデータを更新していることになります。



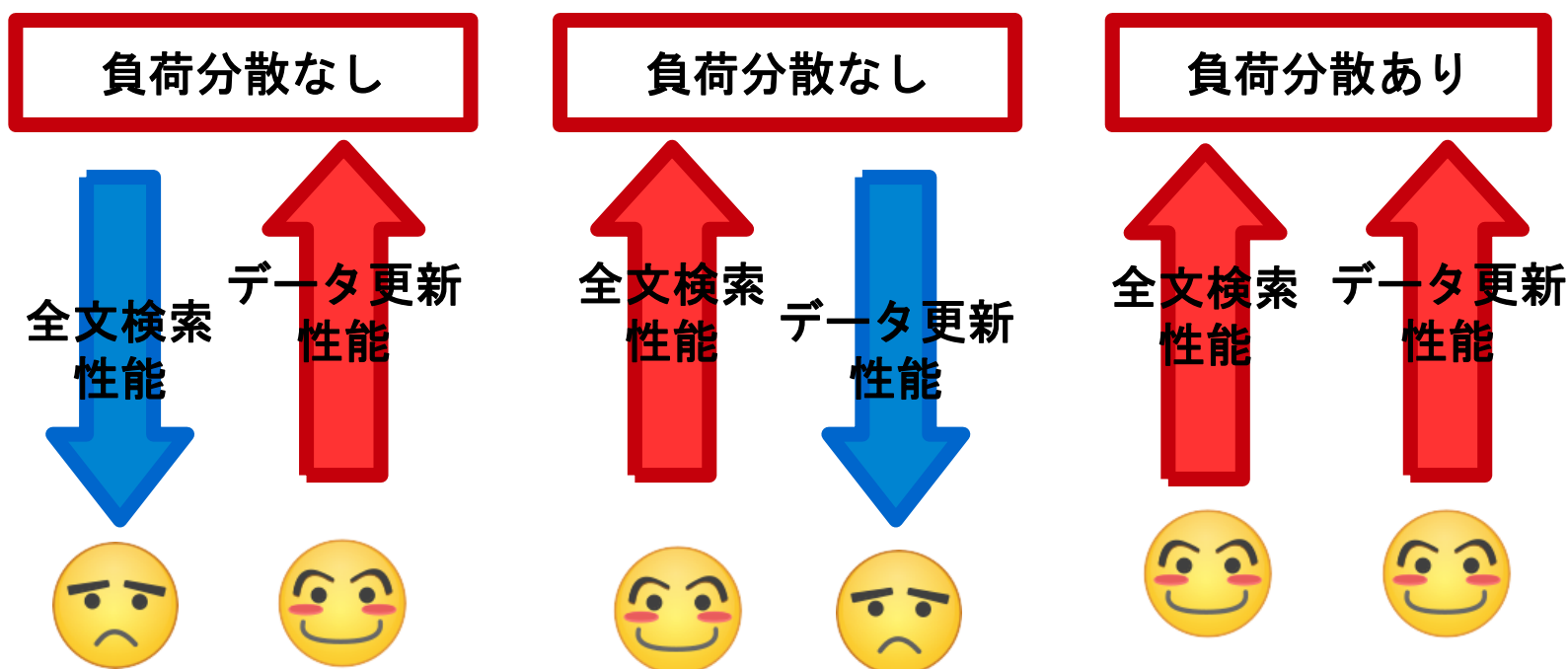
測定結果：全文検索性能の比較

- 負荷分散構成あり、なしでの全文検索性能を比較しました。
- 全文検索性能とは、平均レイテンシで、数値が小さいほど、高速に検索していることになります。



結果まとめ

- 継続的にデータ更新が行われている状態で全文検索を行うケースで、ロジカルレプリケーションを用いた負荷分散構成を検討し、負荷分散の効果を検証しました。
- 結果は以下の通り、負荷分散なしでは、検索か更新どちらか性能が落ちますが、負荷分散ありの構成ではどちらの性能も落とさず高速な検索と更新が可能となります。



レプリケーション検証 成果紹介

レプリケーション検証

■ 概要

- PostgreSQLでは9.0からレプリケーション機能を実装しており、9.6からは複数台への同期レプリケーションが利用可能となった
- 複数台のレプリケーション構成にて懸念される、Slave台数増加に伴う性能低下について、同期・非同期、それらが混在する場合で検証する

■ 測定条件/方法

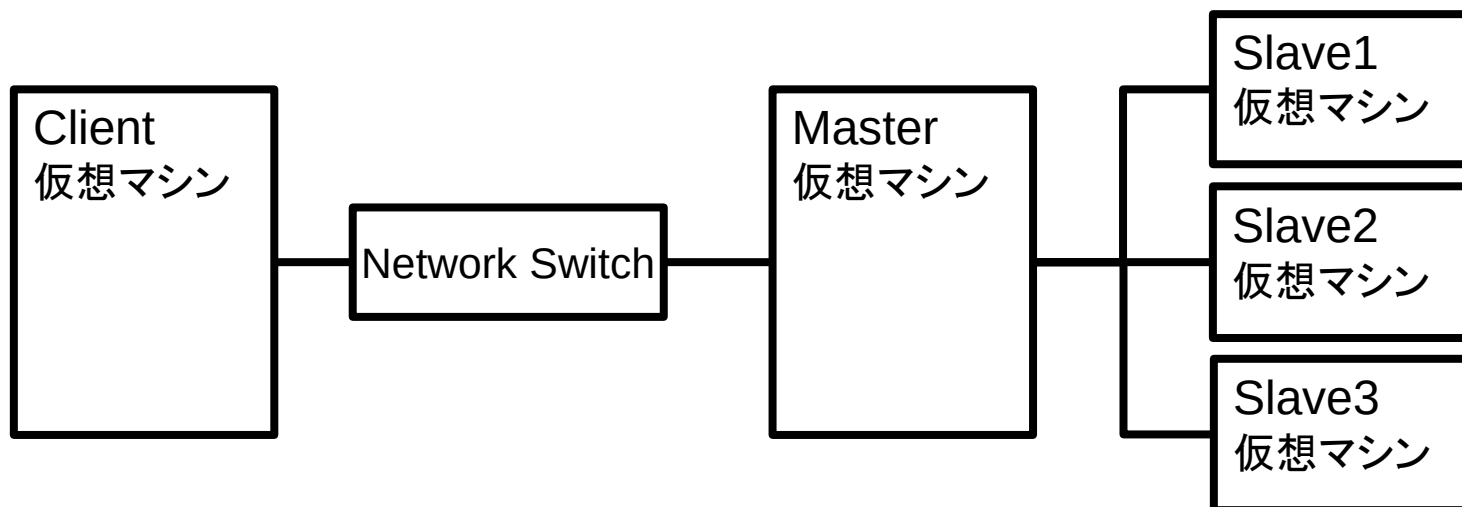
- 以下2つのパターンで測定を実施
 - 検証A:同期のみ／非同期のみ、1～3台の構成
 - 検証B:同期・非同期混在、台数を3台に固定した構成
- pgbenchを使用しデータベースの作成・TPS測定を行う
 - スケールファクタは1,000（データベースのサイズ:約15GB）
 - 更新系のカスタムスクリプトを使用し、MasterでのTPSを測定

マシン構成

■ 測定に使用したマシンのスペック・構成図は以下

- データベース・クライアント共に仮想マシンを利用
- CentOS 7.3、PostgreSQL 10.1 を使用

	vCPU	メモリ	ディスク
Client	4	48GB	200GB
Master	4	48GB	200GB
Slave(1～3)	4	48GB	200GB



結果

■ 検証A(同期のみ／非同期のみの構成)

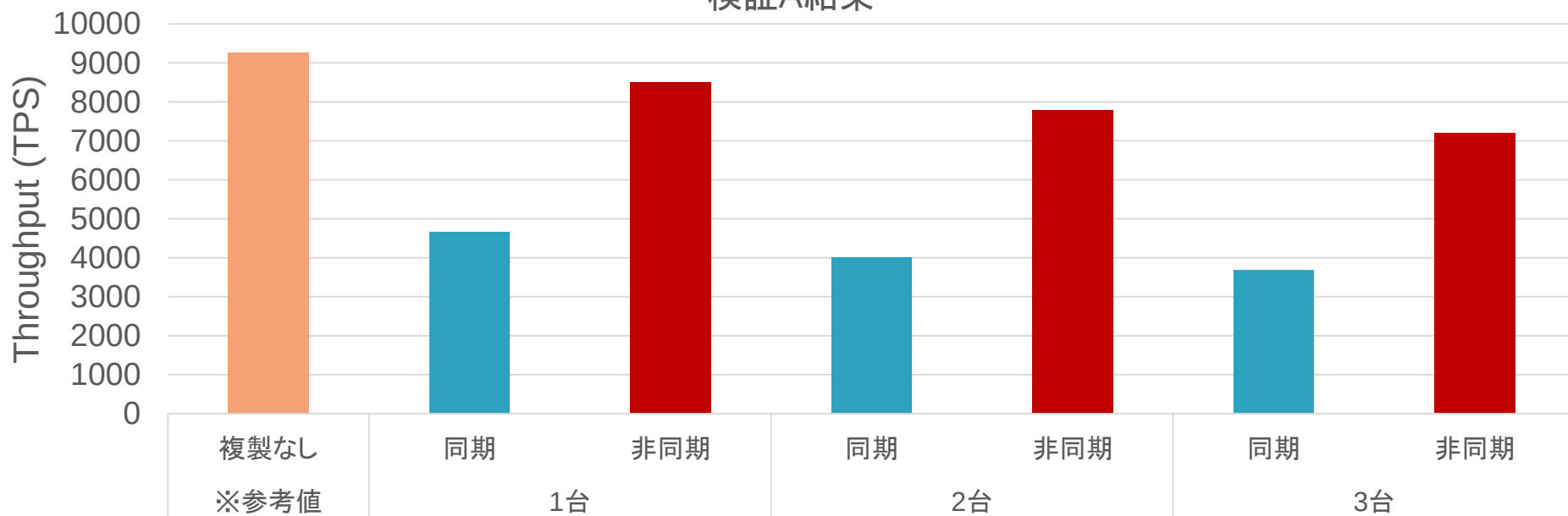
□ 同期は非同期と比べ性能が約1/2に

- 同期の場合は、Master・Slaveの2台分の処理完了を待つ必要があるためと考えられる

□ 同期・非同期ともに台数増加に伴う性能低下有り

- 台数が1台増えるたびに同期・非同期ともに約10%性能低下
- 4台目以降も同様の低下率で低下していくと考えられる

検証A結果



結果

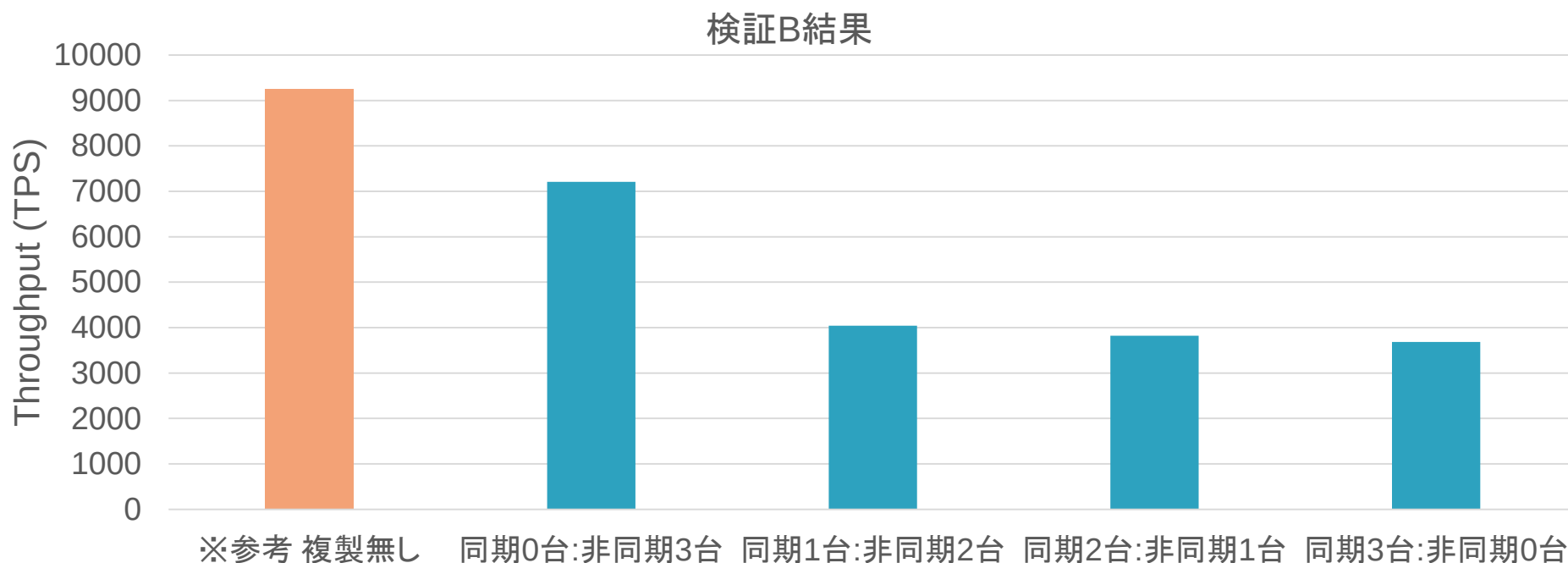
■ 検証B(同期・非同期混在の構成)

□ 同期台数の増加に伴い低能が低下

- 非同期のみの構成と比較すると、同期を1台でも追加すると性能値が急落していることが確認できる

□ 非同期を同期に1台変更することにより、TPSが約5%ずつ低下している

- 検証Aと同様に、一定の割合で性能が逓減していくことが考えられる



まとめ

■ 今回の検証では以下の傾向が見られた

1. 同期のみの構成は、非同期のみの構成と比べ約1/2の性能となる
2. 同期・非同期混在の構成では、総台数が同じでも同期の台数を増やすことで性能の低下が見られる
3. 同期・非同期の台数を変化させることで、ある程度一定の割合で性能値が変化していく

■ 今回の構成では、MasterとSlaveが同一ホスト上で稼働

- Slaveの数が多くなるほど、ネットワークやストレージといった物理的なリソースの競合により性能に影響が出てきたのではないかと考えられる
- レプリケーションの際は、サーバの配置やマシン構成の考慮も必要

FDW検証 成果紹介

FDW検証

■ 検証概要

- 外部データへアクセスする仕組みのFDW(Foreign Data Wrapper)は、PostgreSQLが近年力を入れて強化している機能の一つ
- PostgreSQL 9.3より実装されているpostgres_fdwを使用し、FDWの性能的なオーバーヘッドを、参照・更新・挿入それぞれで検証する

■ 検証方法(ユースケース)

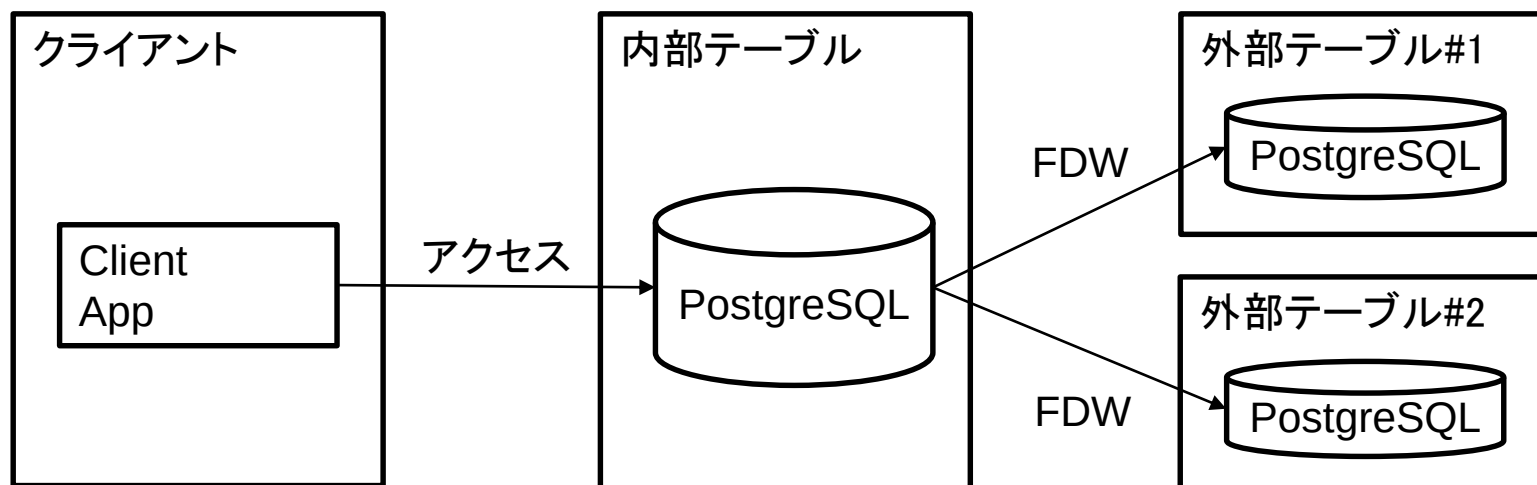
- 簡単な銀行の口座処理を模擬し、顧客情報、口座情報、取引履歴の3つのテーブルを作成して以下の処理を行う
 - 参照:顧客情報と残高を参照する
 - 更新:ある顧客の口座残高を増やす
 - 挿入:取引履歴に新しい履歴を挿入する
- それぞれのテーブルが内部・外部にあるパターンを想定し、性能値を比較することでオーバーヘッドを明らかにする

マシン構成

■ 測定に使用したマシンのスペック・構成図は以下

□ CentOS 7.4、PostgreSQL 10.3を使用

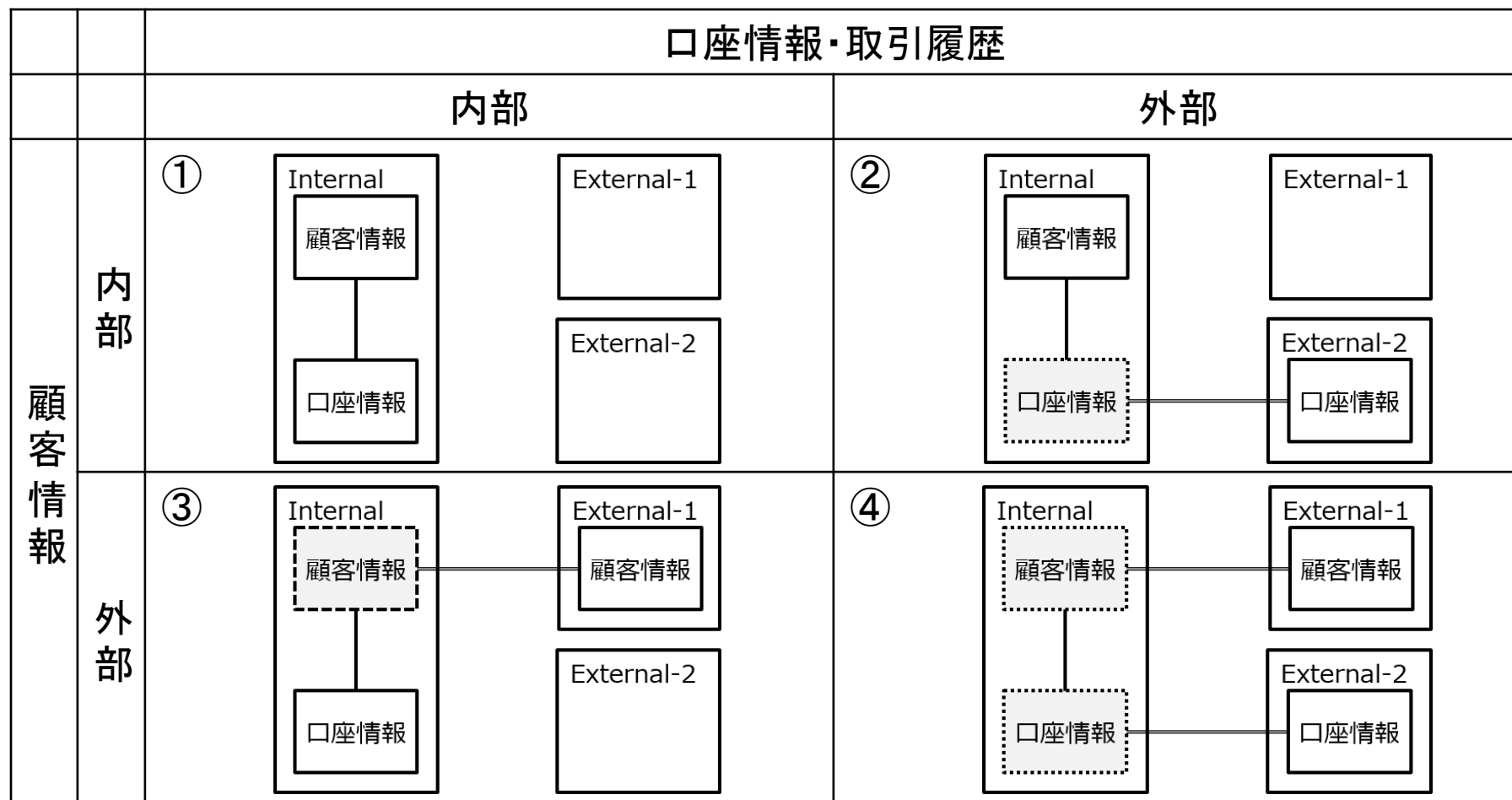
	種別	CPU	メモリ	ディスク
内部テーブル	仮想	2.40GHz 2Cores	4GB	20GB
外部テーブル#1	仮想	2.40GHz 2Cores	4GB	20GB
外部テーブル#2	仮想	2.40GHz 2Cores	4GB	20GB
クライアント	物理	Intel(R) Xeon(R) E5620 2.40GHz 16cores	24GB	



検証パターン

■ 外部テーブルの有無による4パターンを比較

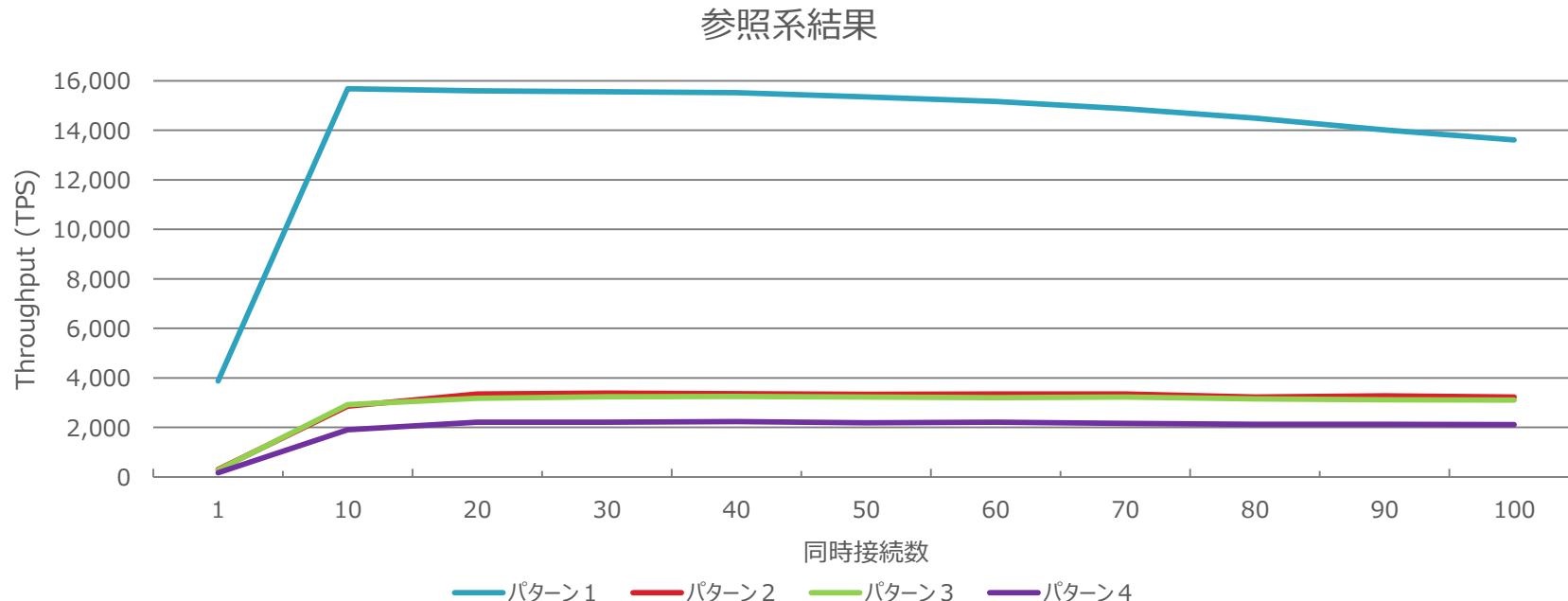
□ 参照ではパターン①～④、更新・挿入ではパターン①と②を比較



結果

■ 参照系

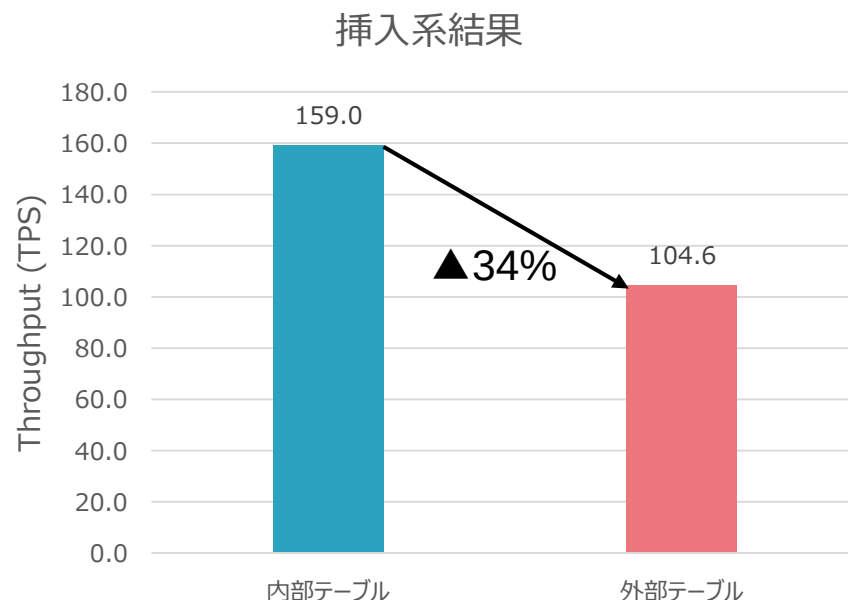
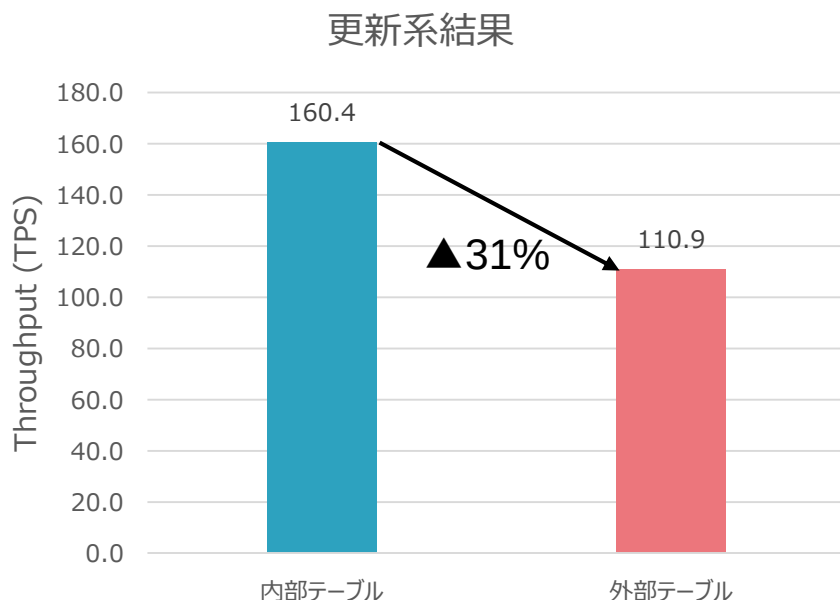
- 外部テーブルありの場合(パターン②～④)は、内部テーブルのみの場合(パターン①)と比較し性能が約80%～90%劣化
- 単純なアクセスパターンであったため、内部テーブルのみのレスポンス時間が非常に短く、外部テーブルへのアクセスによるオーバーヘッドの影響が大きく出たと考えられる



結果

■ 更新系・挿入系

- 外部テーブルへの更新・挿入の場合、内部テーブルの更新・挿入に比べて約30%の性能劣化が見られた
- 参照系と同様に、外部テーブルへのアクセスによるオーバヘッドの影響が出ていると考えられるが、参照系より劣化率は低くなっている



まとめ

■ 参照・更新・挿入のすべての処理で性能の劣化が見られた

- 参照系においては、約80%～90%と大幅に性能が劣化している
 - 個々のアクセスのレスポンス時間は、内部テーブルのみで平均14ms、
全て外部テーブルの場合でも平均70ms
- 更新・挿入においては、約30%の性能劣化
 - レスポンス時間は全パターンで10ms程度

⇒ 外部テーブルへのアクセスのオーバーヘッドが大きく出る結果となった

■ 今後の課題

- 内部テーブルのみであっても時間のかかる処理(テーブル結合など)を対象として測定することで、外部テーブルの処理によるオーバーヘッドをより明らかにできるのではないかと考える

2017年度活動をふりかえって

2017年度活動をふりかえって

- PostgreSQL 10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
- 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！



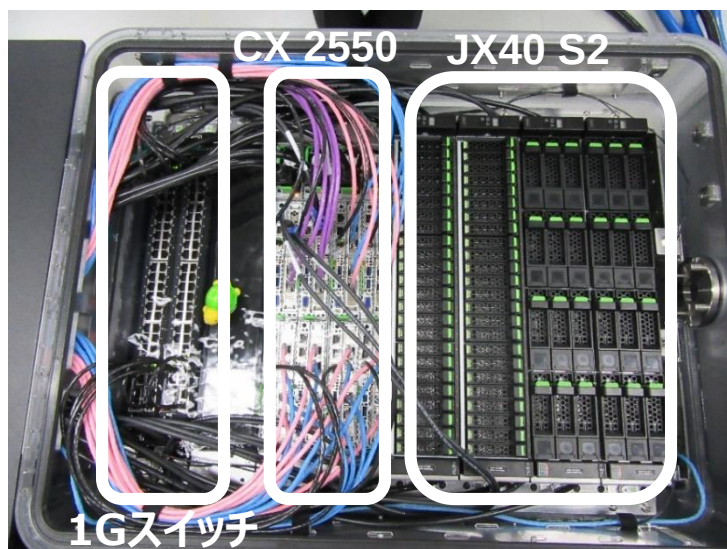
Appendix

レプリケーション検証, FDW検証環境 (富士通株式会社様提供)

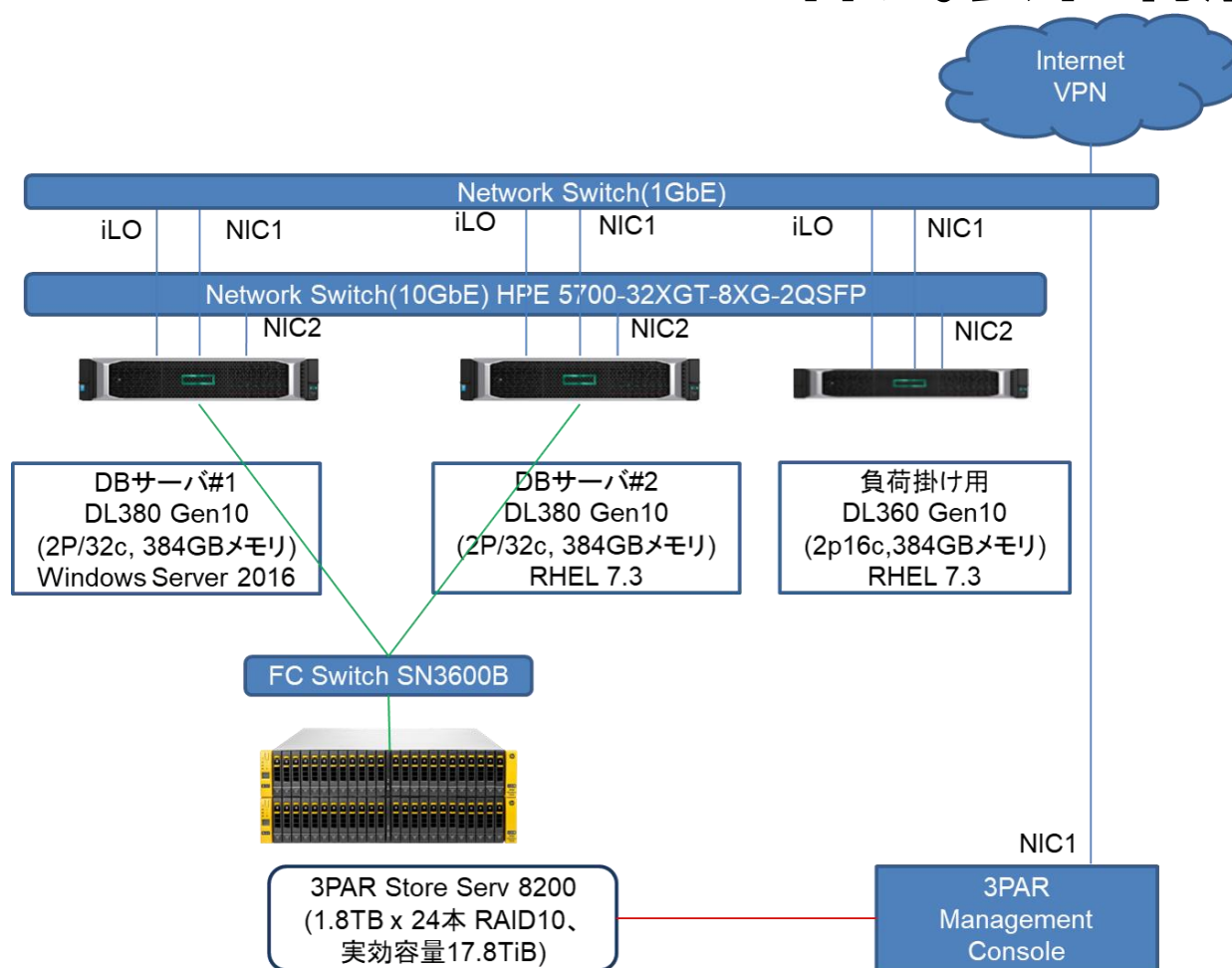
■ 液浸サーバ

- サーバなどのIT機器を丸ごと液体の冷媒に浸す「液浸冷却技術」使用
普通のサーバ／ストレージなどIT機器をそのまま浸すことが可能です

機種	FUJITSU Server PRIMERGY CX2550 M2
CPU	インテル Xeon E5-2650 v4@2.2GHz(48Core)
メモリ	256GB
ストレージ	2.5インチSSD 400GB×24(キャビネット: JX40 S2)
OS	CentOS 7.3



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・パッカー株式会社様提供)



パラレルクエリ検証環境 (日本電気株式会社様提供)

型番	Express5800/R120f-1E
CPU	インテル Xeon E5-2630 v4@2.20GHz
メモリ	256GB
内蔵ストレージ	SSD 400GB x 8 (RAID 5構成)
OS	Red Hat Enterprise Linux 7.2

全文検索検証環境

機種	Amazon Web Service(AWS) m4.2xlarge
CPU	Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz(8コア)
メモリ	32GB
内蔵ストレージ	SSD 256GB



PGECons

PostgreSQL Enterprise Consortium