PGECons
PostgreSQL Enterprise Consortium

2022年度WG3活動成果報告
性能トラブル調査編@Amazon Aurora

PostgreSQL エンタープライズコンソーシアム
WG3 パブリッククラウド検証チーム

Contents

- 活動概要
- 検証結果
  - autovacuumのパラメータ検証
  - 性能状態を把握するための監視
  - ケーススタディ
- 総括

責任範囲

- 本資料は、PGEConsが独自に検証した結果であり、
  結果はPGEConsの責任の元、公開しています。

活動概要

- PGECons WG3の過去検証：”性能トラブル調査編”は、以下の観点で性能トラブル対策の一助となる情報を提供
  - 予防：性能トラブルを発生させないために考慮すべきポイント
  - 検知：性能状態を把握するために必要な監視情報や調査観点
  - 事例：性能トラブル事例をもとに、性能情報等を使用した調査

- 上記の検証を基に、Amazon AuroraのPostgreSQL版（以降、Aurora）で適用する際の考慮点を調査
  - 既存のチューニング手法が適用できるか（実施する必要があるか）
  - Aurora独自の機能で対応可能か

活動概要

- 過去検証の中から、一部項目を抽出し、Auroraにおける影響を確認

- 検証項目
  - autovacuumのパラメータ検証
  - 性能状態を把握するための監視
  - ケーススタディ
    - ディスク性能の考慮漏れによる性能トラブル
    - 適切でない実行計画が選択されてしまうことによる性能トラブル

活動概要

- autovacuumのパラメータ検証
  - autovacuumの処理性能を向上させるパラメータ
    - autovacuum_vacuum_cost_delay
    - autovacuum_vacuum_cost_limit
  - パラメータ変更を行い、過去検証と同じ傾向がAuroraでも得られるかどうか実機検証を実施
  - AuroraはautovacuumのVACUUM実行時間が短くテーブルサイズの肥大化が起こりにくいことを確認

活動概要

- 性能状態を把握するための監視
  - 過去検証で整理されている性能情報の調査方法に対し、Auroraの機能で確認する方法があるか調査
    - Amazon RDS Performance Insights（Performance Insights）
    - Amazon RDS 拡張モニタリング（拡張モニタリング）
    - Amazon CloudWatch Logs（CloudWatch Logs）
  - 上記の機能により、確認することの出来る性能情報を整理し、確認手順も簡単に記載

活動概要

- ケーススタディ
  - ディスク性能の考慮漏れによる性能トラブル
    - ストレージの性能特性(HDD/SSD)の考慮漏れによる性能トラブル、およびパラメータチューニングの対応がAuroraでは発生しない(最適値設定されている)
  - 適切でない実行計画が選択されてしまうことによる性能トラブル
    - チューニングの1つの手法としてAuroraの機能であるQuery Plan Management(QPM)による実行計画の固定化を紹介

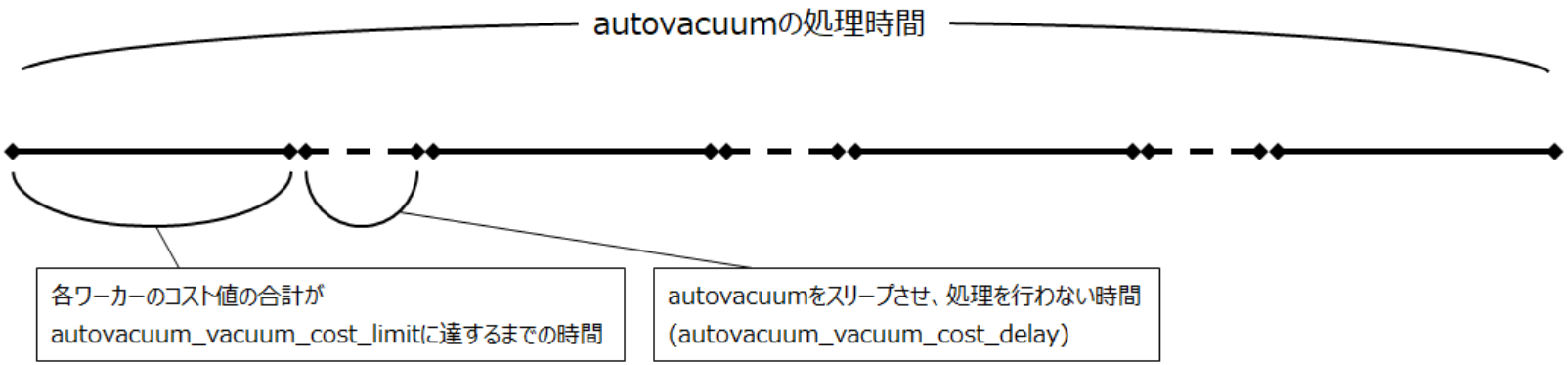
autovacuumのパラメータ検証

autovacuumのパラメータ検証

■ 検証概要

□ autovacuum

- 定期的にテーブル状態を監視し、不要領域が閾値を超えた場合にVACUUMとANALYZEを実行
- VACUUMを実行する際は、各ワーカーでのコスト値の合計がautovacuum_vacuum_cost_limitに達した時にautovacuum_vacuum_cost_delayの間作業を中断することで、I/O負荷が高負荷にならないように動作



※コスト値：シーケンシャルI/Oで1ページを読み込むコストを1.0とした際の相対値

https://pgecons-sec-tech.github.io/tech-report/html_wg3_performance/wg3_performance.html

autovacuumのパラメータ検証

- 検証概要
  - 更新によって発生する不要領域に対してautovacuumが追い付かない場合、テーブルサイズの肥大化が発生する
  - テーブルサイズの肥大化を防ぐautovacuumのパラメータチューニングが検証の目的
  - 以下のパラメータを変更し、autovacuumによる不要領域の回収速度を測定
    - autovacuum_vacuum_cost_limit
    - autovacuum_vacuum_cost_delay

autovacuumのパラメータ検証

■ 検証環境

項目	本検証パラメータ	過去検証
マシンスペック	db.r5.large (2vCPU, 16GB MEM)	2vCPU, 8GB MEM
Aurora	14.5.1	-
PostgreSQL	14.5	11.1
pgbench	PostgreSQL 14.1 付属	PostgreSQL 11.1 付属

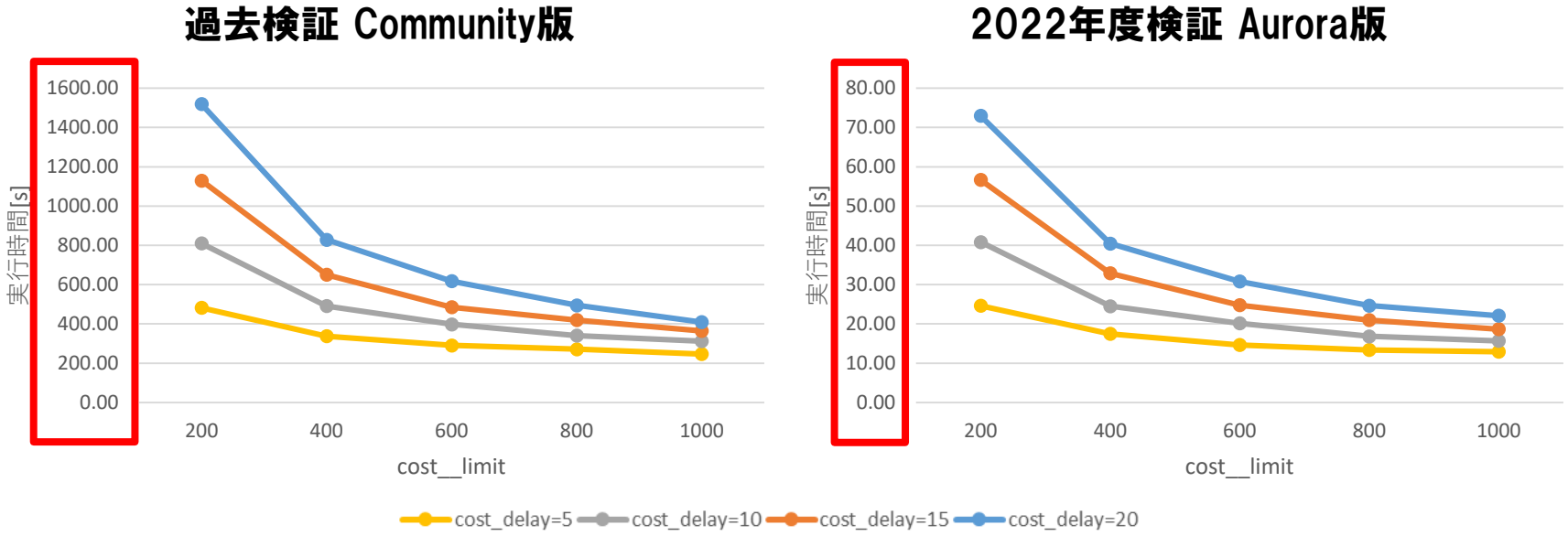
autovacuumのパラメータ検証

■ 検証環境

項目	本検証パラメータ	過去検証
max_connections	300	同左
shared_buffers	2GB	同左
effective_io_concurrency	256	200
max_wal_size	1GB	4GB
min_wal_size	0.5GB	2GB
checkpoint_completion_target	0.9	同左
random_page_cost	1.1	同左
autovacuum_vacuum_cost_limit	{200, 400, 600, 800, 1000}	同左
autovacuum_vacuum_cost_delay	{20, 15, 10, 5}	同左
autovacuum_vacuum_scale_factor	0.2	同左

autovacuumのパラメータ検証

■ 検証結果（autovacuum処理時間）



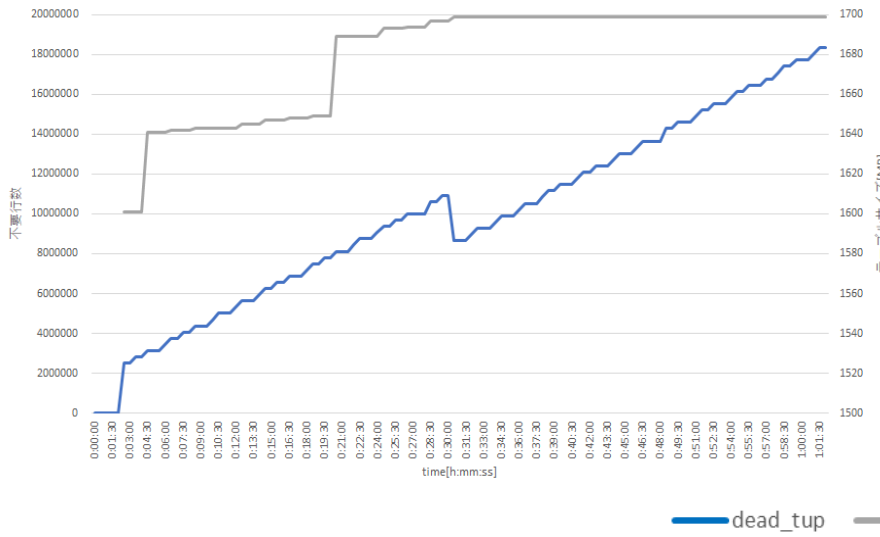
- 各パラメータによるautovacuumの処理時間は過去検証と同じ傾向
- Auroraのautovacuum実行時間が短い

https://pgecons-sec-tech.github.io/tech-report/html_wg3_performance/wg3_performance.html

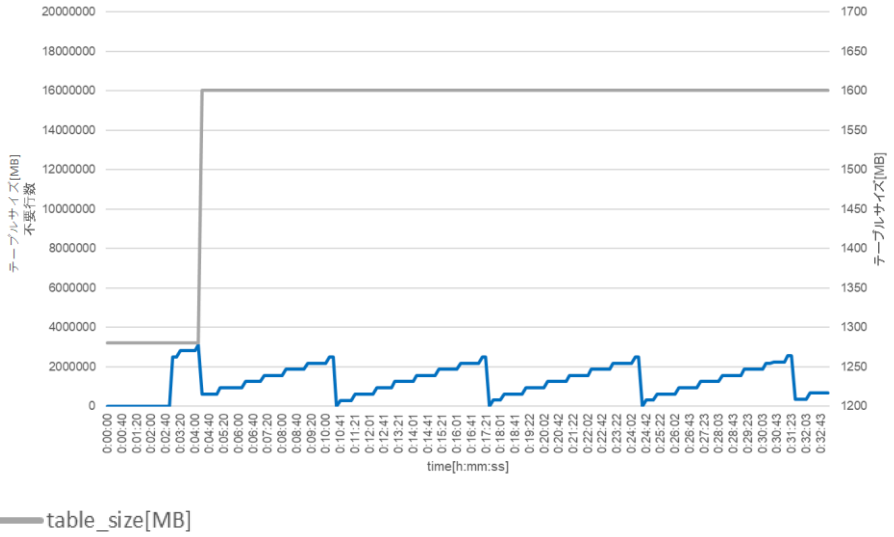
autovacuumのパラメータ検証

- 検証結果（更新トランザクションと不要行数の推移）

過去検証 Community版



2022年度検証 Aurora版



- Auroraのautovacuum実行時間が短い
  - 過去検証と同じパラメータでは、
    不要領域の増加によるテーブル肥大化は発生しない

https://pgecons-sec-tech.github.io/tech-report/html_wg3_performance/wg3_performance.html

autovacuumのパラメータ検証

- 検証結果考察
  - Auroraのautovacuumの性能が高い結果は、以下の要因が複合的に関連していると推測
    - メジャーバージョンによるvacuumパフォーマンスの差異
    - HW 性能の不均一性
    - DB パラメータの差異
    - Auroraのアーキテクチャーの優位性
  - ただし、追加検証から、同一 HW、同一メジャーバージョンという前提において、Auroraにおける アーキテクチャーの違いがautovacuumの性能差異の 大きな要因であると推測
    - ※ Amazon Web Services Japan社にご相談し、追加検証を実施いただきました

autovacuumのパラメータ検証

- 追加検証概要
  - RDSとAuroraの以下PostgreSQLバージョンにおいてautovacuumの処理時間を計測
    - RDS : PostgreSQL 11, 12, 13, 14
    - Aurora : PostgreSQL 11, 12, 13, 14
  - RDSとAuroraの各バージョンにおける測定結果から、複合的に関連している要因について考察

- 結果は成果報告書の4.4節（追加検証内容）を参照

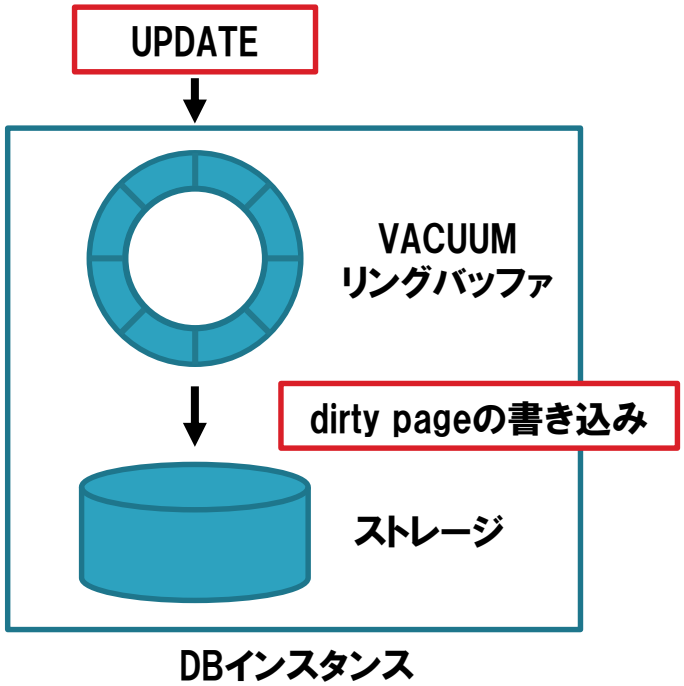
autovacuumのパラメータ検証

■ 検証結果考察

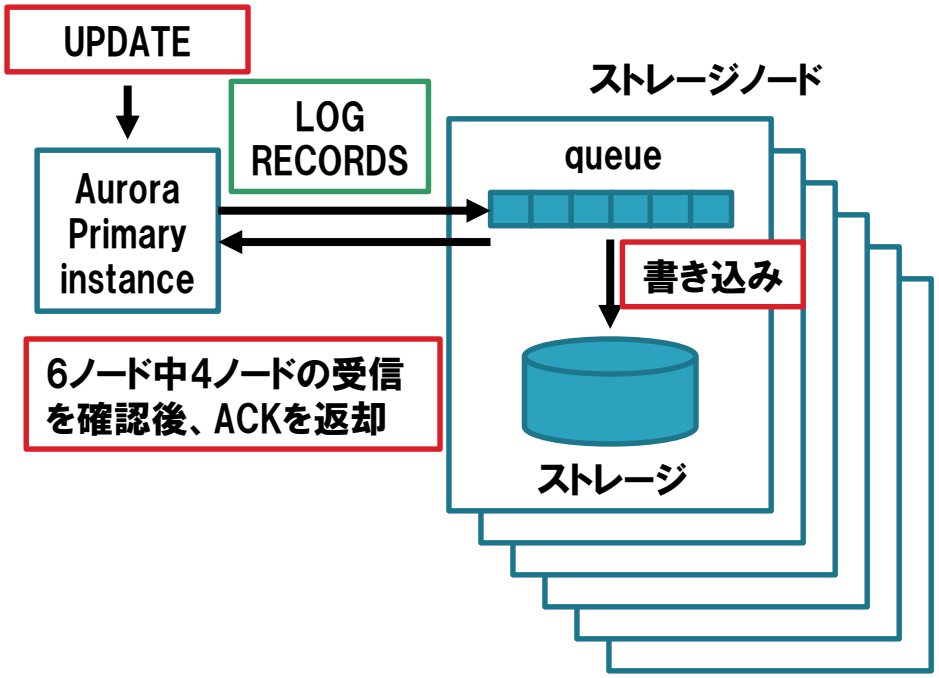
□ Auroraのアーキテクチャー

- ページイメージをストレージに書き込む処理が非同期であるため、Community版と比べて書き込みを待つ必要がなく、処理が高速

Community版



Aurora



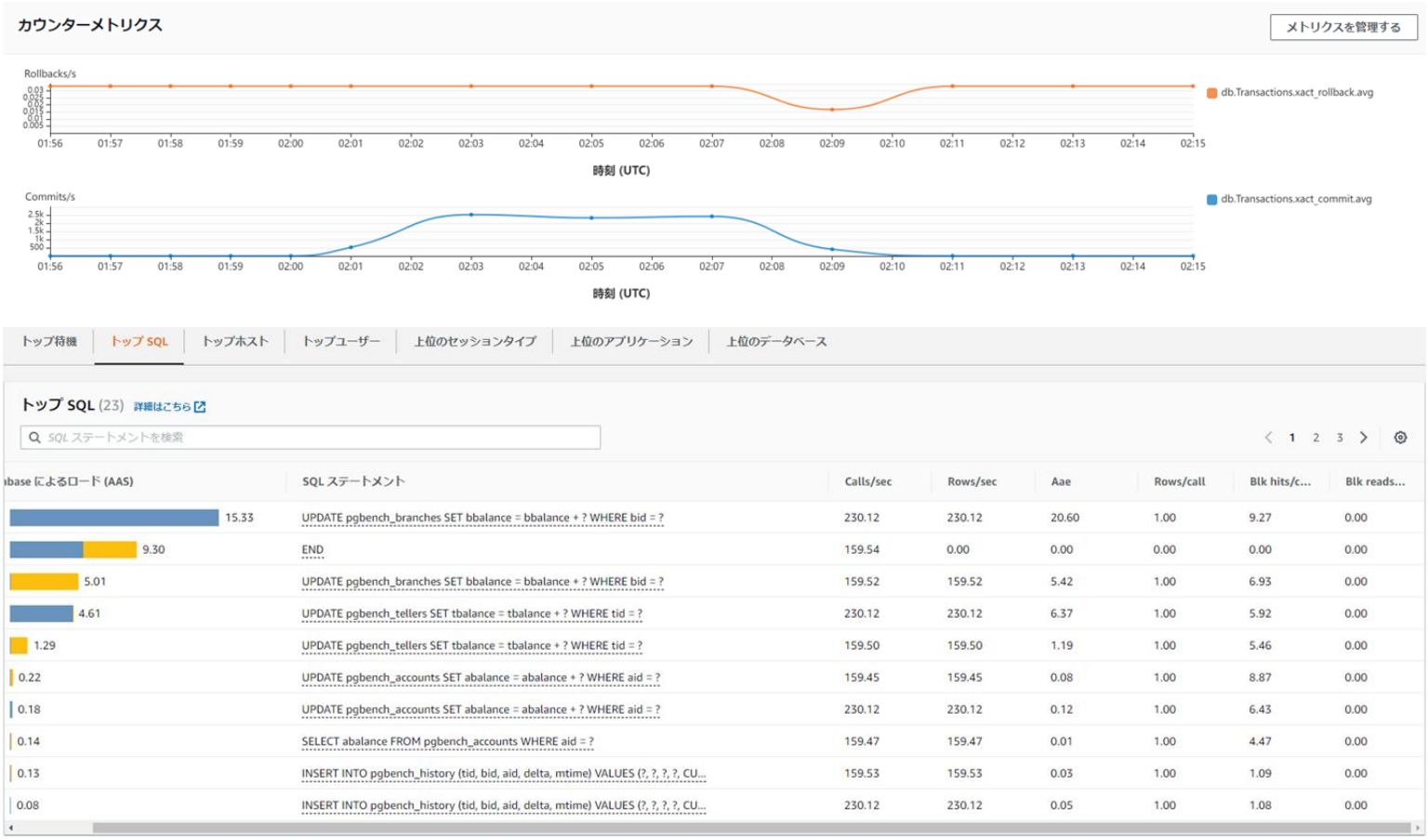
性能状態を把握するための監視

性能状態を把握するための監視

- 検証概要
  - 過去検証では、トランザクション数やCPU使用率などの性能に関連する情報（性能情報）の確認方法を調査
  - 本資料では、Auroraにおける性能情報の確認方法についてまとめる
  - 以下のいずれかの方法で確認
    - Amazon RDS Performance Insights（Performance Insights）
    - Amazon RDS 拡張モニタリング（拡張モニタリング）
    - Amazon CloudWatch Logs（CloudWatch Logs）
    - 確認用 SQL（過去検証と同じ SQL）
  - Performance InsightsなどAurora独自の機能で確認用SQLと同等の性能情報が確認できるか調査

性能状態を把握するための監視

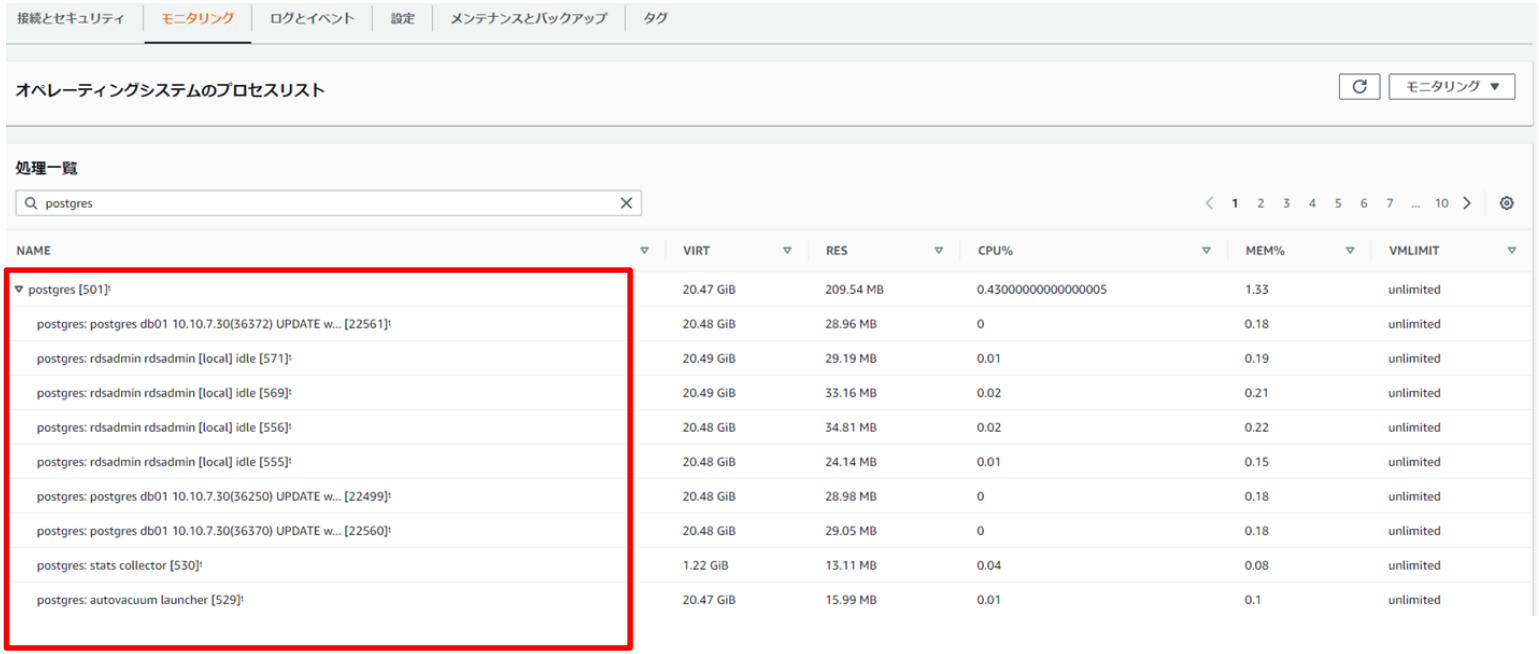
- 検証結果
  - Performance Insights



性能状態を把握するための監視

■ 検証結果

□ 拡張モニタリング

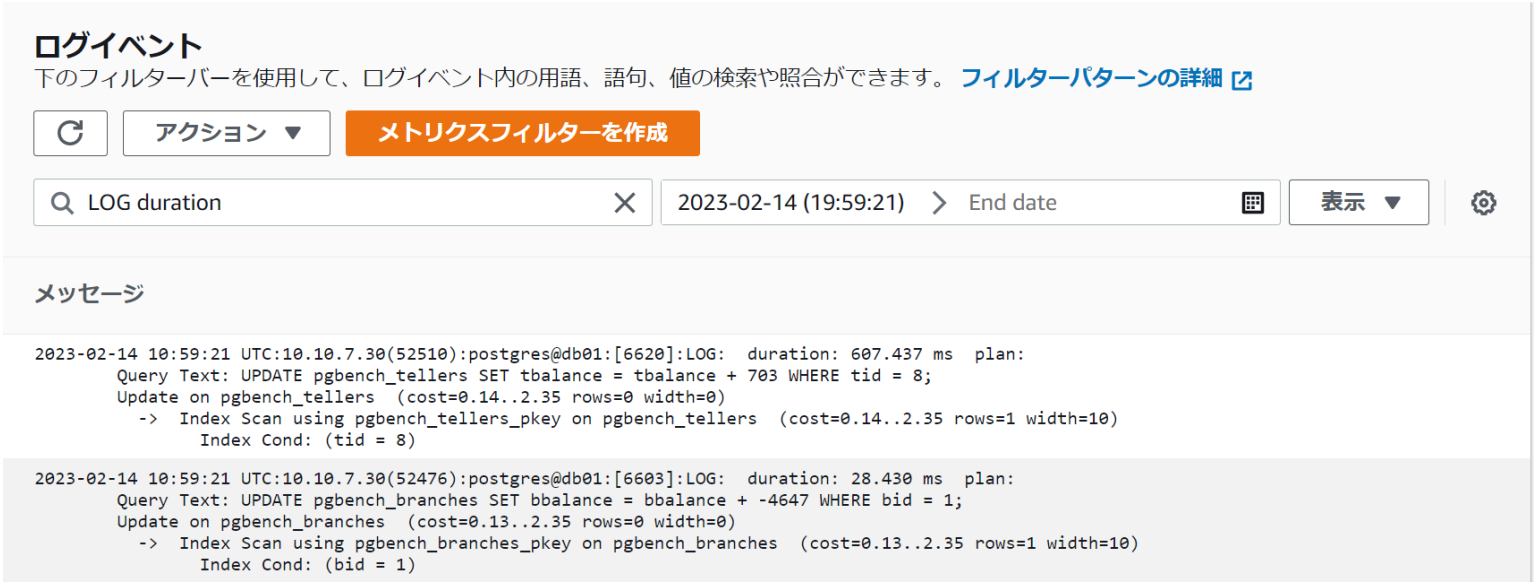


PostgreSQLプロセス

性能状態を把握するための監視

■ 検証結果

□ CloudWatch Logs



性能状態を把握するための監視

- 検証結果まとめ
  - Auroraも過去検証のSQLを実行可能
  - 以下の各種機能は用途に合わせて使い分ける

機能	特徴
Performance Insights	OSメトリクスはCloudWatch等からも確認できるが、グラフィカルにDBメトリクスや高負荷SQLの確認が可能
Amazon RDS 拡張モニタリング	OSメトリクスはCloudWatch等からも確認できるが、OSプロセスごとのメトリクスも確認可能
CloudWatch Logs	RDS標準機能でもログは確認だが、文字列検索など簡単にログデータのリアルタイム分析が可能

  - 基本的にはPerformance Insightsでインスタンス全体の傾向を確認後、詳細情報はSQLで確認といった使い分けが適している

ケーススタディ

ケーススタディ

- 検証概要
  - 過去検証の性能トラブルのケーススタディを通して、Auroraにおけるチューニング手法を調査
    - ディスク性能の考慮漏れによる性能トラブル
      - Auroraにおけるディスク関連の性能トラブルについて整理
    - 適切でない実行計画が選択されてしまうことによる性能トラブル
      - Auroraの機能であるQuery Plan Management（QPM）による実行計画の性能トラブル手順を整理

適切でない実行計画による性能トラブル

- トラブル概要
  - 最適でない実行計画の選択によるSQLの実行時間劣化に対し、過去検証ではDBパラメータの変更や、統計情報の更新を行うことで改善

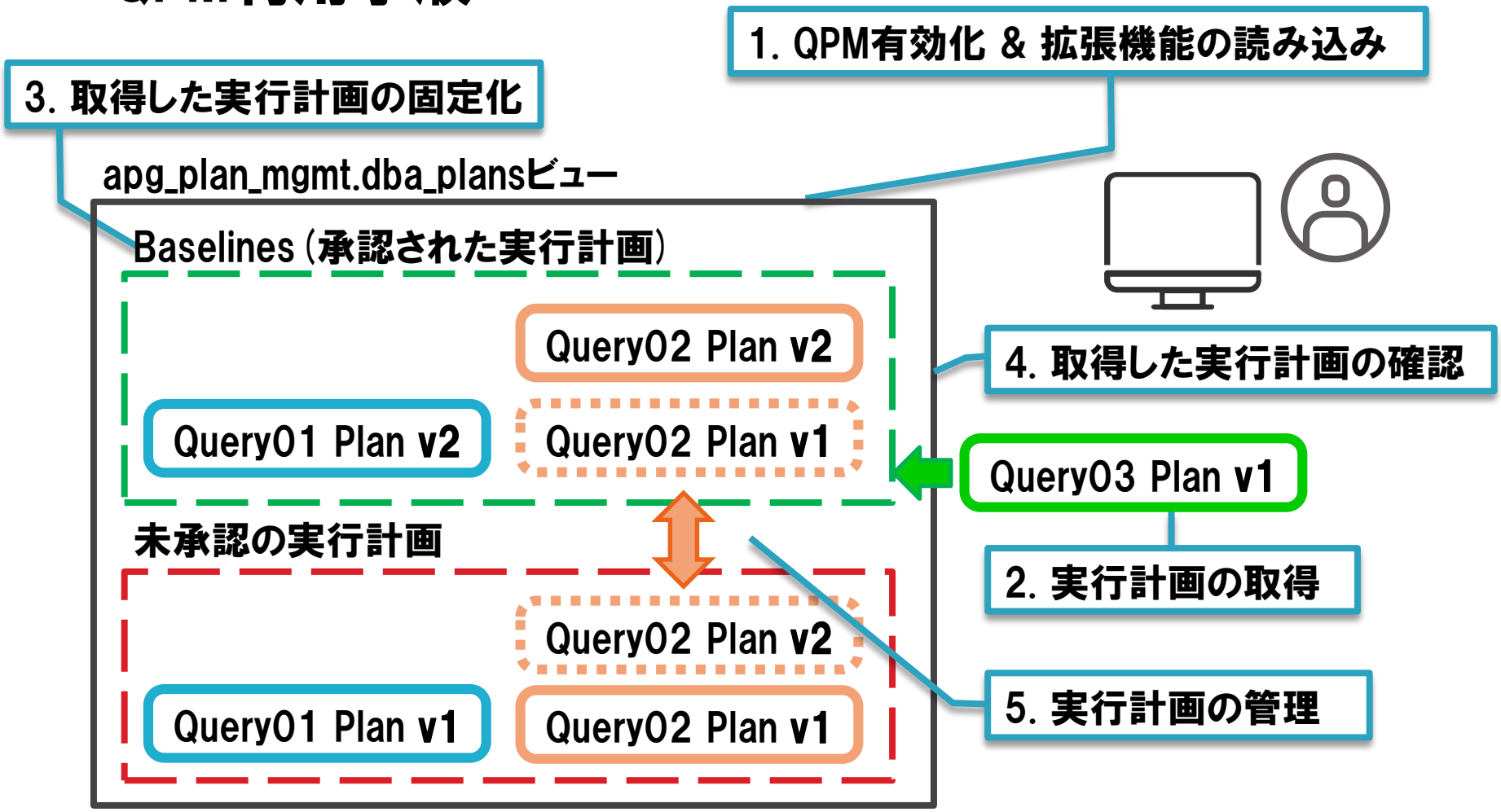
- Auroraにおける調査結果
  - 上記のDBパラメータの変更や、統計情報更新は有効
  - Aurora特有の機能であるQuery Plan Management (QPM)を用いることで、特定クエリの実行計画を固定化が可能
    - 性能チューニング方法の1つとして利用できる

適切でない実行計画による性能トラブル

- QPM概要
  - クエリの実行計画の変更方法と変更タイミングを制御し、管理する機能
  - QPMが有用なケース
    - 実行計画の固定化が可能となり、実行計画のリグレッション回避
    - パッケージ製品を利用している等、アプリケーション(SQL文)の書き換えが難しい場合
    - 緊急対応として直前の実行計画に戻す場合
  - 注意点
    - RDSでは使用できない(Auroraの機能)
    - QPMの管理対象を整理し、実行計画の更新などの運用を行う

適切でない実行計画による性能トラブル

- QPM利用手順



総括

総括

- 性能トラブル対策の一助となる過去検証に対し、Auroraで適用する際の考慮点を確認
  - autovacuumのパラメータ検証
    - Auroraはautovacuumの実行時間が短く、テーブルサイズの肥大化が起こりにくいことを確認
  - 性能状態を把握するための監視
    - Performance Insights等を用いて確認できる項目を整理
  - ケーススタディ
    - 実行計画のコスト算出に用いるディスク性能周りのパラメータが最適な値に設定されていることを確認
    - QPMによる実行計画固定化を用いたチューニング手法の紹介

PGECons

PostgreSQL Enterprise Consortium