SOA に基づくクラウド間連携アーキテクチャの提案

2007MI083　岩本　勝也　2007MI109　小島　弘誉　2007MI186　大倉　奨貴

指導教員　青山　幹雄

1. はじめに

近年，クラウドコンピューティング(以下クラウドと略記)は，コンピュータの新しい利用形態として注目されている[5]. クラウド間やクラウド，オンプレミス間の連携の必要性が増しているが，クラウドプロバイダによりサービス提供形態が異なるため，クラウド間の連携が困難となっている. 本稿では，クラウド間連携を実現するアーキテクチャを提案する.

2. 研究課題

クラウド間連携には疎結合な連携が必要である. 連携する際にクラウド毎にメッセージの振舞いと型が不整合な点が課題である. よって，複数のクラウド間で振舞いと型の整合をとる必要がある. 以下に三点の研究課題を示す.

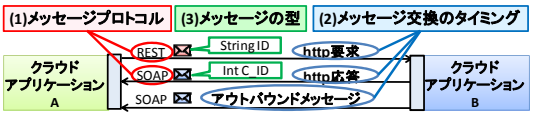

図 1　不整合なメッセージの振舞いと型

(1) メッセージプロトコル

SOAP，REST など送受信可能なメッセージプロトコルが異なる.

(2) メッセージ交換のタイミング

メッセージを送受信するタイミングが異なる.

(3) メッセージの型

データシンタックスとデータセマンティクスが異なる. データシンタックスとはデータの形式を指し，データセマンティクスはデータの属性名と階層構造を指す.

3. 関連研究

現状のクラウドの連携方法として，コネクタを用いた連携方法とビジネスプロセスを用いてシステム全体の振舞いを集中制御する連携方法が提案されている.

3.1. コネクタを用いた連携

SaaS(Software as a Service)毎に，専用のコネクタを用いることで連携を実現する[6]. 連携するアプリケーション間に対応したコネクタが必要となるため，連携の組合せの数だけコネクタを開発する必要がある.

3.2. SOA に基づく SaaS 連携アーキテクチャの提案

SOA(Service-Oriented Architecture)を用いたサービス間の連携に着目し，オンプレミス，SaaS 間の連携にビジネスプロセスを用いて実現するアーキテクチャが提案されている[3]. オンプレミス，SaaS 間を疎結合にすることで，SOA の性質を満たし，拡張性の高いシステム構成を実現する.

4. アプローチ

コレオグラフィ連携モデルと SOA に着目した.

4.1. コレオグラフィ連携モデルの適用

コレオグラフィとは，並列的な対話型モデルである. 集中制御を持たず，各 Web サービスが実行を制御する(図 2). コレオグラフィの適用により特定の制御方法に限定されない連携が可能となる[2].

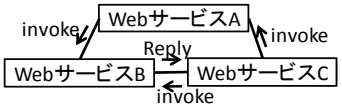

図 2　コレオグラフィ連携モデル

4.2. SOA の適用[4]

SOA は以下のような観点から適用する.

(1) クラウドブローカの機能をサービスとして捉え，組換え可能な方法を構築方針とする.

(2) SOA の実現技術であるパブリッシュ/サブスクライブアーキテクチャを適用することで，不特定多数のクラウド間でメッセージ交換を実現する.

5. 提案アーキテクチャ

本稿では，コレオグラフィ連携モデルと SOA の概念を適用し，N 対 N 連携モデルを実現するアーキテクチャを提案する. N 対 N 連携とは不特定多数のクラウド，オンプレミスが共にメッセージの送受信が可能な連携である(図 3).

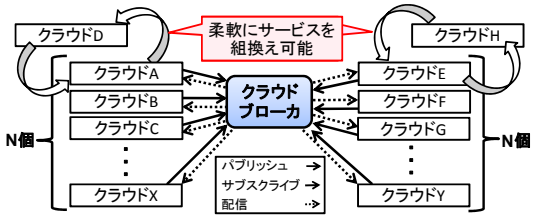

図 3　N 対 N 連携モデル

以下に提案するアーキテクチャの具体的な説明をする.

5.1. クラウドブローカアーキテクチャ

クラウドブローカとはクラウド間の振舞いの整合をとり，連携を可能とする仲介サービスである.

クラウドブローカにパブリッシュ/サブスクライブアーキテクチャを適用し，メッセージのフィルタリングモデルとしてタイプベースフィルタリングを適用する.

パブリッシュ/サブスクライブは，非依存性特徴により，振舞いを整合し，非同期にN対N連携を可能とする．

タイプベースフィルタリングモデルは，階層構造の型を定義することにより，型を整合し，不特定多数のクラウド間連携を可能とする．

5.1.1. パブリッシュ/サブスクライブアーキテクチャの適用

パブリッシュ/サブスクライブとは，メッセージを要求するサブスクライバが，事前にサブスクリプションしておくことで，ブローカに対してパブリッシャの送信したメッセージが，サブスクライバへ送信されるアーキテクチャである[1].

5.1.2. パブリッシュ/サブスクライブの非依存性特徴

パブリッシュ/サブスクライブは，以下の三点の特徴によりクラウド間の疎結合な連携を実現する．

(1) 空間的分離

送受信者は共に相手のエンドポイント情報を保持せずとも，メッセージの交換が可能である．

(2) 時間的分離

送受信者は共に相手のライフラインが切断されていても，メッセージの交換が可能である．

(3) 同期的分離

複数のアクティビティを実行している間，非同期的にメッセージの交換が可能である．

5.2. タイプベースフィルタリングモデルの適用

タイプベースフィルタリングとは，型の階層構造モデルを定義し，フィルタリングを行うモデルである(図4).

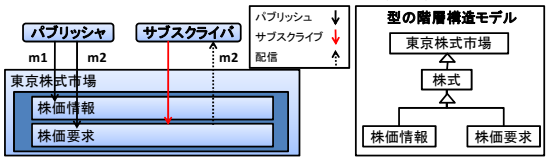

図4 タイプベースフィルタリング

以下にタイプベースフィルタリングモデルの特徴を他のフィルタリングモデルと比較し説明する．

5.2.1. タイプベースフィルタリングモデルの特徴

パブリッシュ/サブスクライブには，三つのフィルタリングモデルがある．トピックベースフィルタリング，コンテンツベースフィルタリング，タイプベースフィルタリングである．以下はそれぞれのモデルの特徴を示す(表1).

表1 フィルタリングモデルの比較

モデル	フィルタリングの種類	フィルタリングの条件
トピック	静的	属性
コンテンツ	動的	属性値
タイプ	静的/動的	型

タイプベースフィルタリングは属性を階層構造で定義した型での静的なフィルタリングと，属性値での動的なフィルタリングが可能である．フィルタリングの条件が属性をまとめた型であるため，パブリッシャは属性の違いを意識せず送信できる．サブスクライバはクラウドブローカが定義した型と属性値を用いてサブスクリプションできる．よって，フィルタリングの柔軟性が高い．

5.3. 振舞いと型の整合をとるアーキテクチャ

SOA に基づき機能をサービスとして分離することで，研究課題で挙げた三点の課題を解決するアーキテクチャを以下で示す．

5.3.1. タイミングの整合

メッセージキューを適用することで，異なるメッセージ交換のタイミングの整合をとる(図5)．クラウドAから送られたメッセージをメッセージキューに登録する．その後クラウドBがメッセージを受信可能なタイミングで要求し，受信する．

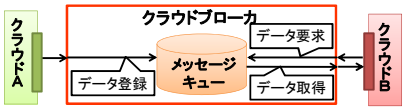

図5 タイミングの整合をとるアーキテクチャ

5.3.2. メッセージプロトコルの整合

クラウドブローカにメッセージが送信される際に，プロトコルサービスでSOAPまたはRESTに変換することで異なるメッセージプロトコルの整合をとる(図6). また，プロトコル変換されたメッセージは，クラウドブローカで処理されたのち，受信側に対応するメッセージプロトコルに変換する．

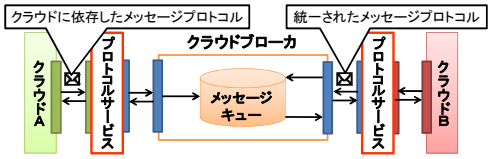

図6 プロトコルの問題を解決するアーキテクチャ

5.3.3. メッセージの型の整合

型変換サービスを仲介することで，データシンタックス，データの属性名の整合をとり，型付けサービスによりデータの階層構造の整合をとる．

(1) データシンタックスの整合

型変換サービスを仲介し，各クラウドのデータ型を標準データ型に変換することで，データシンタックスの整合をとる．例としてIntegerを基準にしたID型，Stringを基準にしたC_ID型と言う二つのユーザ定義型を，Stringを基準にしたCustomer_IDと言うクラウドブローカの標準データ型に変換し整合をとる．

(2) データの属性名の整合

型変換サービスを仲介し，各クラウドのデータ型の属性名を標準データ型に対応した属性名に変換することで整合をとる．例えば，クラウドAのIDと言う属性名に対して，顧客のIDを示す場合はConsumer_IDという属性名に変換するなど，クラウドブローカの標準データ型に対応した属性名に変換することで整合をとる(図7).

(3) データの階層構造の整合

型付けサービスに標準データ型の階層データモデ

ルを定義することで，データの階層構造の整合をとる．

例えば，クラウド A，B の異なる階層構造の顧客データを，クラウドブローカで定義した標準データ型の階層データモデルにより整合をとる(図 8)．

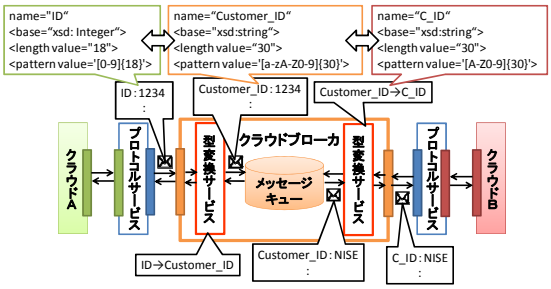

図 7 データシンタックスの整合

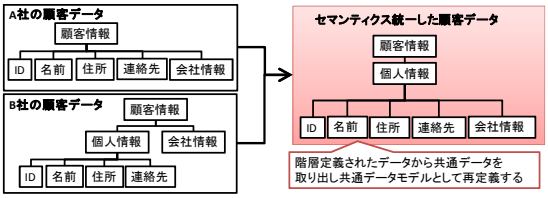

図 8 データセマンティクスの整合

5.4. クラウド間連携アーキテクチャ

タイプベースフィルタリング型パブリッシュ/サブスクライブを用いたクラウド間連携アーキテクチャを提案する(図 9)．

提案アーキテクチャはメッセージ交換のタイミング，メッセージプロトコル，メッセージの型の整合をとる．また，コレオグラフィと SOA の概念の導入により，クラウド間の疎結合な連携を実現し，不特定多数のクラウド間連携を可能とする．

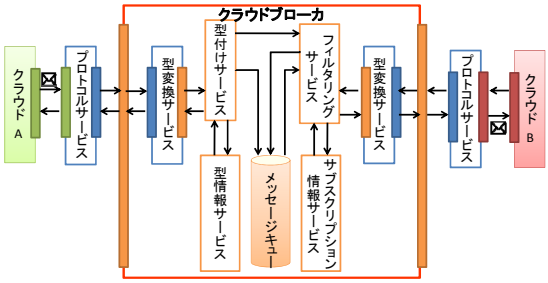

図 9 提案アーキテクチャ

クラウドブローカを実現するため，型付けサービス，型情報サービス，フィルタリングサービス，サブスクリプションサービス，メッセージキューサービスを提案する．型付けサービスはメッセージを型情報に基づきメッセージキューに登録する．フィルタリングサービスがサブスクリプション情報に基づき，キューからメッセージを取得する．これによりパブリッシュ/サブスクライブアーキテクチャを実現する．

6. プロトタイプ

提案アーキテクチャの妥当性を確認するためプロトタイプを開発した．

6.1. プロトタイプの仕様

クラウド A を Salesforce.com，クラウド B を GoogleMaps とし，クラウドブローカによって振舞いと型の整合をとる．Salesforce.com と GoogleMaps ではメッセージ送信のタイミングが異なる．以下に各メッセージ送信方法を説明し，プロトタイプ全体の構成図を図 10 に示す．

(1) Salesforce.com のアウトバウンドメッセージ

非同期にメッセージを送信する機能である．ワークフロールールを設定することでユーザの任意のタイミングでメッセージを送信することができる．

(2) GoogleMaps のリクエスト/レスポンス

サーバにリクエストを送信することでデータをレスポンスとして受信する仕組みである．リクエストを送信したタイミングでメッセージを受信できる．

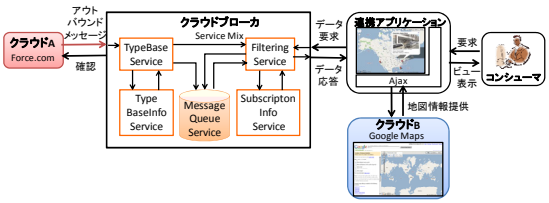

図 10 プロトタイプの構成

プロトタイプの機能を図 11 に示す．

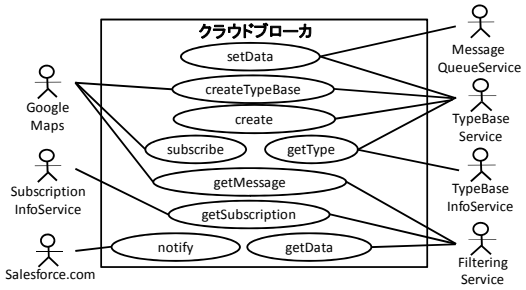

図 11 プロトタイプのユースケース図

プロトタイプの振舞いを図 12 に示す．

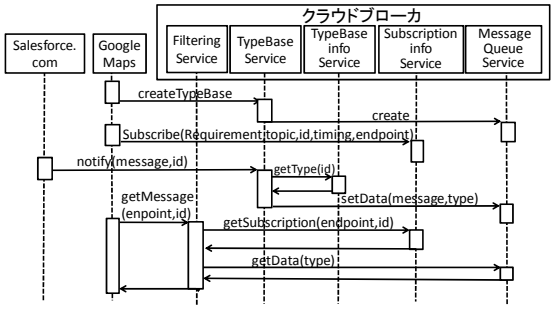

図 12 シーケンス図

6.2. クラウドブローカのプラットフォーム

クラウドブローカではプラットフォームとして ServiceMix を利用する．

ServiceMix とは JBI(Java Business Integration)準拠のオープンソース ESB(Enterprise Service Bus)であり，多種多様なコンポーネントが提供されている．

(1) Apache ActiveMQ
　メッセージブローカとして機能するコンポーネントである．キューを用意してメッセージの登録と取得を非同期に行う．

(2) Apache CXF
　Web サービスの機能を提供するフレームワークである．パフォーマンスおよび拡張性に優れたサービスを作成することが可能である．

6.3. プロトタイプの実装

型付けサービス，型情報サービス，フィルタリングサービス，サブスクリプション情報サービスは CXF コンポーネントを利用し，メッセージキューは ActiveMQ コンポーネントを利用する．CRM アプリケーションとクラウドブローカのメッセージキューサービスを実装した．

6.3.1. CRM アプリケーション(Salesforce.com)

Force.com を利用し CRM アプリケーションを開発した．システム管理者が顧客データを登録する際にアウトバウンドメッセージを送信するようにワークフロールールを設定した．

6.3.2. クラウドブローカ(ServiceMix)

クラウドブローカのメッセージキューサービスを実装した．ServiceMix 上に，サービスをバンドルとして実装した．

6.3.3. メッセージキューサービス(CXF，ActiveMQ)

メッセージキューサービスでは，SOAP 解析を行うために CXF の機能である WSDL2Java を利用し，スタブを生成する．また，解析したメッセージをキューに登録するために ActiveMQ を利用し，非同期にメッセージの登録と取得を行う．以下に実装した仕組みについて説明する．

(1) SOAP 解析: Salesforce.com から送信される SOAP に格納されている顧客データを取得する．

(2) メッセージ登録: 取得した顧客データをメッセージキューに登録する．メッセージに格納されているインスタンスの ID 毎にキューを生成し，メッセージが送信されたタイミングで登録する．

(3) メッセージ取得: キューに登録された顧客データを取得する．コンシューマからの HTTP リクエストを受信するタイミングでキューにアクセスし，データを取得する．

以下に開発したプログラムの実装結果を示す(表 2).

表 2 実装結果

成果物	言語	モジュール数	行数
プログラム	Java	1	101
メタデータ	XML	2	165

7. 評価・考察

提案アーキテクチャは疎結合な連携を実現した．コネクタと比較し，パブリッシュ/サブスクライブアーキテクチャの適用により，不特定多数のクラウド間連携を可能とし，クラウド間連携の課題を振舞いと型に分離して整合をとったため，疎結合な連携を実現した．各整合方法の評価を示す．

(1) 振舞いの整合
　メッセージプロトコルを SOAP/REST に対応することで整合をとり，メッセージ交換にメッセージキューを仲介することでタイミングの整合をとった．

(2) 型の整合
　データシンタックスは，より抽象度の高いデータ形式に変換し，データセマンティクスは共通のデータモデルに変換することで，整合をとった．
　振舞いの整合により不特定多数のクラウド間連携を可能とし，型の整合をすることで，組換え可能となり疎結合な連携を実現した．

8. 今後の課題

提案アーキテクチャの今後の課題を以下に挙げる．

(1) 提案アーキテクチャのプロトタイプの拡張
　プロトコルサービス，コネクタを含めた全体のアーキテクチャの動作確認をするためのプロトタイプ開発が必要である．また，型定義に用いる共通データモデルを定義し，型の整合を確認する必要がある．

(2) 応答時間によるオーバーヘッドの評価
　提案アーキテクチャは，クラウドブローカを仲介するためオーバーヘッドが発生する．応答時間を計測し，オーバーヘッドを評価する必要がある．

9. まとめ

クラウド間連携を実現するため，SOA に基づくクラウド間連携アーキテクチャを提案した．クラウド毎に異なるプロトコル，タイミング，メッセージの型を振舞いと型に分離し，整合をとるクラウドブローカを提案した．コレオグラフィ連携モデルに着目し，パブリッシュ/サブスクライブアーキテクチャを適用した．プロトタイプを作成し，評価した．

参考文献

[1] P. T. Eugster, et al., The Many Faces of Publish/Subscribe, ACM Computing Survey, Jun. 2003. pp. 114-131.

[2] M. B. Juric, Business Process Execution Language for Web Services, 2nd ed., PACKT, 2006.

[3] 近藤 洋介 他, SOA に基づく SaaS インテグレーションアーキテクチャの提案, 情報処理学会(第72回)全国大会, No 1ZC-1, Mar. 2010, pp. 383-384.

[4] D. Krafzig, et al., Enterprise SOA, Prentice Hall, 2005.

[5] 中田 敦 他, クラウド大全, 第2版, 日経 BP 社, 2010.

[6] 山谷 正己, 図解でわかる SaaS の全て, オーム社, 2009.