TOSCAを用いたハイブリッドクラウド上でのアプリケーション分散構築の自動化基盤アーキテクチャの提案と評価

M2012MM043　高木 裕之

指導教員　青山 幹雄

1. はじめに

クラウドコンピューティングの普及により，異なるクラウド基盤を連携するハイブリッドクラウドが注目されている．しかし，クラウド基盤ごとに API のシグネチャと抽象度，そして，構成管理方法が異なるため，ユーザの要求を満たすアプリケーションの分散構築が困難である．本研究では，アプリケーションの構成管理方法を統一し，クラウド基盤ごとの API を抽象化する．これにより，クラウド基盤によらないアプリケーションの配置を実現する．

2. 研究課題

2.1. クラウド基盤ごとに API が異なる

クラウドアプリケーションの分散構築を行うためには，仮想マシンの生成が必要である．しかし，クラウド基盤ごとに API のシグネチャとその抽象度が異なる．

2.2. アプリケーションの構成管理方法が異なる

アプリケーションの構成管理方法はクラウド基盤ごとに異なるため，構築したいアプリケーションの構成を異なる基盤ごとに表現する必要がある．近年，異なるクラウド基盤間の構成管理方法を統一する仕様の TOSCA が標準化されているが，アプリケーションの分散構築に対応していない．

3. 関連研究

3.1. ハイブリッドクラウド[4]

パブリックやプライベート環境に構築されたハイパーバイザを制御しアプリケーションを構築する研究が行われている．しかし，異なるクラウド基盤で構成されるハイブリッドクラウドへのアプリケーション構築は対応していない．

3.2. TOSCA[2]

異なるクラウド基盤間でアプリケーションの構成を統一的に扱うための標準記述仕様である．サーバ，OS，ミドルウェア，アプリケーションをノードとして扱い，それらの関係を定義することでトポロジを表現する．これを Service Template と呼ぶ．さらに構築手順を Plan として定義することでアプリケーションの自動構築を可能にする．また，TOSCA 処理環境の TOSCA コンテナを必要とする．

4. アプローチ

本研究では，TOSCA を用いて異なるクラウド基盤間でアプリケーションの構成管理方法を統一する．そして，ハイブリッドクラウドを管理し，クラウド基盤によらない仮想マシンの生成を実現する抽象 API を実装し，ブローカを構築する．また，配置する仮想マシンをクラウド基盤に割り当てる機能を持った TOSCA コンテナを構築する．これにより，Service Template から仮想マシンの情報を抽出し，配置可能なクラウド基盤を選択する．そして，TOSCA で定義された構築手順に従い，クラウド基盤によらない自動的なアプリケーションの配置を可能にする(**図 1**).

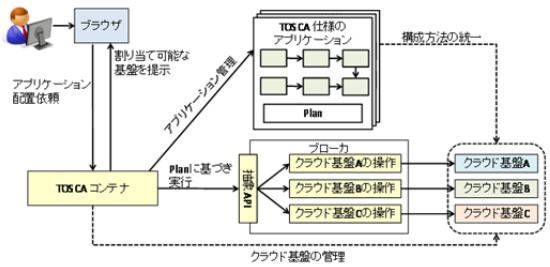

図 1 アプローチ

5. 分散構築自動化基盤アーキテクチャ

アプローチに基づきアプリケーションを異なるクラウド基盤へ自動的に分散構築する，分散構築自動化基盤アーキテクチャを提案する．

5.1. ブローカによるハイブリッドクラウドの管理と操作

分散構築自動化基盤を実現するためには，独立した異なるクラウド基盤を管理し，仮想マシンを自動的に生成する必要がある．仮想マシンを生成する抽象 API のシグネチャを XML Schema を用いて Service Template に定義する．そしてクラウド基盤を管理するブローカに抽象 API を実装する．ブローカを用いることによって，クラウド基盤の仕様変更や任意のクラウド基盤の追加に対する影響を局所化，透過的な仮想マシン生成を可能にする．抽象 API を用いた Plan をプロセスエンジンから実行することで，ハイブリッドクラウドにアプリケーションの自動的な分散構築を実現する．

5.2. TOSCA コンテナの拡張

TOSCA 仕様ではアプリケーションの分散構築が考慮されていない．分散構築を実現するため，参照実装の TOSCA コンテナを拡張する．Deploy Manager と Cloud DB を連携しクラウド基盤の情報をブラウザ上に表示する．仮想

マシンの構成単位である Tier の情報を Service Template から抽出して，配置可能なクラウド環境へ仮想マシンを割り当て，クラウドアプリケーションのインスタンスを生成する．

5.3. クラウド基盤の API

異なるクラウド基盤ごとに API の抽象度は異なっている(**図 2**)．例として，Amazon EC2[1]と OpenStack[3]の仮想マシン生成で操作可能なサーバプロパティを示す．この例では，OpenStack が低水準 API を持ち，仮想マシンの詳細を定義することが可能である．一方，Amazon EC2 では高水準 API を持ち仮想マシンの詳細な設定が不可能で，固定のインスタンスタイプを利用する必要がある．

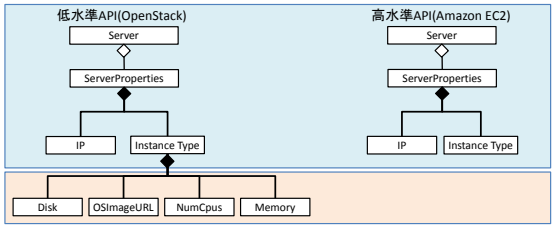

図 2 クラウド基盤の API

5.4. 抽象 API のデータモデル

クラウド基盤ごとに仮想マシンに対するAPIの抽象度が異なる．クラウド基盤間でインスタンスタイプを統一して高水準APIに統一すると，低水準APIを用いるクラウド基盤で詳細な設定ができない．高水準APIで利用するインスタンスタイプは，低水準なAPIの引数からユーザの要求を満たす最小のタイプを導出可能である．そのため，抽象APIのデータモデルは低水準APIを用いて抽象度を統一する．そして，抽象APIの内部でユーザの要求を満たす最小なタイプを導き，クラウド基盤のAPIに対する抽象度の差異を吸収する．

以下に作成した抽象 API のデータモデルを示す(**図 3**)．ユーザは仮想マシンのリソース要求を Service Template の Server と Tier のプロパティに定義することで，TOSCA から作成されるアプリケーションのインスタンスを生成する．また，スクリプトやノードの実体は Definition と共に CSAR (Cloud Service ARchive)ファイルに格納される．

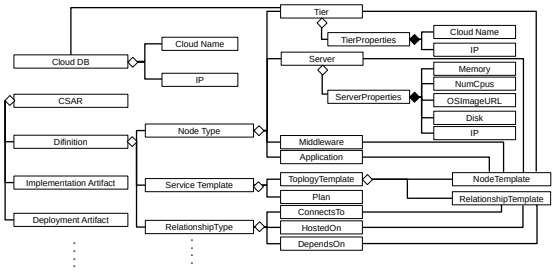

図 3 抽象APIのデータモデル

5.5. 抽象 API の階層構造

抽象 API は異なるクラウド基盤への仮想マシン生成を抽象化する．以下に抽象 API の階層構造を示す(**図 4**)．

(1) ファクトリメソッド：ユーザからインスタンス名，サーバ名を受け取り，Service Template から仮想マシンのリソース要求を取得する．

(2) 基盤ごとの仮想マシン生成：ファクトリメソッドから呼ばれるクラウド基盤ごとの仮想マシン生成 API．仮想マシンのリソース要求を受け取り，生成処理を行う．

(3) 詳細な操作：各クラウド基盤にアクセスし操作する API 群．クラウド基盤ごとに利用可能な複数の API を利用して，仮想マシンの生成に必要な操作を行う．高水準な API を持つクラウド基盤に対しては，予め定義したインスタンスタイプの中から，リソース要求を満たすタイプを導き，仮想マシンの構築を行う．また，低水準APIを持つクラウド基盤に対しては，リソース要求から仮想マシンのインスタンスタイプを定義し，仮想マシンの生成を行う．

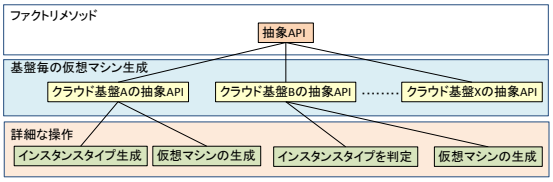

図 4 抽象 API の階層構造

5.6. 分散構築自動化基盤の機能

分散構築自動化基盤のユースケースを示す(**図 5**)．サービスカタログとは TOSCA 仕様のアプリケーションを提供するシステムである．

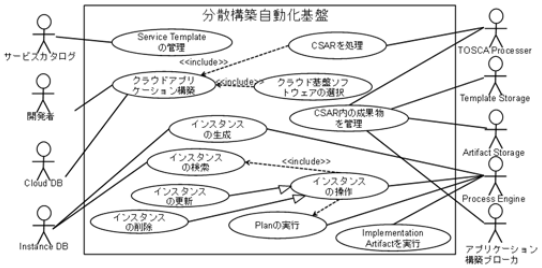

図 5 分散構築自動化基盤のユースケース

5.7. 分散構築自動化基盤の構造

分散構築自動化基盤の構造を示す(**図 6**)．CSAR のアップロードや仮想マシンのリソースを取得するときは，TOSCA コンテナの Container Façade の API を利用する．

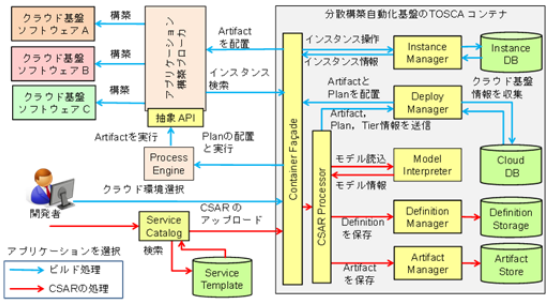

図 6 ブローカを用いた分散構築自動化基盤の構造

5.8. 分散構築自動化基盤の振舞い

分散構築自動化基盤を用いてクラウドアプリケーションのインスタンスを自動的に分散構築する振舞いを示す(**図 7**). 仮想マシンは Tier に割り当てたクラウド基盤へ生成される.

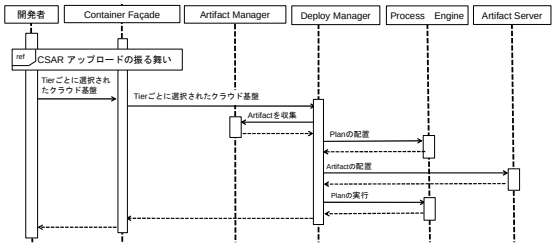

図 7 アプリケーション自動構築の振舞い

6. プロトタイプの構築

6.1.1. プロトタイプの目的

(1) 抽象 API の妥当性検証

異なるクラウド基盤からハイブリッドクラウドを構成する. そして抽象 API を実装し, クラウド基盤によらない仮想マシンの生成が可能かを検証する.

(2) ハイブリッドクラウドに対するサーバ構築の効率化検証

異なるクラウド基盤から構築したハイブリッドクラウドへ, TOSCA で記述したクラウドアプリケーションが分散構築可能か検証する.

6.2. プロトタイプの環境

プロトタイプのクラウド環境としてパブリッククラウドに Amazon EC2 を用い, プライベートクラウドに Openstack を用いた. この 2 つのクラウド基盤を用いる理由は仮想マシン生成に利用する API とその抽象度が異なり, Amazon EC2 がパブリッククラウドの IaaS 環境として最も普及していること, そして, OpenStack は TOSCA 仕様によって利用が推奨されているためである. この2つの異なる API の差異を吸収する抽象 API を実装し, アプリケーション構築ブローカによって管理する. また, プロトタイプではアプリケーションの構築をプロセスエンジンからではなく, クライアント側で構築手順を定義したスクリプトから実行する.

6.3. プロトタイプの機能

前提条件として, TOSCA コンテナにクラウドアプリケーションが登録され, 各種 Artifact の配置が完了した状態とする. 以下にプロトタイプのユースケースを示す(**図 8**).

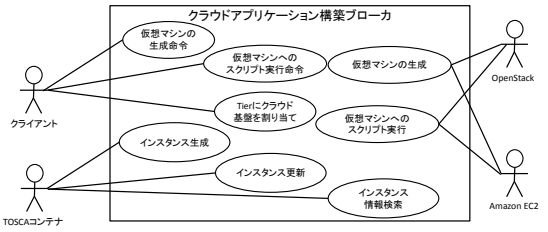

図 8 プロトタイプのユースケース

6.4. クラウドアプリケーションのインスタンス管理

プロトタイプでは, インスタンスの管理に XML ファイルを用いた. クラウドアプリケーションのインスタンスが生成されるのは, TOSCA コンテナに存在するアプリケーションのインスタンス名を定義し, Tier 情報にクラウド基盤情報を割り当てたときである. これにより, Tier 情報にクラウド基盤情報を持つ, 定義したインスタンス名の XML ファイルが作成される. このインスタンスの XML ファイルに対してカスタマイズや, 仮想マシン生成時に発生する情報を登録する.

6.5. インスタンスの情報検索

インスタンスの情報検索は, TOSCA コンテナに格納されるインスタンスの XML ファイルから行う(**図 9**). 仮想マシンの生成を行う抽象 API が実行されると, XML ファイルから Tier の情報を検索して割り当てられたクラウド基盤の情報を取得する. そして, そのクラウド基盤に対して仮想マシンの生成を行う. 仮想マシンに対してミドルウェアやアプリケーションなどのインストールを行うスクリプトが実行されると, XML ファイルのトポロジ情報から, スクリプトが実行されるべきサーバの IP アドレスを検索する. その後, SSH を用いてスクリプトの実行を行う. この情報検索方法により, Service Template に定義された構成情報を取得してアプリケーションを構築する.

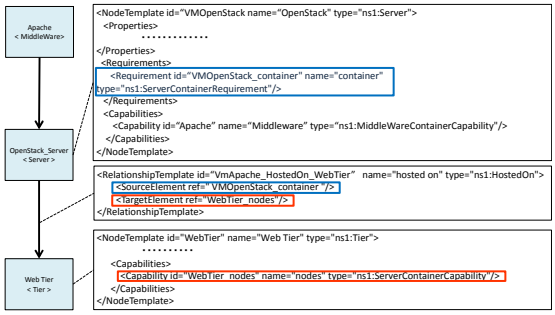

図 9 インスタンスの情報検索

6.6. プロトタイプの構造

作成したプロトタイプの構造を示す(**図 10**). プロトタイプは, クライアントで実行された抽象 API に応じてアプリケーションの分散構築を実現する.

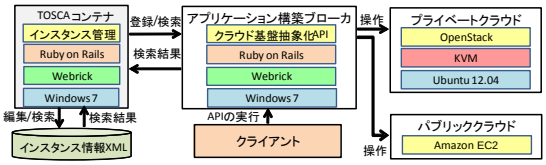

図 10 プロトタイプの構造

7. プロトタイプの適用

7.1. 適用シナリオ

Apache が動作する 2 つの仮想マシンを構築する TOSCA 仕様を作成する. この 2 つの仮想マシンを Amazon EC2 と OpenStack へ割り当てる. そして, 抽象 API の実行によりサ

ーバの構築を行う. 構築後にクライアントから各サーバが外部アクセス可能かを検証した(**図 11**).

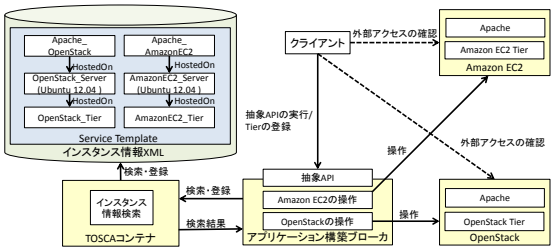

図 11 プロトタイプのシナリオ

7.2. 適用シナリオの振舞い

以下に例題の振舞いを示す(**図 12**).

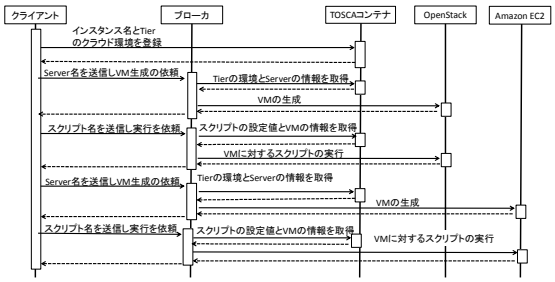

図 12 プロトタイプの振舞い

8. 評価と考察

8.1. 抽象 API の評価

(1) 抽象 API の機能要求の実現性

ハイブリッドクラウドにアプリケーションを分散構築するためには, サーバのリソース要求を満たす仮想マシンを生成し, 外部からアクセスできる環境構築が必要である. これに対して, 抽象 API は Service Template にリソースの要求を設定した構成情報から仮想マシンの生成を実現し, 外部からアプリケーションを利用, アプリケーション間で連携するための外部アクセス可能な IP アドレスの割り当てを実現する.

(2) 抽象 API の妥当性

仮想マシンの生成を行う抽象 API の引数をクラウドアプリケーションのインスタンス名とサーバ名に抽象化し, 抽象 API 実行後に Service Template から詳細なパラメータを取得する. 詳細なパラメータは低水準 API の引数に対応しているため, OpenStack では詳細なインスタンスタイプを生成でき, Amazon EC2 ではユーザの要求を満たす最小のインスタンスタイプを導出する. Amazon EC2 はインスタンスタイプに応じてコストが発生するため, 最小のインスタンスタイプを割り当てることでコストも最小に抑えることができる. この結果, クラウド基盤ごとに異なる API のシグネチャとその抽象度を吸収し, クラウド基盤によらない仮想マシンの生成ができるため, 抽象 API は妥当である.

8.2. ハイブリッドクラウドに対するサーバ構築の効率化

TOSCA により異なるクラウド基盤間でアプリケーションの構成定義を統一した. そして, ブローカにより異なるクラウド基盤へ透過的に仮想マシンの生成が行え, クラウド基盤ごとの仮想マシン生成スクリプトが不要になった. また, Plan からアプリケーションの自動構築が可能である. この結果, 従来のクラウド基盤ごとに異なる API を操作しアプリケーションを構築する方法と比べ, サーバ構築業務の効率向上が期待できる.

8.3. 提案アーキテクチャの妥当性

ブローカにより, ハイブリッドクラウドを構成するクラウド基盤の仕様変更や, 異なるクラウド API を持つクラウド基盤の追加に対する変更を局所化できる. この結果, クラウドアプリケーションを利用するユーザごとに異なるクラウド基盤を組み合わせ, ハイブリッドクラウドを容易に構築できる.

9. 今後の課題

9.1. 異なるクラウド基盤間の設定を同期

クラウド基盤では, 仮想マシンのインスタンスタイプやセキュリティグループなど, 様々な要素をあらかじめ設定することが可能である. 保守や運用で設定値の差異によって混乱が生じないように異なるクラウド基盤間で設定を同期する必要がある.

9.2. クラウドアプリケーションのインスタンスを移植

ユーザの要求するクラウド環境を構築するために, クラウドサービスの停止や安価な新規サービスの提供に対して, 構築したクラウドアプリケーションのインスタンスが容易に移植できることが望ましい. そのため, インスタンス運用時に記録したデータなどを抽出し, 異なるクラウド環境へ自動的にインスタンスを構築する機能を追加する必要がある.

10. まとめ

TOSCA による構成管理方法の統一と, クラウド基盤ごとに異なる API の差異を吸収する抽象 API を実装したブローカにより, クラウド基盤によらないアプリケーションの構築を可能にした. この結果, Plan によって, ハイブリッドクラウドへクラウドアプリケーションを自動構築できる. 提案したアーキテクチャのプロトタイプを実装し, 抽象 API の妥当性, ハイブリッドクラウドに対するサーバ構築の効率化, 提案アーキテクチャの妥当性を評価した.

参考文献

[1] Amazon, Amazon EC2 (Amazon Elastic Compute Cloud), http://aws.amazon.com/jp/ec2/.

[2] OASIS, Topology and Orchestration Specification for Cloud Applications Version 1.0, Nov. 2013, http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html

[3] OpenStack, Open Source Cloud Computing Software, http://www.openstack.org/.

[4] B. Sotomayor, et al., Virtual Infrastructure Management in Private and Hybrid Clouds, IEEE Internet Computing, Vol. 13 . No. 5, 2009 , pp .14 - 22.