PGECons
PostgreSQL Enterprise Consortium

大規模DBと仮想環境を見据えたPostgreSQLの性能検証

2014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. 仮想化環境(KVM)検証
  4. コンテナ環境(Docker)検証
  5. 2014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
  - 仮想環境でのストレージ構成について

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
  - 具体的な課題解決に向けて活動テーマを選定
  - 2014年度は「大規模DB」に加えて多様化した稼動環境の「仮想環境」の観点からも活動テーマを選定
- 活動スタイル
  - 毎年のPostgreSQLの新バージョンリリースにあわせて調査を実施。
  - 実機検証（約1か月間）でデータを収集して結果を検討
    - 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
    - メンバによるレビューで品質を担保

WG1の活動テーマ（赤字は2014年度活動テーマ）

大規模DBと仮想環境を見据えたPostgreSQL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

※「定点観測」と呼び毎年実施

■ 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（※）
  - ●多同時接続検証（※）
  - ●WAL書き込み改善
- 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
  - ●パーティショニングの有効ケース
  - ●多パーティションでのオーバヘッド
- ストレージ高速化
  - ●SSDを有効活用するデータ配置
- 多様な環境適用
  - ●仮想化（KVM）検証
  - ●Docker検証

2014年度WG1活動テーマ(PostgreSQL9.4)

- 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(PostgreSQL9.2/9.3)

- **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**

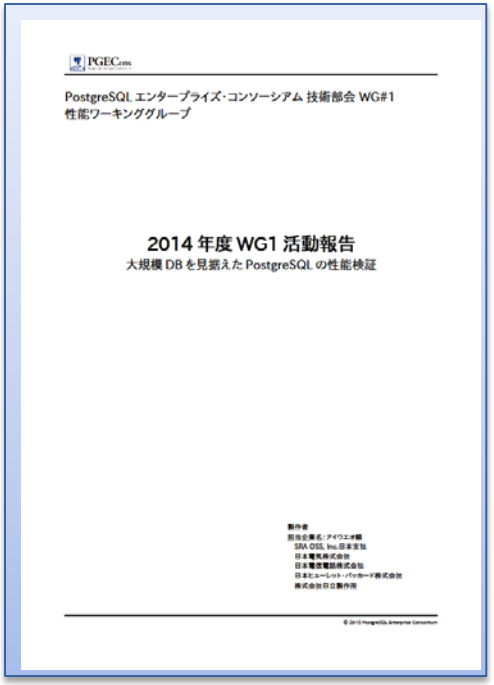
  パーティションテーブルへの投入・検索・メンテナンス性能検証

- **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

2014 年度 WG1 活動報告
大規模 DB を見据えた PostgreSQL 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
- 日本ヒューレット・パッカード株式会社 (主査)
- 株式会社日立製作所

(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]

定点観測(スケールアップ検証)概要

■2012年度から参照系ベンチマークを継続的に実施

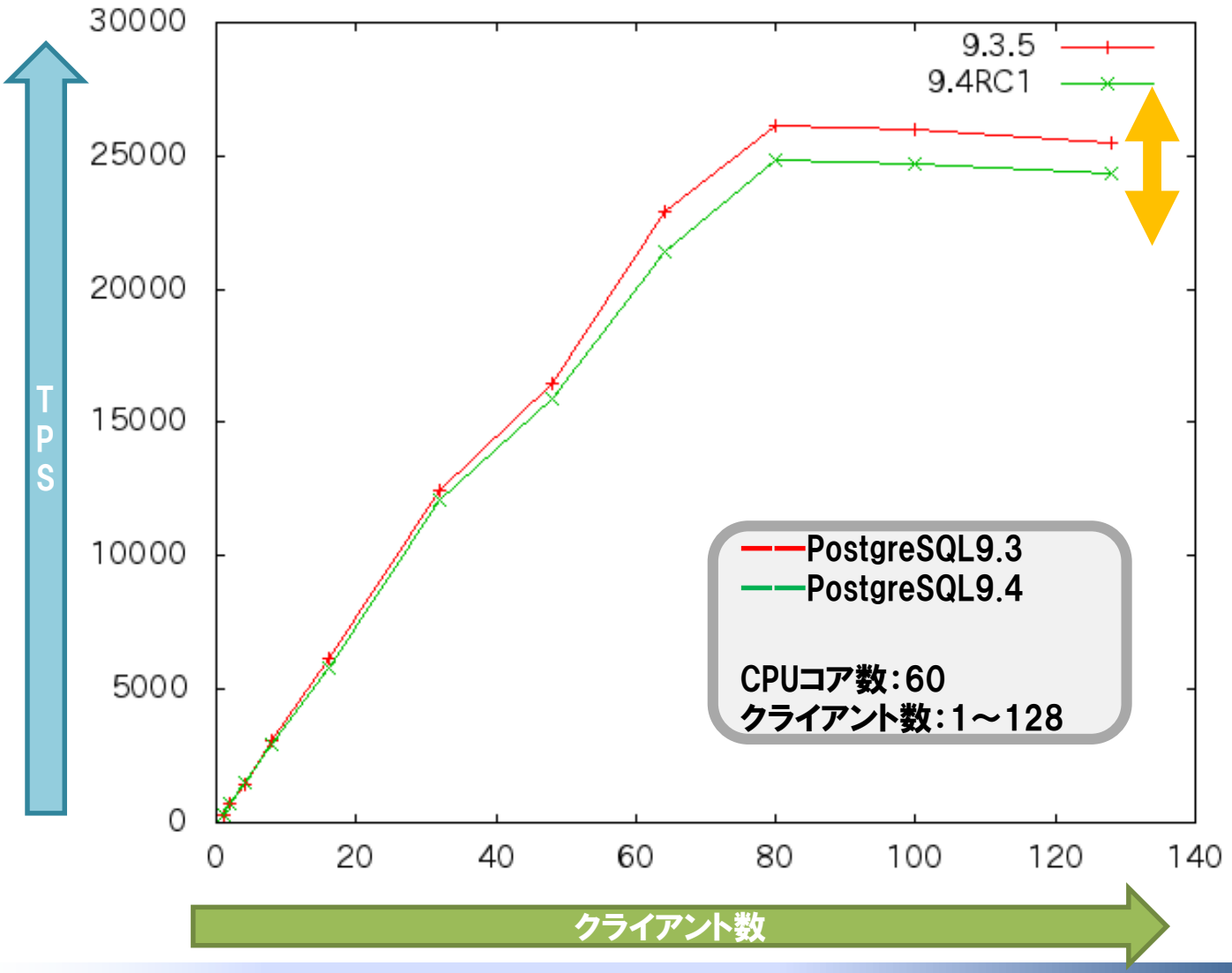
■計測内容は以下2項目

- 9.3と9.4の参照性能の比較
  9.3と9.4で同傾向・同程度の参照性能が出るか？
- 9.4のpage checksum機能有無での参照性能の比較
  page checksum機能のオーバーヘッドは？(2013年度の検証の再試)

■pgbenchで計測

- スケールファクタ1,000で初期化(DBサイズは15GB)
- ランダムに10,000行取得するカスタムスクリプトを300秒ずつ実行(pgbench -S では十分な負荷がかからないため)
- 3回の計測結果の中央値を結果とした

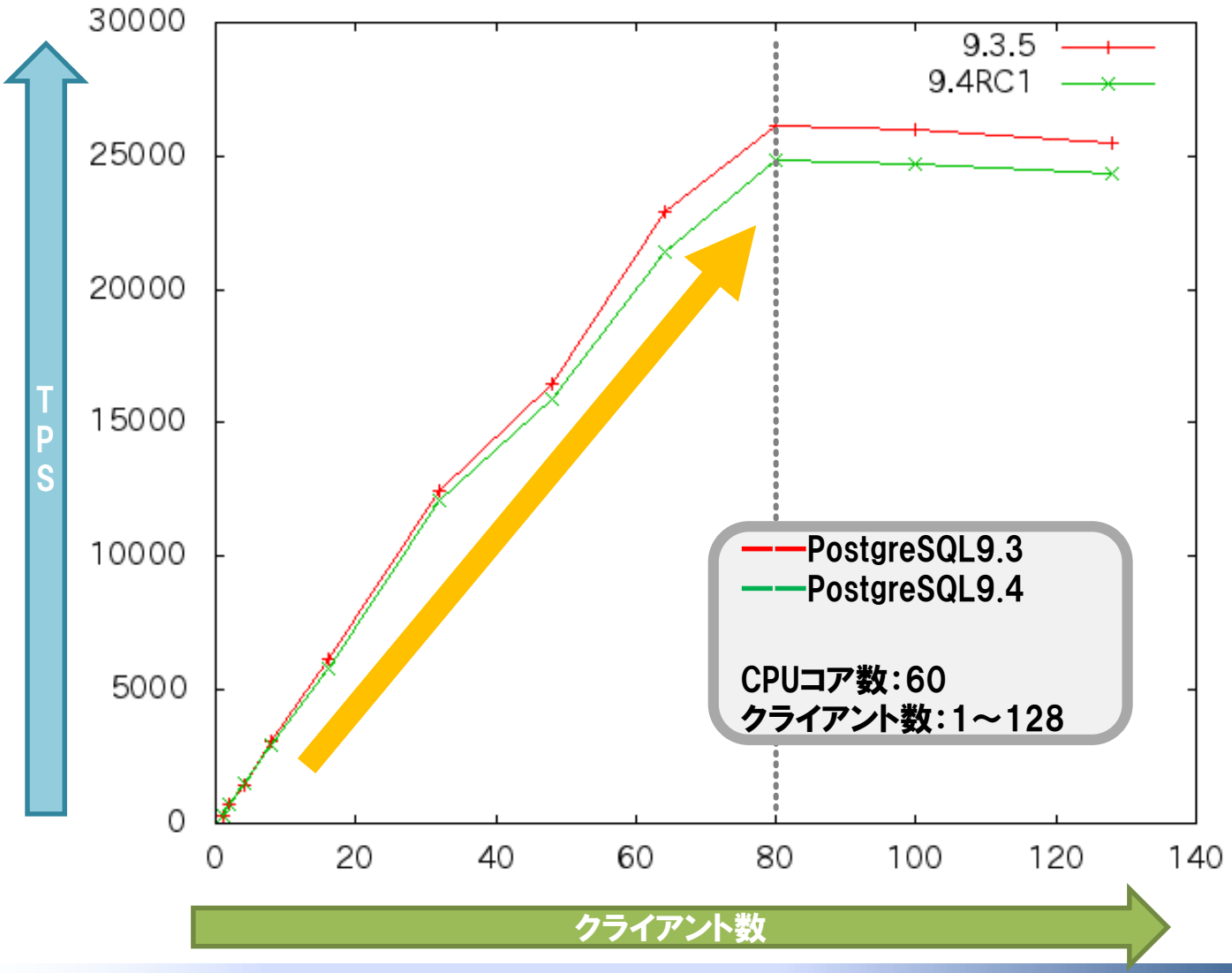
9.3 と 9.4 の参照性能の比較 <TPS値>



9.3 より 9.4 の方がわずかにTPSが低く、2～7%程度低下

9.4で行われた何らかの変更が影響？

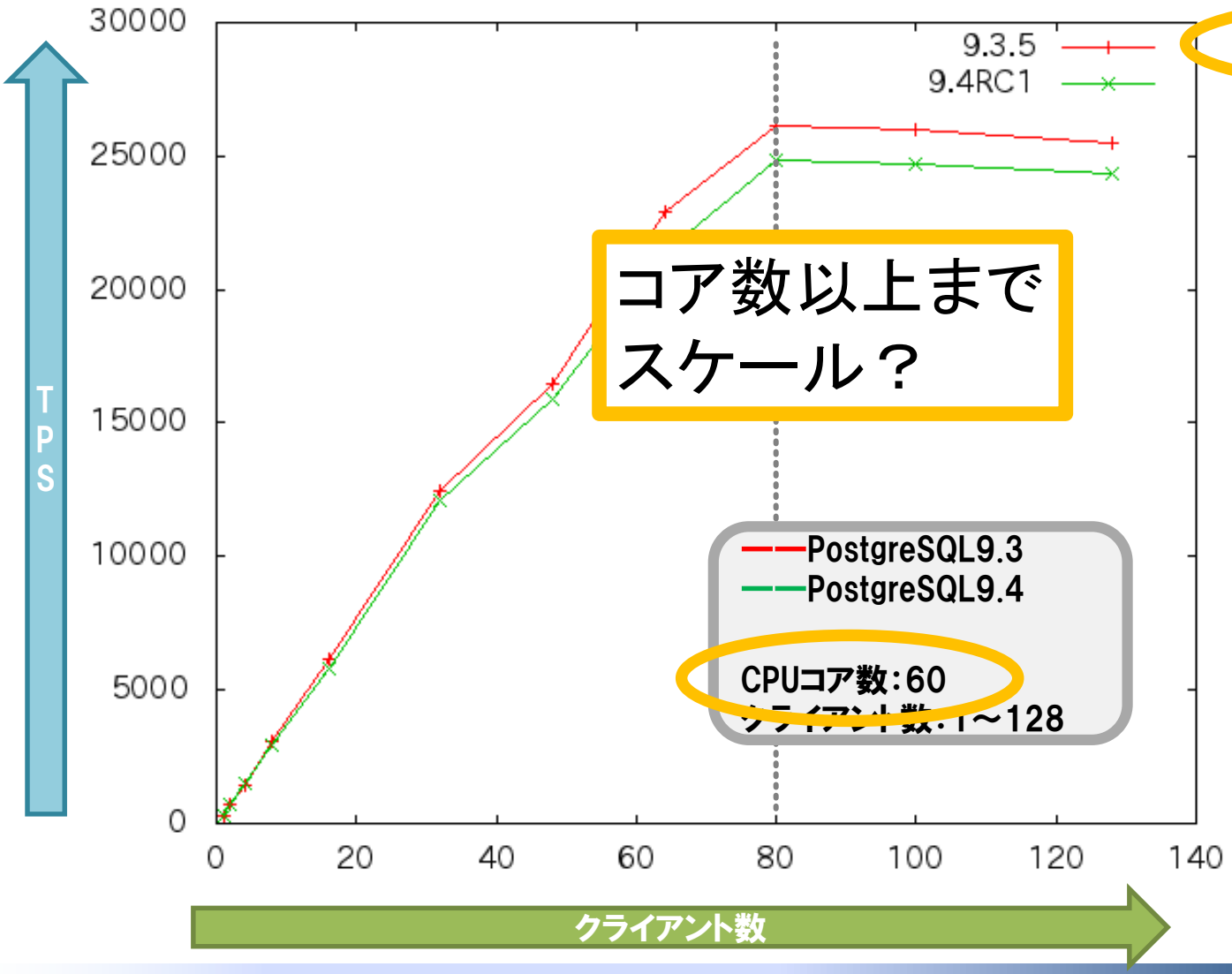
9.3 と 9.4 の参照性能の比較 <スケーラビリティ>



80 クライアントまで、ほぼリニアにスケール

9.4 と 9.3で同傾向のスケーラビリティ特性を持っていることを確認

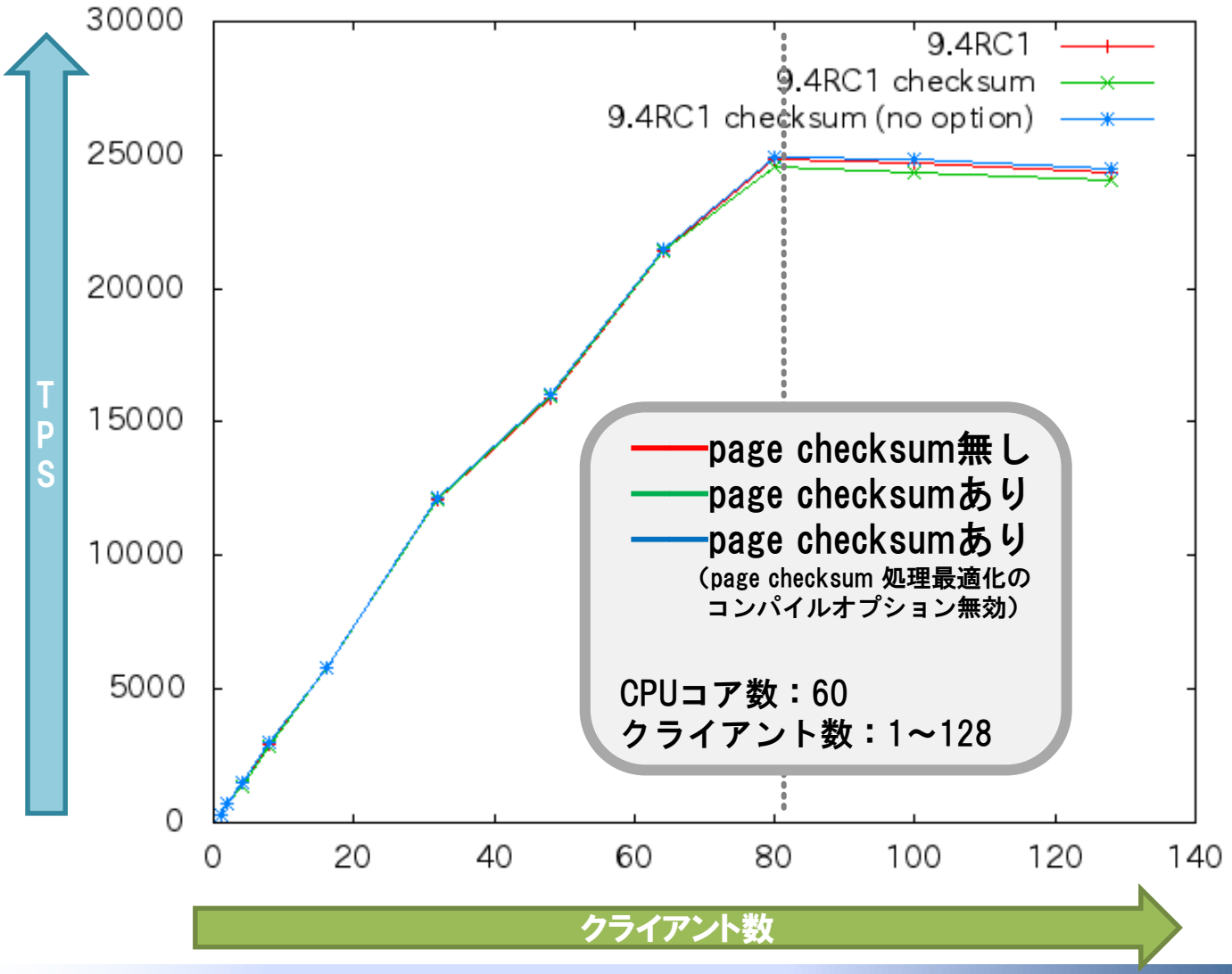
9.3 と 9.4 の参照性能の比較 <スケーラビリティ>



80 クライアントまで、ほぼリニアにスケール

クエリの負荷が不十分もしくはネットワークレイテンシの影響で、サーバのCPUを使い切れていないのが原因と考えられる

9.4 page checksum有無での参照性能の比較



全条件でTPSにほとんど差がなかった

9.4ではpage checksum が参照性能に与える影響は無視できるほど小さい

活動報告2

WAL改善/
スケールアップ検証 [更新系]

WAL改善検証の背景・目的

- 2012年度、2013年度は参照系のスケーラビリティを確認してきたが、更新系は未実施であった

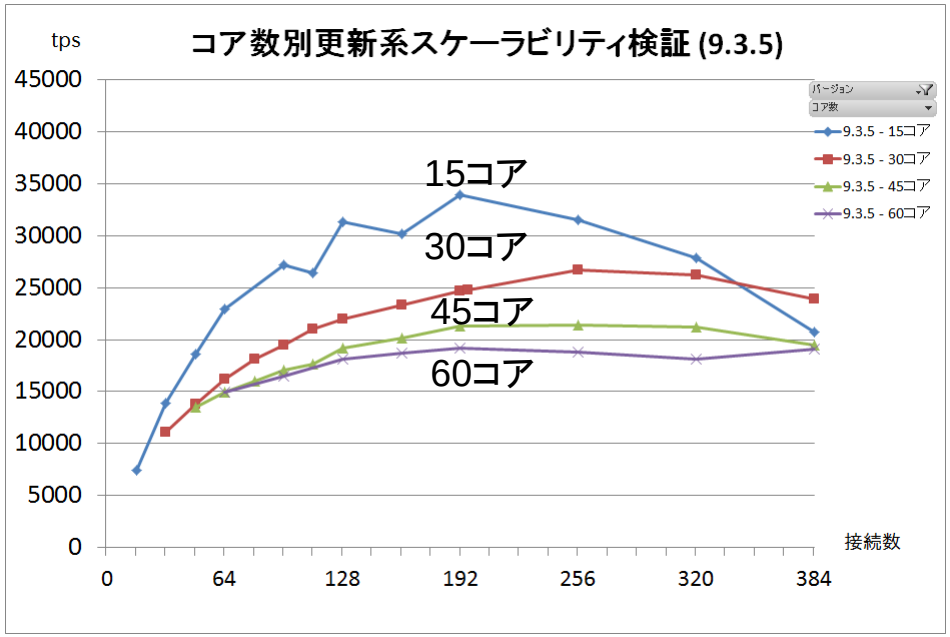
- PostgreSQL9.4では2つの更新系性能改善が実装された
  - WAL（Write Ahead Log）データ圧縮によるファイル出力量の軽減
  - WAL出力のロック競合の軽減による並列書込み性能の改善

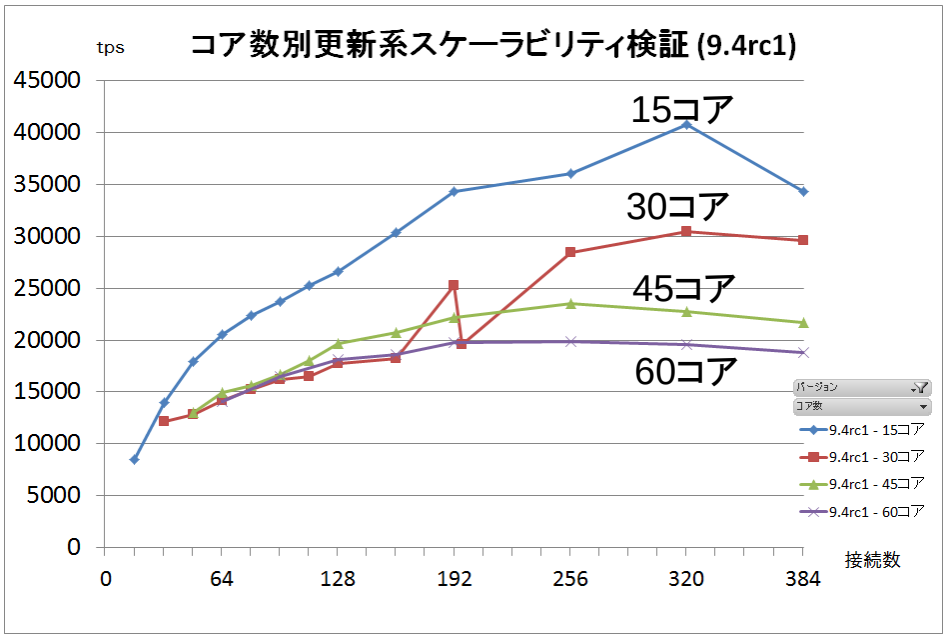
- これら性能改善を機に、目的を以下の2つを確認することに設定
  1. PostgreSQLの更新系のCPUスケーラビリティの達成状況
  2. PostgreSQL9.3.5 とPostgreSQL9.4RC1 を比較し、更新性能の改善状況

測定条件・環境

- マシンは後述の検証環境を使用
- 測定ツールはpgbenchを利用し、SF=7,000（約100GB）でDBを作成
  - 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
- 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
- サーバ側のコア数（15,30,45,60コア）とpgbenchの接続数を変更して測定
- 測定中にCHECKPOINTは発生させないように設定
- 3回の計測結果の中央値を結果とする

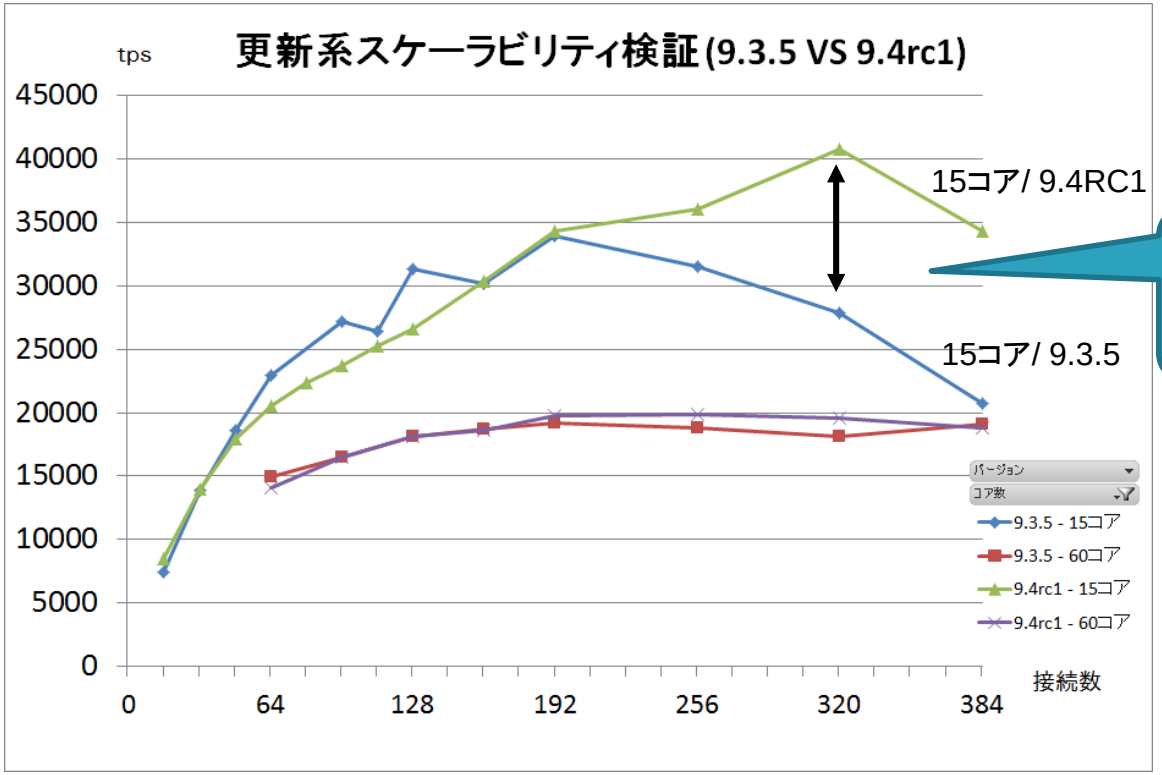
9.3.5/9.4RC1コア数別スケールアップ検証結果





- 参照系とは異なりI/Oが発生し、コア数よりも多くの接続数でピーク値となる
  - 例： 9.3.5/15コア：192接続　　9.4RC1/ 15コア 320接続
- PostgreSQL9.3.5、9.4RC1共に検証の中で最小の15コアが最大性能に
- 全体の傾向としてコア数が増加すると、スループットが低下している
  - 9.3.5では負荷が高いところで15コアと30コアが逆転しているところも
  - 9.4RC1では負荷が低いところで30コアが45コアよりも下がっている、原因は不明

15コア/60コアの9.3.5 VS 9.4RC1



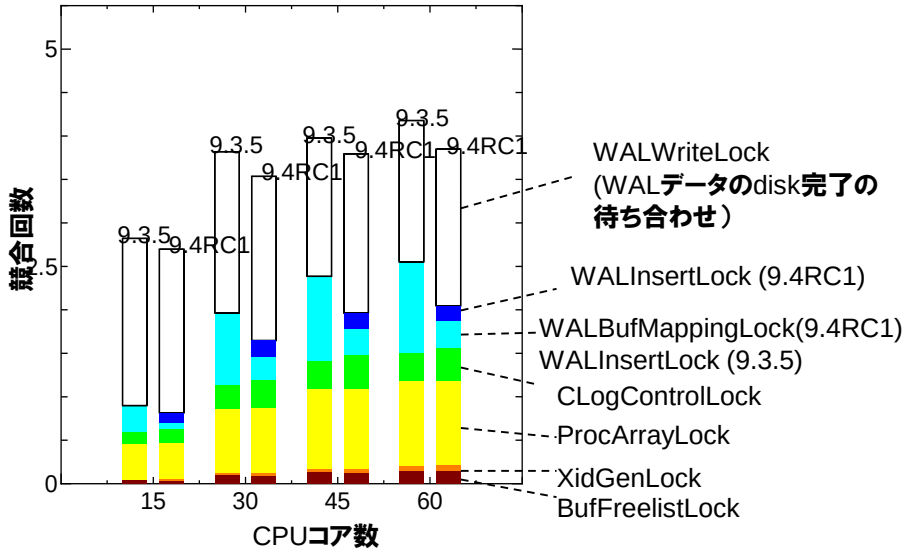
9.3.5比で9.4RC1が最大46%性能改善している

- コア数が少ない(15コア)場合
  - 負荷(接続数)が高いところでは9.4RC1が高い性能が出ている
  - ピーク値も9.4RC1のほうが高い負荷に対応できている
- コア数が多い(60コア)場合
  - 9.3.5と9.4RC1の傾向に差が出ていない

測定結果の分析

- 9.4RC1の改善について
  - ピーク時のストレージやCPU負荷がボトルネックでない可能性が高くロックの動作状況を確認
  - WALInsertLockの負荷が実質的に下がっていることが有効であったと想定

1トランザクションあたりのPostgreSQL9.4RC1と9.3.5の実質的なロック競合想定回数



- CPUスケーラビリティについて
  - 同様にピーク時のストレージI/O、CPU負荷のボトルネックは確認できず
  - コア数増加に伴い、ProcArrayLockを中心にロック競合の回数が増加しており、性能が伸びない一因であると推定

詳細は、2014年度WG1活動報告をご参照ください

結果まとめ

- （今回では特に15コアで）9.4RC1が9.3.5と比較して更新性能が改善されることが確認できた
- 15/30/45/60コアの測定単位では、単純な更新のみのワークロードにおいて9.3.5、9.4RC1共にCPUスケーラビリティについて確認できなかった
  - 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

- 来年度以降の課題：計測時のコア数粒度をもっと少なくし、コア数のピーク値と傾向を判断する
  - ピーク値は10コア？それとも20コア？

活動報告3

仮想化環境(KVM)検証

検証概要

- 検証目的
  - 仮想環境におけるデータベース性能への影響調査
  - ベンチマーク結果公開のため仮想化技術はKVMを使用
- 検証の観点とシナリオ
  - 参照系処理への影響の大きさを知る

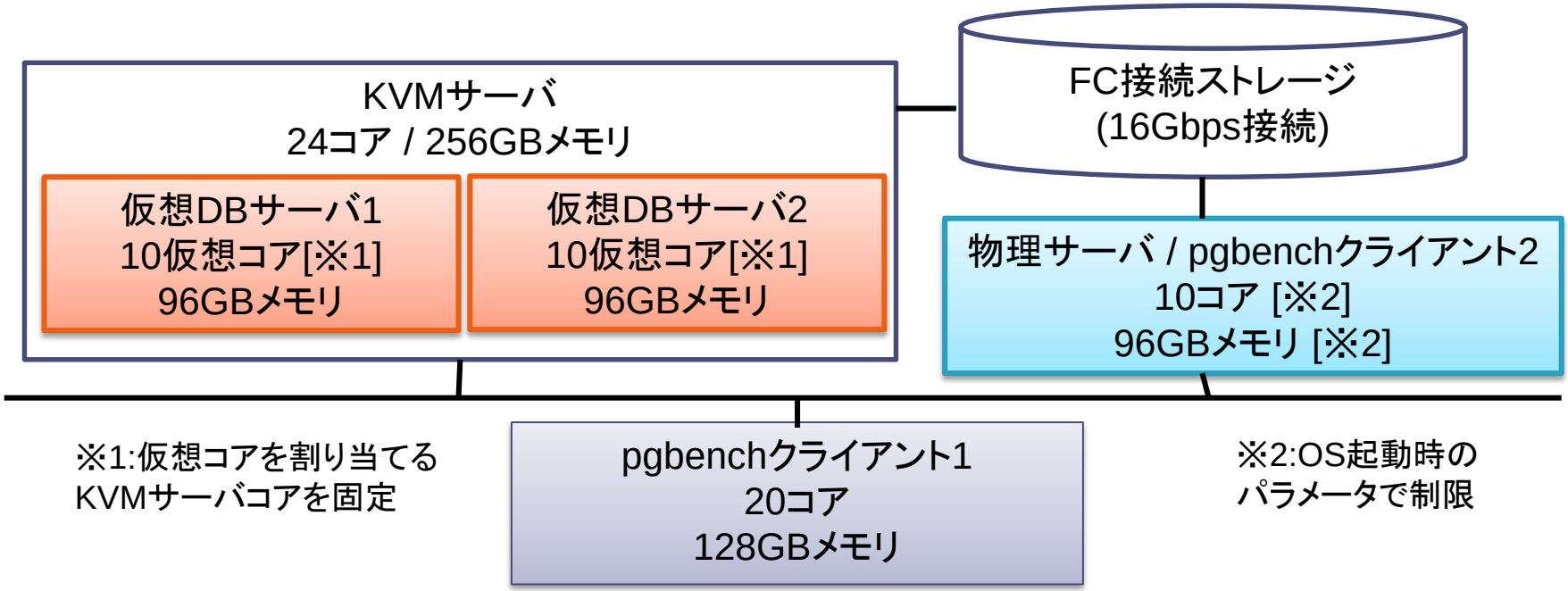
    ⇒シナリオ1：参照系ワークロードにて、仮想・物理の性能を比較
  - 更新系処理への影響の大きさを知る

    ⇒シナリオ2：更新系ワークロードにて、仮想・物理の性能を比較
  - 仮想DBを同時に実行した場合の性能への影響を知る

    ⇒シナリオ3：仮想DBの単独実行と仮想DBを同時に2つ実行した場合で性能を比較

測定環境

- KVMサーバに仮想サーバを二つ作成しそれぞれにDBを作成
- 物理環境と仮想環境のコア数とメモリ量を統一して比較
  - コア数：10コア、メモリサイズ：96GB
- WAL領域は仮想サーバ内から直接外部のストレージ領域に格納
- ベンチマークプログラムは定点観測（参照系・更新系）に同じ

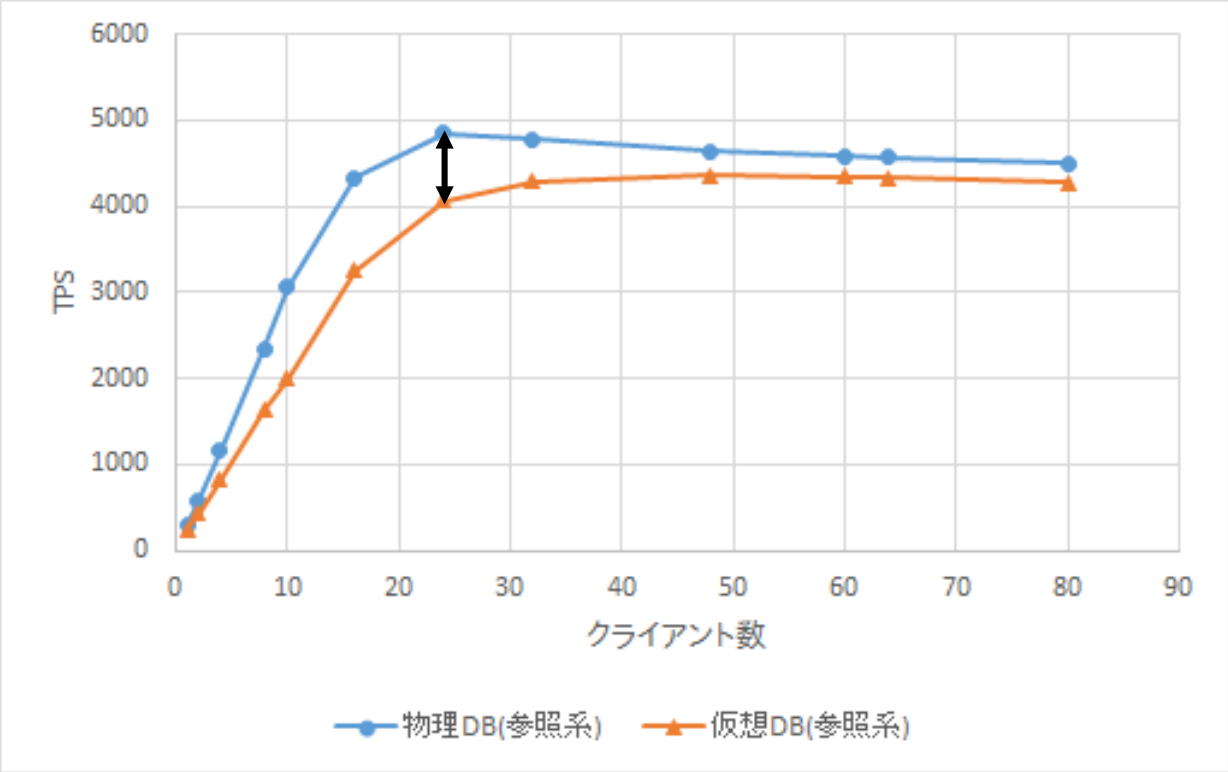


測定条件

- 参照系：(シナリオ1)
  仮想化によるCPU処理のオーバヘッドの影響を確認する
  - スケールアップ検証（参照系）と同じ pgbench のカスタムスクリプトを使用
  - DBサイズは約15GBで、検索はオンメモリで処理される
- 更新系：(シナリオ2,3)
  仮想化によるストレージの書き込みへの影響を確認する
  - スケールアップ検証（更新系）と同じくpgbenchを使用
  - DBサイズは約100GBで、検索はほぼオンメモリで処理される
  - ストレージへの書き込みはWAL（トランザクションログ）出力のみとなるように、CHECKPOINTが実行されない設定をとする
    - DBクラスタは仮想マシン上のストレージプール、
      WALファイルは直接外部のストレージ領域に格納
      （詳細は付録を参照）

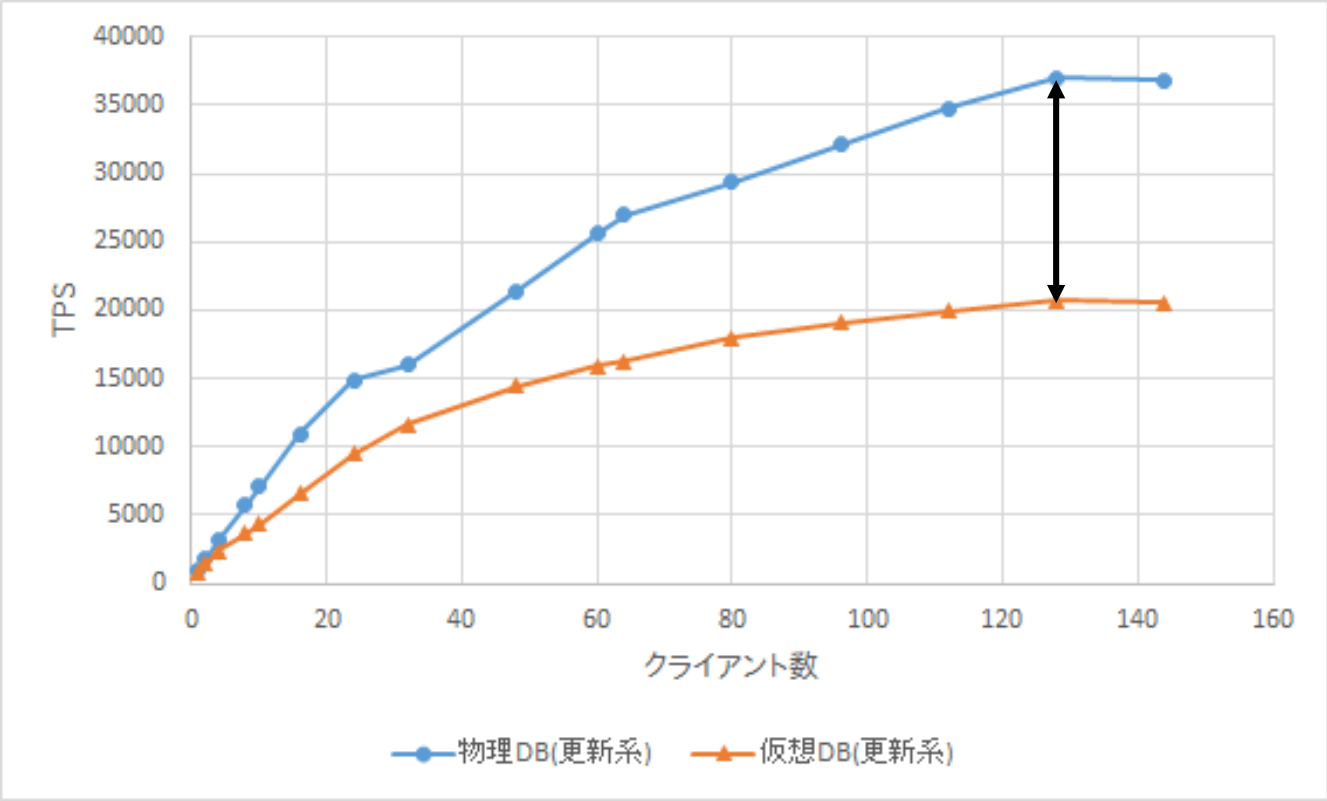
シナリオ1：参照系での物理と仮想の性能比較

- ピーク時性能の差は約10%
  - 物理は24クライアント、仮想は48クライアントでピーク
  - 24クライアントで比較すると差は約16%
- クライアント数が多くなると性能差は減少



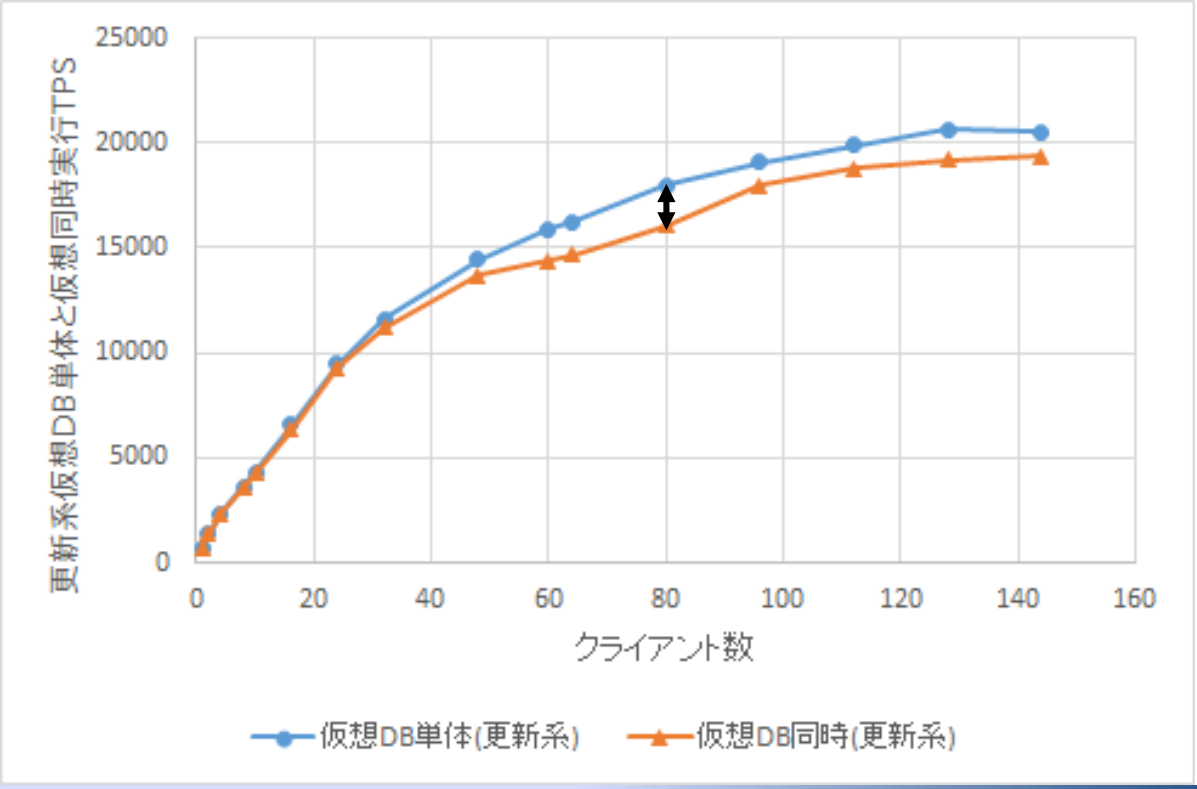
シナリオ2：更新系での物理と仮想の性能比較

- ピーク時性能の差は約40%（128クライアント時）
- クライアント数が増えると性能差が増大



シナリオ3：更新系仮想単体と仮想複数同時性能比較

- シナリオ2（更新系）と同じベンチマークを使用
- ピーク性能の差は約7%（128クライアント時）
  - 最大で約10%の差が見られた（80クライアント時）
- クライアント数が増えても、性能差の変動は見られない



結果まとめ（参照系ー検証1）

- ピーク値
  - 物理で4,900TPS、仮想で4,400TPS程度と約10%の違い
  - ピーク値状態でのCPU利用率はともに、ほぼ100%
    - 仮想環境でのピーク値低下は仮想化によるオーバヘッドによるものと思われる
- ピーク値に達するクライアント数
  - 物理で24、仮想で48と差がある
  - 24クライアント（仮想）でのCPU利用率を見るとidleが見られることから、仮想化によるレイテンシの増加があると思われる

結果まとめ（更新系ー検証2）

- ピーク値
  - 物理で37,000TPS、仮想で20,700TPS程度となり仮想環境では約半分の性能しか出ない
    - 物理・仮想で約20,000TPSの時のCPU利用率はほぼ同じ
      - CPU処理のオーバヘッドが原因ではない
    - 仮想環境では、ピーク値が出ている時でも、CPU利用率のidleが20%を超えておりCPUを使いきれていない
  - 検証3にて2DBを同時走行させたケースのピーク値は19,400TPS程度であり、検証2の仮想環境と大差はない
    - ホストOS側やその先のストレージ装置がボトルネックになっているのではなく、ゲスト-ホスト間のレイヤのどこかにボトルネックがあることが推測される。
- ピーク値に達するクライアント数
  - 物理・仮想ともに128で同じ

結果まとめ（単体と同時実行の比較ー検証3）

- ピーク値
  - 単体走行で約20,700TPS、同時走行で約19,400TPSであり、単体の方が約7%優れる
  - 同時走行した2つのDBで、ほぼ同じ
    - クライアント数を変えても、ほぼ同じスループットであった
- ピーク値に達するクライアント数
  - ピーク値に達するクライアント数は単体、同時走行ともに128であった

活動報告4

コンテナ環境(Docker)検証

コンテナ環境(Docker)での性能検証

- 背景
  - アプリケーションの可搬性の向上、運用コストの削減、デプロイの効率化などの理由でコンテナ実装のひとつであるDockerが注目されている
  - しかし、PostgreSQLなどのOSSデータベースをコンテナ環境で動作させたときの性能を含めた検証はあまり行われていない
- 目的
  Dockerを上でPostgreSQLを動作させた際の性能を実機検証によって確認する
  - オーバヘッドの有無とオーバヘッドを軽減する運用手法の調査
  - 更新系・参照系での性能傾向の把握

測定条件・環境

- 測定ツールはpgbenchを利用し、SF=1,000（約15GB）でDBを作成
  - 負荷かけサーバは20コア、メモリ128GB、Red Hat Enterprise Linux（以下RHEL）6.5、PostgreSQL 9.4RC1
  - コンテナ検証サーバは24コア、メモリ256GB、RHEL6.6、PostgreSQL 9.4RC1
    （RHELが6.6なのは、6.5だとDockerをインストールできないため）
  - DockerコンテナのOSはRHEL6.5の公式イメージからインストール
- 実際に使用したマシンは付録をご覧ください

測定条件・環境(続き)

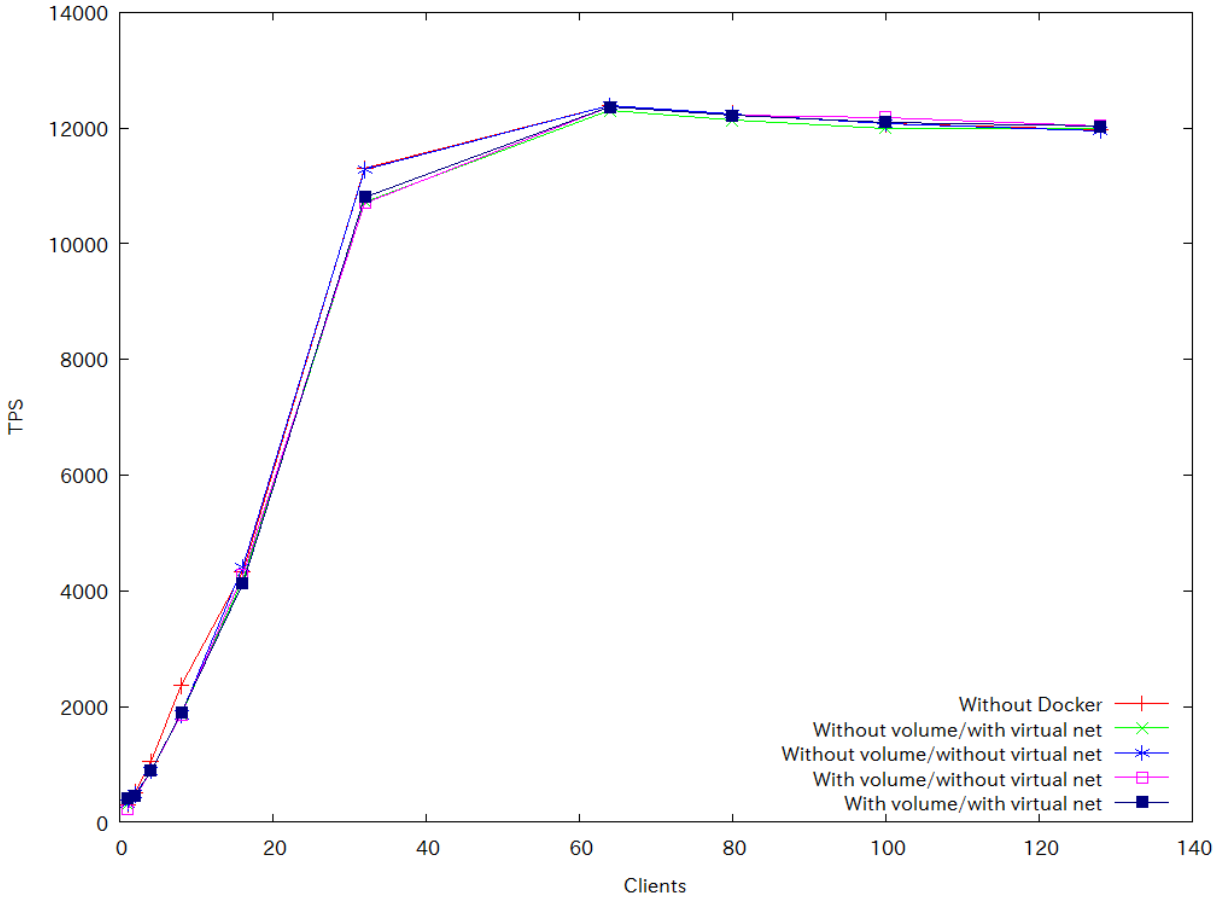
- テスト共通
  - Dockerを使わない場合と、Dockerを使う場合(4通り)を比較

Dockerを使う場合のテストケース		ネットワーク	
		コンテナ仮想	ホスト直接
ファイルシステム	ホスト直接	ケース1	ケース2
	コンテナ内	ケース3	ケース4

  - 3回の計測結果の中央値を結果とする(各回5分)
- 参照系のテスト
  - クライアントからの同時接続数を1から128まで変化
- 更新系のテスト
  - 同時32ユーザのみ測定
  - 小さなテーブルへ更新が集中してボトルネックになるのを避けるためにpgbench -Nを使用

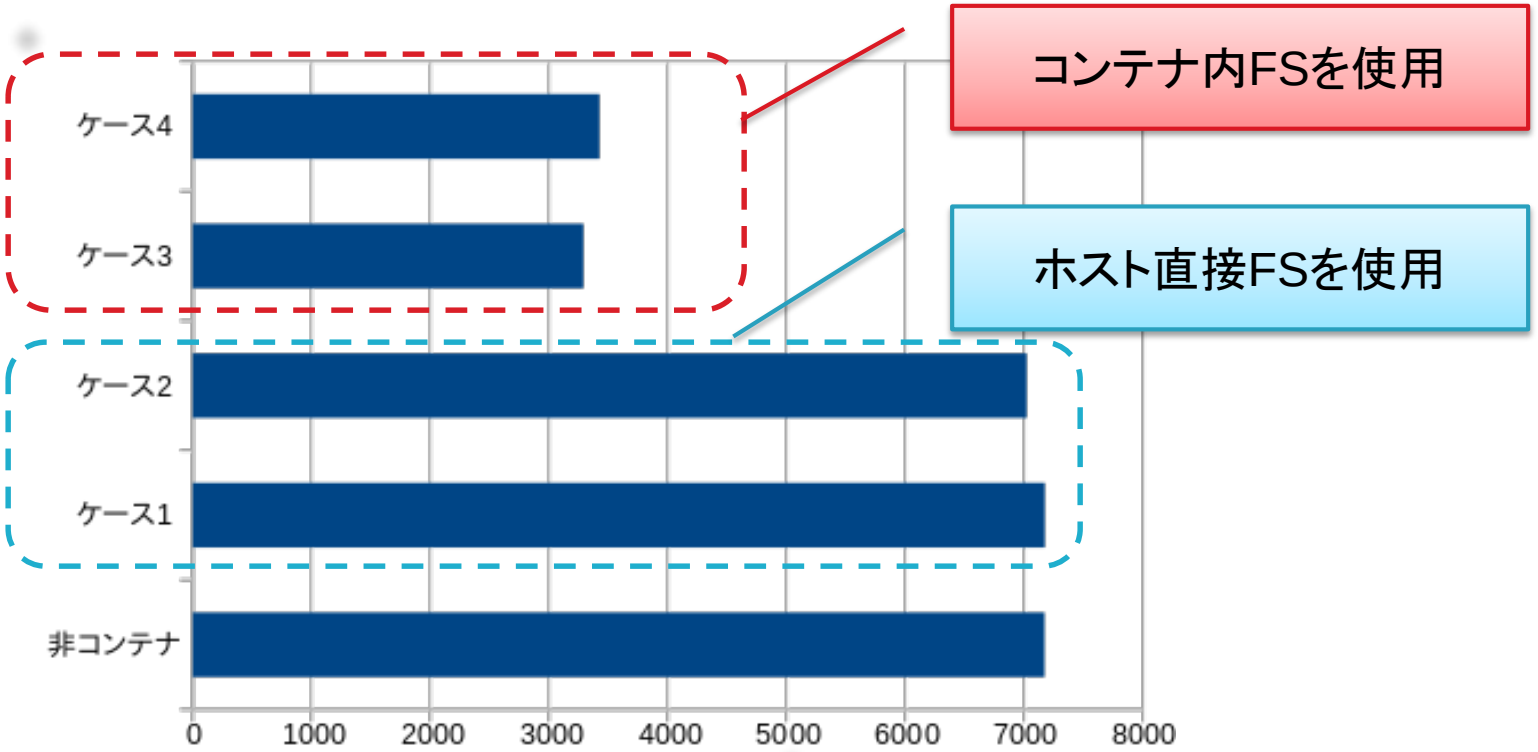
参照系性能測定結果

- すべての測定パターンでほぼ結果が同じ（誤差の範囲）
- 参照性能へのDockerの影響は最小限



更新系性能測定結果

- コンテナ内のファイルシステムを使うケースでは大きな性能低下が見られる



結果まとめ

- 参照系
  - オーバヘッドによる性能低下は見られない
- 更新系
  - ホストOSのファイルシステムにDBを格納するケースではオーバヘッドによる性能低下は見られない

    このようなファイル配置とすることで、オーバヘッドによる性能低下を回避できる
  - コンテナ内のファイルシステムにDBを格納するケースで大きく低下する

    ただし、コンテナ内にDBを置くような運用はプロダクション環境ではあまり考えられず、実運用の障害にはならないと思われる
- 全体的な性能の傾向
  - オーバヘッドを除いて、物理環境と大きな違いはなく、Dockerにも不安定な挙動などは見られなかった

活動報告5

2014年度活動をふりかえって

2014年度活動をふりかえって

- 今年度のWG1検証はいつもの9月頃リリースから大幅に遅れて待ちに待ったPostgreSQL9.4を2014年12月に検証を実施しました。
- 今年度は9.4新機能のWAL改善の観点も含めて更新系ベンチマークを実施しました。新しい環境として仮想化環境やコンテナ環境などの新しいことにもチャレンジしました。
- 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。

2014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

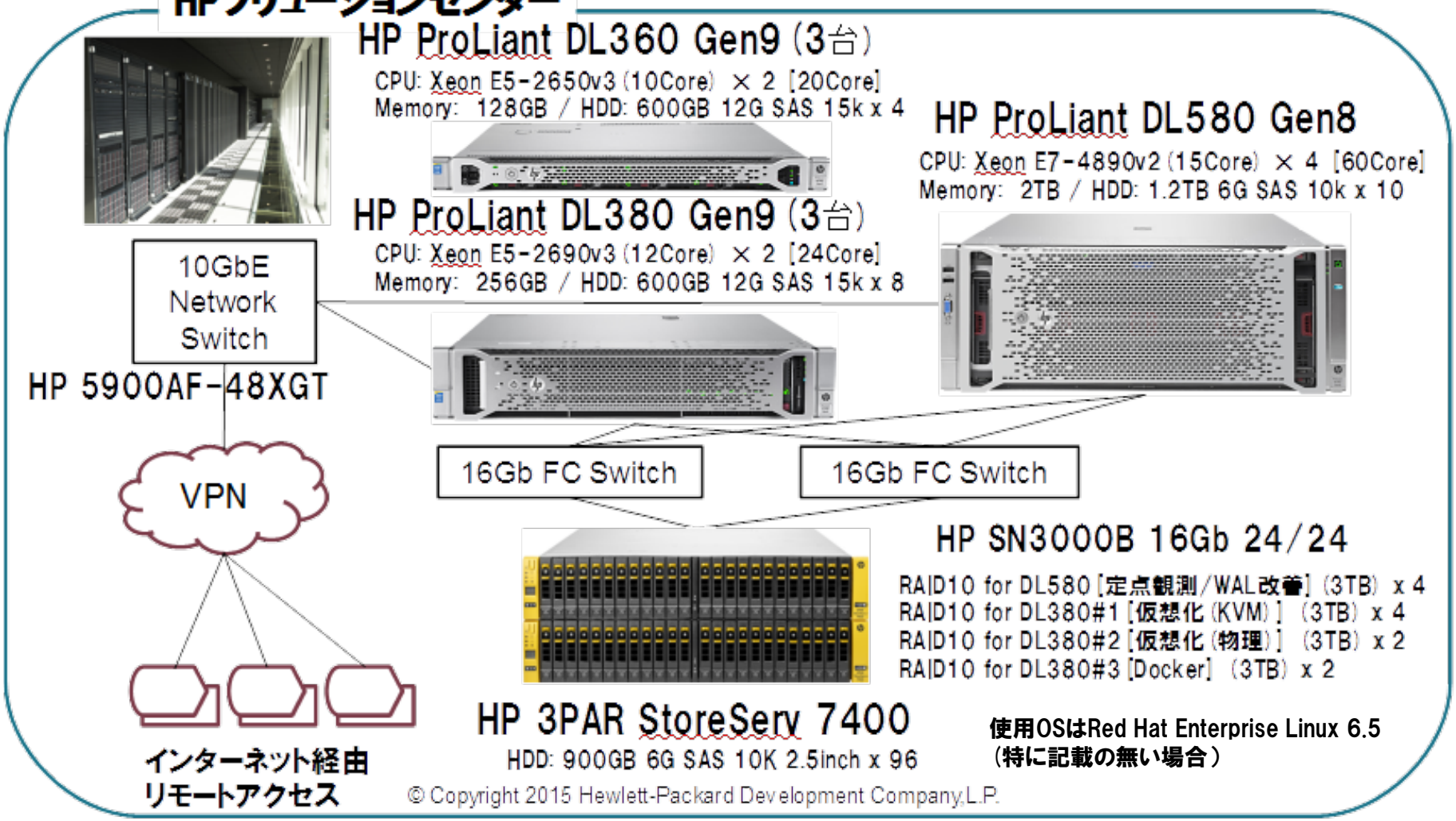
今回使用した検証環境について
仮想環境でのストレージ構成について

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。

付録：検証環境 （提供：日本ヒューレット・パッカード株式会社）

- 2014年度WG1性能検証での使用機器／設備



付録：仮想環境でのストレージ構成について

作成コマンド(ストレージプール)

```
#virsh pool-define-as vv13 fs - - /dev/mapper/mpathdp1 - /var/lib/libvirt/vv13/
```

- 仮想環境で使用するストレージは、仮想マシンを作成したストレージプールと、仮想マシンから直接mountするホスト上の外部ブロックデバイスの双方が利用可能。

```
#virsh attach-device vmpsql01 ~/NewStorage.xml
```

作成コマンド(外部ストレージ)

NewStorage.xml

```
<disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none'/>
    <source dev='/dev/mapper/mpathbp1'/>
    <target dev='vdb' bus='virtio'/>
</disk>
```

- 今回の更新系ベンチマークはCHECKPOINTが発生しないようにしてWALファイルへの書き込みが性能の上限となって、外部ブロックデバイスへのWALファイル配置が性能の決め手となった。

パターン	ファイルのマウント先		性能
	データベースクラスタ	WALファイル	
1	ストレージプール	ストレージプール	低
2(今回採用)	ストレージプール	外部ブロックデバイス	高
3	外部ブロックデバイス	外部ブロックデバイス	高

PGECons

PostgreSQL Enterprise Consortium