C-010

オンプレミスとパブリッククラウドで構成されるデータベースのスケールアウト方式の提案と評価
Proposal and evaluation of scale-out method for database consisting of on-premises and public cloud

和田 清美† 林 真一† 金子 聡†
Kiyomi Wada Shinichi Hayashi Satoshi Kaneko

1. はじめに

オンプレミスのプライベートクラウドは，使用できるリソース量に限りがあるので，負荷変動が小さい定常業務での利用が適している。一方，月次処理などのバッチ処理や季節的な負荷変動が大きい処理は，パブリッククラウドを一時利用することで，リソース所有コストを増やさず，高速に処理することができるようになる。このとき，処理するデータが機密データであるため，パブリッククラウドに保存できない場合がある。そこで，オンプレミスに機密データを保存したまま，パブリッククラウドのスケール可能な計算リソースを利用することで，高速処理と所有リソースコスト低減を実現する。

本報告では，大規模データをオンライン分析処理するため，機密データはオンプレミスのストレージボリュームに保存したまま，パブリッククラウド上の Database(DB)をスケールアウトする。ここで，パブリッククラウドとオンプレミス間で分析処理のためのデータ転送の他に，DB のスケールアウトによるデータ再分配のためのデータ転送が加わり，通信性能ネックとなりスケールアウト完了に時間がかかる。

本研究では，予め最大スケールアウトに対応するボリュームのコピーを作成し，スケールアウト時に DB の再配置と，DB とストレージボリューム間の接続を変更することで，パブリッククラウドとオンプレミスに跨ったデータアクセスの DB 処理性能が低下しないスケールアウト方式を提案する。

2. 従来方式と課題

2.1 システム概要

図 1 は，オンプレミスとパブリッククラウドからなるクラウドバースティング対応のシステム構成である。

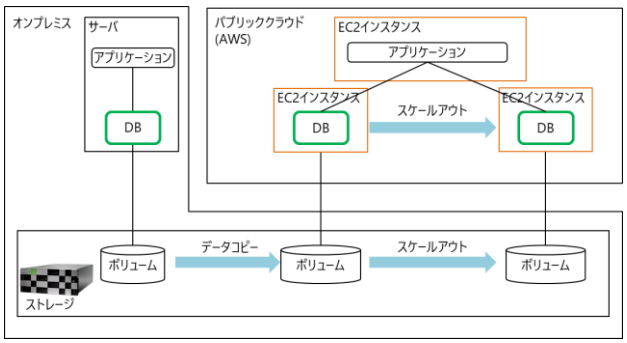

図 1 システム概要

†株式会社 日立製作所，Hitachi, Ltd.

図 1 において，オンプレミスで定常処理を実行し，大規模データ分析時は，パブリッククラウド Amazon Web Services(AWS)の Elastic Compute Cloud(EC2)インスタンスのスケールアウトに合わせて，オンプレミスのストレージボリュームをスケールする。

2.2 従来技術と課題

図 2 は，PostgreSQL の拡張機能 Citus[1]によるスケールアウト型 DB システム構成である。図 2 において，パブリッククラウドの EC2 インスタンス 1 台に複数 worker をデプロイする。アプリケーションが要求するクエリを複数 worker で並列処理する場合，EC2 インスタンスがストレージボリュームから大量のデータを読み込んで分析処理するため，EC2 インスタンスとオンプレミスのストレージ間で大量のデータ転送が発生する。このため，EC2 インスタンスの Network Interface Card(NIC)およびストレージポートは転送速度の上限に達して，性能ネックになる。そこで，EC2 インスタンスをスケールアウトし，新規 worker を追加し，既存 worker のデータの一部を新規 worker に割り当てる。ここで，データ再分配は，オンプレミスとパブリッククラウド間のデータ転送であるため，性能ネック部位に負荷が加わり，スケールアウトが完了するまで時間がかかる。

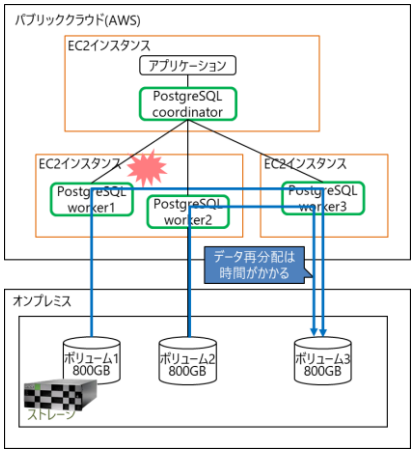

図 2 従来方式

3. 提案方式

図 3 は提案方式である。図 3 において，左がスケールアウトを実行するためのスケール制御機能，右がスケールアウト型 DB システム構成例で，右上がスケールアウト前，右下がスケールアウト後である。パブリッククラウド(AWS)には，coordinator 用の EC2 インスタンス(1 台)と，worker 用の EC2 インスタンス(初期構築時は 1 台)があり，EC2 インスタンス内に 2 つの worker を配置する。worker のデプロイ先の EC2 インスタンスは，AWS Auto Scaling group に登録する。オンプレミスのストレージには，

worker のデータを格納するボリュームがある。提案方式は，予めスケールアウトに対応するボリュームのコピーを作成し，パブリッククラウドの EC2 インスタンスのリソース過負荷を検知すると，EC2 インスタンスをスケールアウトする。このとき，インスタンス内の worker 再配置と，オンプレミスのストレージボリュームの接続先を事前コピーしたボリュームに変更する。これより，スケールアウトするためにデータ再分配を不要とし，パブリッククラウドとオンプレミス間の通信負荷をかけない。

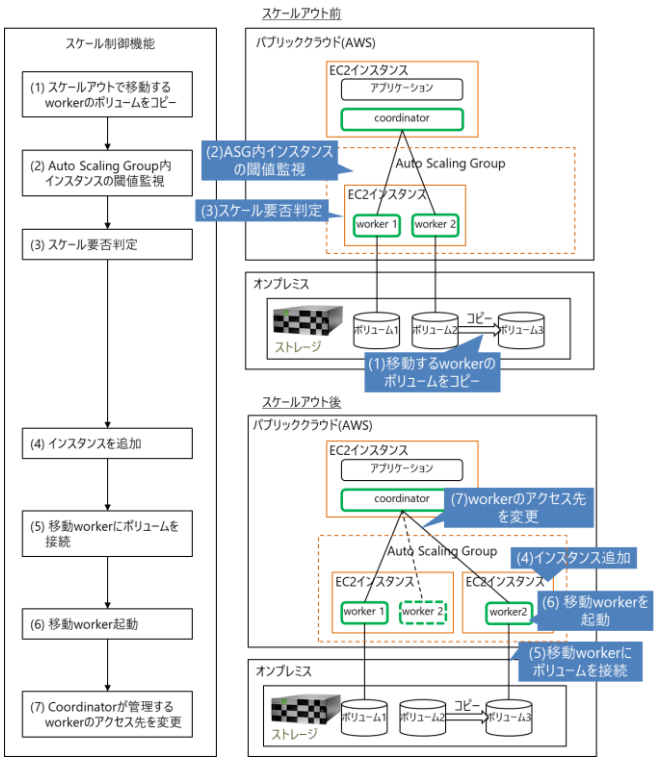

図 3 提案方式

スケールアウト処理は(1)~(5)である。(1)は，スケールアウトで移動する worker のデータ格納先ボリュームを事前コピーする。(2)は，Auto Scaling group 内の EC2 インスタンスの CPU 使用率，ネットワーク転送量を Amazon CloudWatch で閾値監視する。(3)は，閾値超過に対してスケール要否判定をする。(4)は，EC2 インスタンスを新規追加，起動し，Auto Scaling group に登録する。(5)は，新規追加したインスタンスと，移動 worker のデータ保存先であるコピー済みボリュームを接続する。(6)は，新規追加インスタンス内の移動 worker を起動する。(7)は，coordinator が管理する worker 情報を変更する。

4. 実験結果と考察

4.1 実験方法

図 3 のスケールアウト型 DB システム構成において，ベンチマークとして，オンライン分析 Online Analytic Processing(OLAP)のベンチマークプログラム Star Schema Benchmark(SSB)[2]を使用する。SSB の総データサイズは 100GB とし，分析対象となる実績値の明細データのファクトテーブルは，シャーディング(行方向でテーブル分割)して，worker に分配する。SSB の分析クエリ(計 13 クエリ)は，各 worker 毎に割り当てられたデータに対して処理実行後，coordinator で実行結果を集約する。分析クエリはデータ参照のみでデータ更新は行わない。

4.2 結果と考察

図 4 は，SSB 実行時の CPU 使用率の時系列変化である。EC2 インスタンス数が 2 台から 4 台へスケールアウトすると，CPU 使用率は 100%から 70%以下になる。

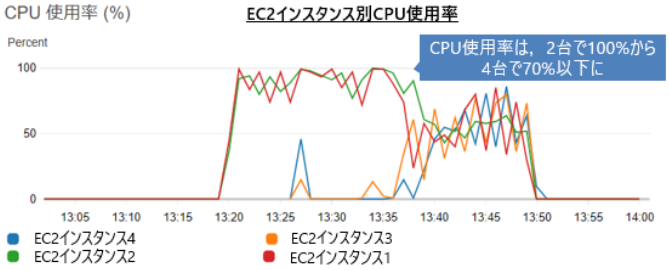

図 4 SSB 実行時の CPU 使用率の変化

図 5 は，EC インスタンス数別の SSB 処理時間を比較したものである。EC インスタンス数が 2 台から 4 台へスケールアウトすると，処理時間は 62%まで短縮できた。

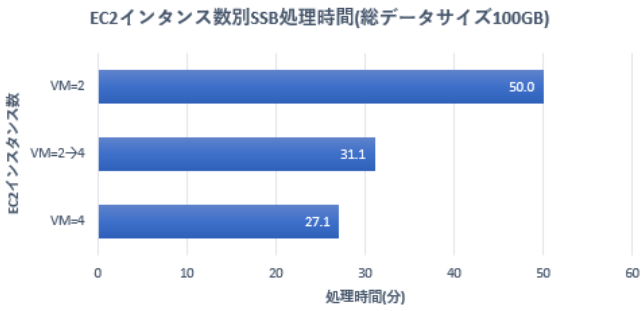

図 5 インスタンス数別 SSB 処理時間の比較

以上の実験結果より，事前に使用する EC2 インスタンス数の見積りが困難な場合，実行時に EC2 インスタンスのリソース不足を検知し，EC2 インスタンスを追加することで，処理時間を短縮できることを確認した。

5. おわりに

パブリッククラウド上のスケールアウト型 DB に，オンプレミスのストレージボリュームを割り当て，ネットワーク転送量および CPU 使用率に基づき DB をスケールし，アクセス先を事前コピーしたストレージボリュームに切り替えるオートスケール方式を提案した。OLAP ベンチマークで本方式を適用した結果， DB 無停止でスケールアウトでき，インスタンス数を 2 倍にすると，一連のクエリ処理時間は 62%まで短縮できた。

商標について

参考文献

[1] Citus Documentation, http://docs.citusdata.com/en/v10.0/
[2] Jimi Sanchez, “A Review of Star Schema Benchmark”, arXiv:1606.00295v1[cs.DB] 1 Jun 2016, https://arxiv.org/pdf/1606.00295.pdf