Galera Clusterセグメント分割技法を用いた広域OpenStack Keystoneの性能改善

勝野 恭治[1,a)]　高橋 ひとみ[1,b)]　小野寺 民也[1,c)]

概要：クラウド基盤を構築するオープンソースソフトウェアのデファクトスタンダートとなりつつあるOpenStackは、WANで接続された複数のデータセンター間に渡ったクラウド基盤の構築にも利用されつつある。そのような広域OpenStackを構築する際の問題の一つとして、WAN上で行われるMySQLクラスターのデータ複製処理による、OpenStack Keystoneサービスの性能劣化が挙げられる。
本論文では、MySQLクラスターとして幅広く用いられている、Galera Clusterの最新版が搭載する機能の一つである、クラスターセグメント分割技法に着目する。広域OpenStack Keystoneエミュレーション環境を構築し、クラスターセグメント分割技法が広域OpenStack Keystoneの性能改善に与える影響をベンチーマークを用いて評価する。

1. はじめに

1.1 背景

OpenStack[1]は、クラウド基盤を構築するオープンソースソフトウェアのデファクトスタンダートとなりつつある。OpenStackは当初、単一データーセンター内のクラウド基盤構築に利用されていたが、近年、WANで接続された複数のデータセンター間に渡った広域クラウド基盤の構築にも利用されつつある。

WAN上に構築された広域OpenStackは、従来のLAN上に構築されたOpenStackと比べて、利用ユーザー数やサーバー数、ネットワーク品質などの違いから、特有の問題が発生する。その中の一つに広域OpenStack Keystoneの性能低下がある。

OpenStack Keystoneはユーザー認証・認可管理機能を担当する。各データセンターで独立してOpenStack Keystoneを運営するデータセンター独立型OpenStack Keystoneでは、ユーザーIDがデーターセンター毎に異なり、利便性が悪くなる。データセンター独立型OpenStack Keystoneの例を、図1に示す。データセンターA、B、Cそれぞれに独立したOpenStack Keystone A, B, Cが用意されているため、ユーザーは、katsuno-A, katsuno-B, katsuno-Cといった3つのユーザーIDをデータセンター毎に使い分けなければならない。

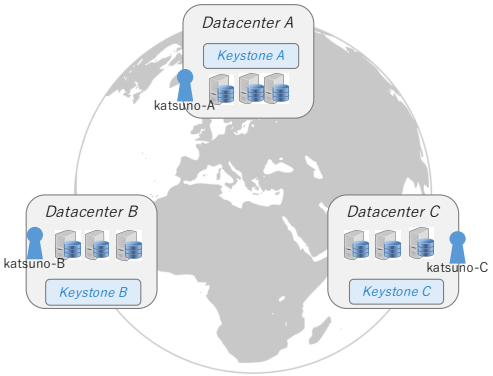

図 1　データセンター独立型 OpenStack Keystone

同一ユーザーIDでどのデータセンターにもあるOpenStackにアクセスできるようにするための方法の一つに、OpenStack Keystoneを各データセンターでそれぞれ独立して運営するのではなく、データセンター共有型OpenStack Keystoneとしてデータセンター共通の共有サービスとして運営する方法がある。データセンター共有型OpenStack Keystoneを実現するために、ユーザー情報などを管理するために使われているMySQLデータベースをデータセンター間でクラスター化する手法がよく用いられている。MySQLクラスターによるデータセンター共有型OpenStack Keystoneの例を、図2に示す。データセンターA、B、Cそれぞれには MySQLクラスターによってデーターベースを共有しているOpenStack Keystoneが用意され

[1] IBM 東京基礎研究所、東京都中央区日本橋箱崎町 19-21
IBM Research – Tokyo, 19-21 Nihonbashi, Hakozaki-cho, Chuo-ku, Tokyo, Japan
a) katsuno@jp.ibm.com
b) hitomi@jp.ibm.com
c) tonodera@jp.ibm.com

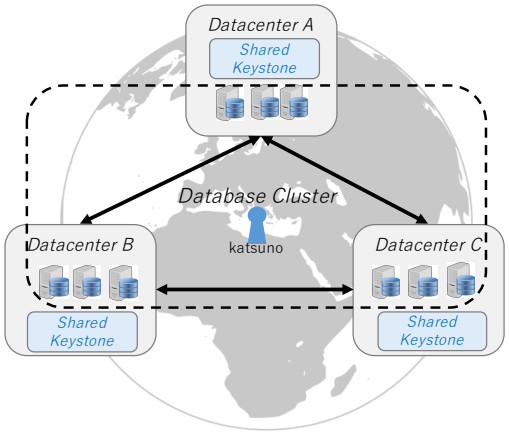

図 2 MySQL クラスターによるデータセンター共有型 OpenStack Keystone

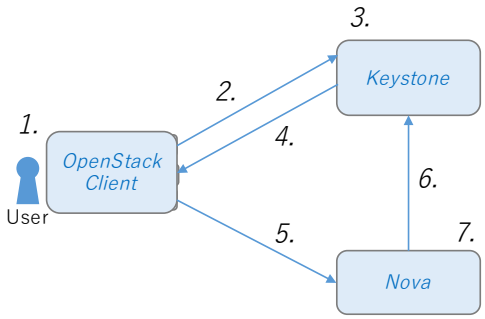

図 3 Nova を利用する際の Keystone によるユーザー認証・許可処理

ているため、ユーザーは、katsuno といった単一のユーザー ID でデータセンター A、B、C それぞれの OpenStack Keystone にアクセス可能となる。

データセンター間は WAN で接続されており、データベース間のデータ同期作業は WAN 上で行われる。WAN はネットワーク遅延やパケットドロップ率といったネットワーク品質が LAN と比べて劣化する。そのため、WAN 上で構築される MySQL クラスターの性能、特にデータ複製処理の性能が低下し、その結果として、OpenStack Keystone の性能に影響を及ぼすことが懸念される。

1.2 本論文の貢献

本論文では、MySQL クラスターとして幅広く用いられている、Galera Cluster[3] の最新版が提供する機能の一つである、クラスターセグメント分割技法に着目する。広域 OpenStack Keystone エミュレーション環境を構築し、OpenStack 公式ベンチマークプログラムによる OpenStack Keystone の性能測定を通じて、クラスターセグメント分割技法が OpenStack Keystone の性能改善に与える影響を検証する。

本論文の構成は次の通りである。第 2 章で OpenStack Keystone、第 3 章で Galera Cluster の概要についてそれぞれ述べる。第 4 章で広域 OpenStack Keystone エミュレーション環境を構築し、第 5 章でエミュレーション環境上で Keystone の性能評価を行う。最後に、第 6 章で結論と今後の課題について述べる。

2. OpenStack Keystone

OpenStack[1] は、Rackspace と NASA によって 2010 年に立ち上げられた、オープンソースのクラウド基盤である。OpenStack は機能別にモジュール化されている。例えば、仮想サーバー管理機能は Nova、ネットワーク管理機能は Neutron、Web ダッシュボード機能は Horizon、によってそれぞれ提供される。

Keystone[2] は、ユーザー認証・認可管理機能を担当する。Keystone は認証・許可処理を一元的に行い、OpenStack の各モジュールに対して提供する。Nova を利用する際の Keystone によるユーザー認証・許可処理を図 3 に示す。

(1) ユーザーは OpenStack クライアント（OpenStack コマンド、Horizon）に自分のユーザー ID とパスワードを入力する
(2) OpenStack クライアントは、ユーザー ID とパスワードを添付して認証処理を Keystone に要求する
(3) Keystone は、ユーザー ID とパスワードを元に認証処理を行う
(4) Keystone は、認証が成功した場合、認証トークンを発行し、認証トークンと Nova の URL を OpenStack クライアントに送信する
(5) OpenStack クライアントは、認証トークンを添付して Nova URL にアクセスする
(6) Nova は OpenStack クライアントから送られてきた認証トークンが有効かどうかを Keystone に確認する
(7) Nova は、Keystone の確認が取れたら、OpenStack クライアントが要求した処理を実施する

3. Galera Cluster

Galera Cluster[3] は、Codership 社によって開発された、MySQL データベース及び MySQL 互換データベースのための、オープンソースのデータベースクラスターである。

マルチマスター同期レプリケーション方式を採用しており、全てのノードがアクティブでかつマスターとなり、クラスター内のどのノードに対しても読み込み・書き込み処理が可能となっている。データベース利用側は通常の MySQL と同様にアクセスができることもあり、幅広く用いられている。

Galera Cluster では、マルチマスター同期レプリケーションを実現するために、同一クラスター内のノードは他の全てのノードとデータ複製処理を行う。そのため、クラスターに参加しているノードが WAN 上に分散している場

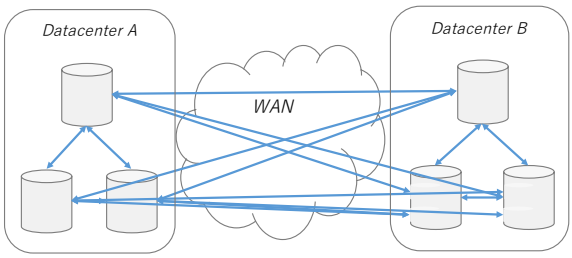

図 4 WAN 上の Galera Cluster のデータ複製処理

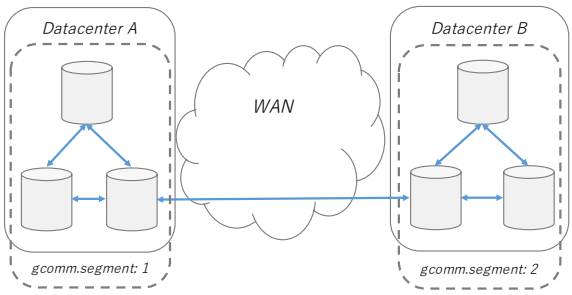

図 5 セグメント分割機能を用いた場合の WAN 上の Galera Cluster のデータ複製処理

合、WAN 上で大量のデータ複製処理が実施され、性能低下が懸念される（図 4）。

本問題を解決するために、Galera Cluster はバージョン 3 からセグメント分割機能を導入している。各ノードに対してセグメント番号を割り当て（デフォルトの値は 0）、同一セグメント番号のノードとは通常のデータ複製処理を行い、異なるセグメント番号のノードとは、基本的に同一セグメント内で自動的に選出された特定のノードのみがデータ複製処理を行う（図 5）。本機能によって WAN 上で処理されるデータ複製を大幅に削減でき、性能低下を抑えることができると期待されている。

4. 広域 OpenStack Keystone エミュレーション環境の構築

広域 OpenStack Keystone エミュレーション環境を図 6 に示す。本環境は、物理サーバー IBM Flex System x240 (Inte Xeon E5-2670 2.60GHz 32 CPU Cores, 256G Memory, Ubuntu 14.04.4 LTS, VirtualBox 4.3.36) 上の 9 個の仮想サーバー（2 CPU Cores, 4G Memory, CentOS 7.2）から構成されている。

各仮想サーバーの仕様を次に示す。

4.1 データベースクラスターサーバー

6 個のデータベースサーバー：DB1, DB2, DB3, DB4, DB5, DB6 によって、Galera Cluster が 1 セット構築されている。DB1, DB2, DB3 はデータセンター A 側で、DB4, DB5, DB6 はデータセンター B 側にあると想定している。MySQL データベースは、MySQL 互換の MariaDB バージョン 10.0.24 を利用した。

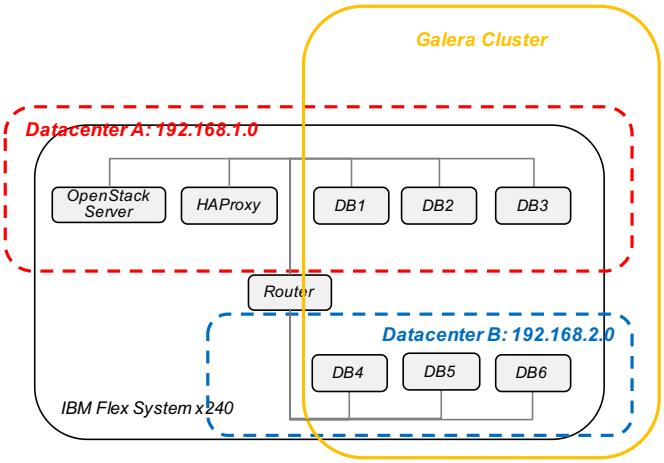

図 6 広域 OpenStack Keystone エミュレーション環境

4.2 ロードバランシングサーバー

Galera Cluster はデータベースサーバーが数台サービス提供不能になったとしても、サービス提供可能なサーバーからデータベースサービスを提供できる。しかし、OpenStack Keystone からデータベースサーバーを指定する際には単一の IP アドレスが利用されるため、該当サーバーがサービス提供不能になると、Galera Cluster 自体が動作していても、OpenStack Keystone からはデータベースサービスが利用不能になったと判断される。

本問題を解決するために、OpenStack サーバーとデータベースクラスターサーバーの間に、ロードバランシングサーバーを設置する。ロードバランシングサーバーとして HAProxy[4] バージョン 1.5.4 を利用し、ロードバランシングアルゴリズムはラウンドロビンを選択した。HAProxy はデータセンター A 側で動かし、HAProxy 配下のサーバーは、データセンター A 側のデータベースサーバーである、DB1, DB2, DB3 となる。

4.3 OpenStack サーバー

OpenStack サーバー上には、OpenStack の 12 番目のリリースである Liberty[5] にバンドルされている Keystone と、OpenStack 公式ベンチマークプログラムの一つである Rally[6] が動作している。

Rally は、OpenStack に対して、サーバー作成・削除といった典型的な利用シナリオを繰り返し実行して、平均実行時間などを取得することができる。Rally は利用シナリオの種類にかかわらず、ベンチマーク開始前に、Keystone、Nova、Neutron の稼働状況をチェックする。そのため、Keystone サーバー以外に、必要最小限の Nova、Neutron サービスとして neutron-server と openstack-nova-api も動作している。

Rally による Keystone ワークロードの生成（例：ユーザー作成）はデータセンター A 側で行われ、データベースクラスターサーバーへのアクセスはデータセンター A 側か

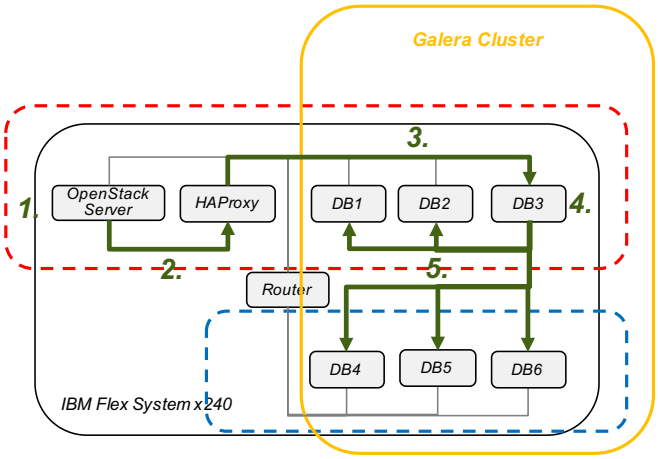

図 7 Rally によるデーターフロー

らの単方向となる。データーフローを図 7 に示す。

(1) Rally は Keystone ワークロード を生成し、OpenStack Keystone に送信する。

(2) OpenStack Keystone は Keystone データベースサーバーとして登録している、HAProxy にデータベース処理を要求する。

(3) HAProxy はラウンドロビンアルゴリズムに基づいてデータベース処理を行うデータベースサーバーを決定し、データベース処理要求をフォワードする。

(4) HAProxy からデータベース処理要求を依頼されたデータベースサーバーは指定されたデータベース処理を行う。

(5) データベース処理内容がデータベースへの書き込みの場合、データベースサーバーは他のサーバーに対して更新内容を反映させる。

4.4 IP ルーター

データセンター A とデータセンター B をそれぞれエミュレーションする IP ネットワーク：192.168.1.0 と 192.168.2.0 を相互接続するために、IP ルーティングを行うサーバーを設置した。net.ipv4.ip_forward の値を 1 にセットし、IP ルーティングテーブルを設定することで、192.168.1.0 ネットワーク上の DB1, DB2, DB3 と 192.168.2.0 ネットワーク上の DB4, DB5, DB6 が通信できるようにした。

データセンター A とデータセンター B が広域 WAN で接続されていることをエミュレーションするために、データセンター A の IP ネットワーク：192.168.1.0 と、データセンター B の IP ネットワーク：192.168.2.0 を相互接続する、IP ルーター上で WAN 特有のネットワーク遅延を仮想的に実現する必要がある。

本論文では、Linux tc (Traffic Control)[7] によってネットワーク遅延を仮想的に実現した。Linux tc は、Linux カーネル内のパケットスケジューラーを操作してネットワークインターフェース単位でネットワーク遅延やパケットドロップ率などを発生させることができるユーティリティコマンドである。Linux TC を IP ルーティングサーバーのネットワークインターフェースに適用して、IP ネットワーク：192.168.1.0 と 192.168.2.0 の間に特定時間のネットワーク遅延を発生させた。

5. 広域 OpenStack Keystone の性能評価

5.1 性能評価パラメーター

5.1.1 Rally ベンチマークシナリオ

Rally は様々な利用シナリオを提供している [8]。本論文では、Keystone に関する利用シナリオの中から、ユーザー作成を行う：create-user、とユーザー作成及びリスト表示を行う：create-and-list-users、の 2 種類を用いた。なお、Rally はユーザーが所属するグループを示すテナントも自動的に作成する。

5.1.2 Rally ベンチマーク設定変数

Rally の各テンプレートには、同時リクエスト数を設定する concurrency、リソース作成総数（上記シナリオの場合は、ユーザー）を設定する times、という項目がある。本論文では、concurrency: 5、times: 20、と設定した。

5.1.3 OpenStack ユーザー・テナント数

本論文が想定する広域 OpenStack Keystone 環境は大勢のユーザーが使用するため、大量の OpenStack ユーザーとユーザーが所属するグループを示す OpenStack テナントが事前に作成されていると考えられる。本論文では、事前作成されている OpenStack ユーザー数を 1000、OpenStack テナント数を 100 とした。

5.1.4 Keystone Revocation Event 数

OpenStack Keystone は次のイベントが発生すると、Keystone データベース内の revocation_event テーブルにエントリーを作成する。

- 利用者が OpenStack Horizon 上で所属テナントを変更した
- 利用者が OpenStack Horizon 上でログアウトを行った
- OpenStack ユーザーが削除された
- OpenStack ユーザーロールが削除された
- OpenStack ユーザーがプロジェクトから脱退した
- Keystone アクセストークンが REST API アクセスによって故意に無効にされた

revocation_event テーブルのエントリー数は Keystone のパフォーマンスに影響を与えることが報告されているため [9]、OpenStack Rally ベンチマークを実行する前のエントリー数を一定にする必要がある。本論文では、REST API アクセスによって Keystone アクセストークンの作成・削除を繰り返すスクリプトを開発し、OpenStack Rally ベンチマークを実行する直前に revocation_event テーブルのエントリーを全て削除し、本スクリプトを実行して、revocation_event テーブルのエントリー数が 50 になるよう

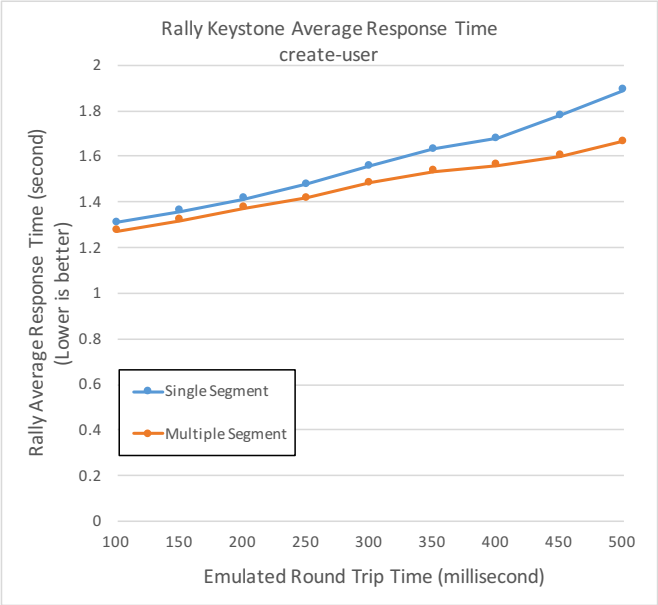

図 8 Rally Keystone create-user の平均返答時間

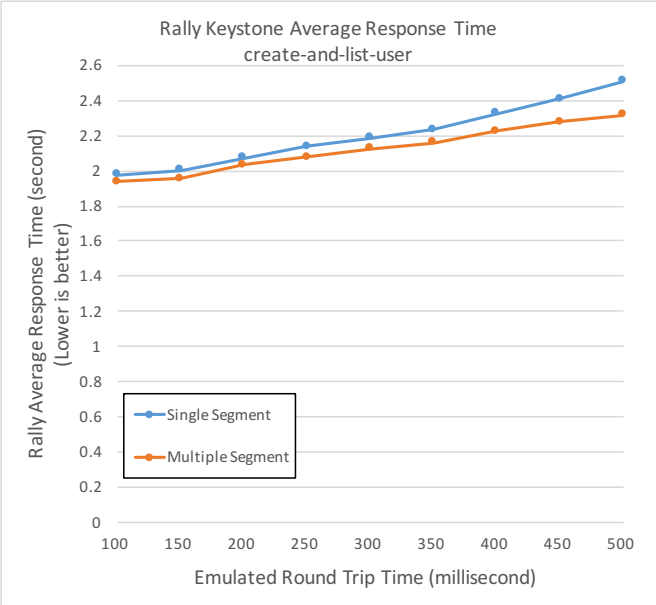

図 9 Rally Keystone create-user-and-list の平均返答時間

にした。

5.1.5 WAN ネットワーク遅延

Linux tc によって発生させる WAN ネットワーク遅延は、Verizon Enterprise Solutions がニューヨーク、ロンドン間のラウンドトリップ時間としてサービス保証している 90ms 以下という値を基準値として [10]、100ms から 500ms まで 50ms 刻みで変化させた。

5.2 性能評価結果

Rally Keystone create-user テンプレートと create-user-and-list テンプレートの平均返答時間結果を図 8 と図 9 に、Galera Cluster セグメント分割機能による Rally Keystone 平均返答時間の改善率を図 10 に、それぞれ示す。

5.3 評価結果の考察

5.3.1 create-user テンプレートの平均返答時間

図 8 と図 10 より、ラウンドトリップ時間が 100ms から 200ms の間は、Galera Cluster セグメント分割機能による Rally Keystone 平均返答時間の改善率が 3 パーセント程度であるが、200ms より大きくなると改善率は大きくなり、500ms の時は約 12 パーセントになり、100ms の時と比べて 6 倍向上する。

create-user テンプレートが実行される度に、Galera Cluster 内にデータが作成され、データベースへの書き込みが発生する。データが作成されたデータベースサーバーは他のデータベースサーバーに対してデータ複製処理を行う。データセンター A とデータセンター B の間のラウンドトリップ時間が増加するに従って、データレプリケーション処理に時間がかかり、データセンター間のデータレプリケーション回数が Rally Keystone 平均返答時間に影響を

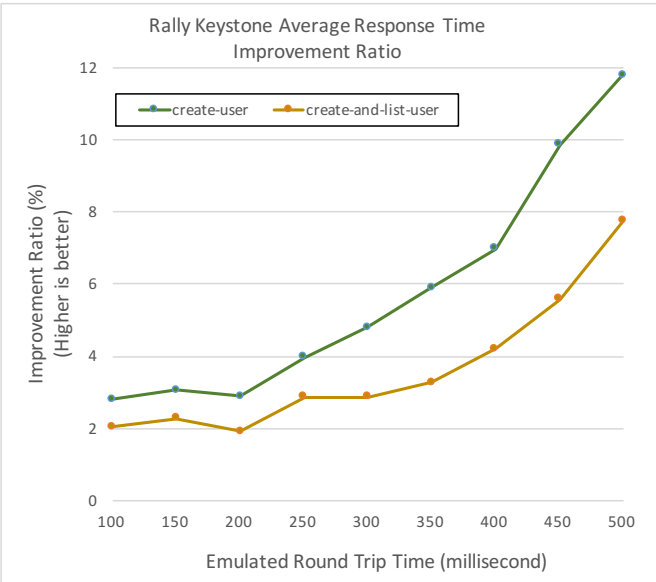

図 10 Galera Cluster セグメント分割機能による Rally Keystone 平均返答時間の改善率

与えることとなる。

図 6 に示した今回構築した広域 OpenStack Keystone エミュレーション環境では、図 4 と図 5 の比較より、Galera Cluster セグメント分割機能を使わない単一セグメントの場合（図 4）は、Galera Cluster セグメント分割機能を使う複数セグメントの場合（図 5）と比べて、データセンター間のデータ通信回数が 9 倍少ない。そのため、図 10 が示す通り、ラウンドトリップ時間が増えるに従って、Rally Keystone 平均返答時間の改善率が向上する。

5.3.2 create-user-and-list テンプレートの平均返答時間

図 9 と図 10 より、ラウンドトリップ時間が 100ms から

200ms の間は、Galera Cluster セグメント分割機能による Rally Keystone 平均返答時間の改善率が 2 パーセント程度であるが、200ms より大きくなると改善率は大きくなり、500ms の時は約 8 パーセントになり、100ms の時と比べて 4 倍向上する。さらに、create-user テンプレートの場合と同様に、ラウンドトリップ時間が増えるに従って、Rally Keystone 平均返答時間が向上する。

Galera Cluster セグメント分割機能による Rally Keystone 平均返答時間の改善率が、create-user テンプレートの場合と比べて低い結果となった。これは、テンプレートが実行される際に行われるデータベースアクセスの種類が原因だと考えられる。create-user テンプレートの場合は、ユーザーの作成が行われ、データベースへのアクセスは書き込みのみである。それに対して、本 create-user-and-list テンプレートの場合は、ユーザーの作成とユーザーのリスト表示が行われ、データベースへのアクセスは書き込みと読み込みとなる。データベースへの書き込みはデータレプリケーションが発生し、Galera Cluster セグメント分割機能は効果を発揮する。しかし、データベースへの読み込みはデータレプリケーションが発生しないため、Galera Cluster セグメント分割機能は効果を発揮しないと考えらえる。

6. おわりに

WAN で接続された複数データーセンター間で OpenStack Keystone サービスを構築する際の性能ボトルネックの一つに、MySQL クラスターにおける WAN 上で行われるデータ複製処理がある。本論文では、最新の Galera Cluster が搭載する機能の一つである、クラスターセグメント分割技法に着目した。広域 OpenStack Keystone エミュレーション環境を構築し、クラスターセグメント分割技法が Keystone の性能に与える影響をベンチーマークを用いて評価した。その結果、クラスターセグメント分割技法が、WAN ネットワーク遅延による性能低下を 2 パーセントから最大 12 パーセント抑えられることを実証した。

本論文では、2 つのデータセンターサイトで、データベースに障害が起きていない状態において評価を行ったが、データベースに障害が起きている状態や障害から回復している状態における評価を行うことは有益である。また、WAN で接続された地点間（例：東京、ニューヨーク）で実際に評価を行い、本論文のエミュレーション環境下における評価結果と比較することも有益である。

参考文献

[1] Openstack. http://www.openstack.org.
[2] Openstack keystone. http://docs.openstack.org/developer/keystone/.
[3] Galera cluster. http://galeracluster.com.
[4] Haproxy. http://www.haproxy.org.
[5] Openstack liberty. https://www.openstack.org/software/liberty/.
[6] Rally. https://wiki.openstack.org/wiki/Rally.
[7] Linux tc. http://lartc.org/manpages/tc.txt.
[8] Rally scenarios configuration samples. https://github.com/openstack/rally/tree/master/samples/tasks/scenarios.
[9] Keystone token revocations cripple validation performance. http://www.mattfischer.com/blog/p=672.
[10] Verizon ip latency statistics. http://www.verizonenterprise.com/about/network/latency/.