Vol.2015-MPS-103 No.18
Vol.2015-BIO-42 No.18
2015/6/23

クラウドブローカーのための抽象的なシステム記述の検討

三浦 克宜[1,a)]　齋藤 篤志[2,b)]　玉家 武博[2,c)]　棟朝 雅晴[2,d)]

概要：IaaS のためのクラウド事業者の増加に伴い，複数のクラウド事業者のサービスを連携したインタークラウドが，構築するシステムに関する耐障害性の向上や柔軟なスケールアウトを実現するために注目されている．クラウドブローカーは，顧客要求に適したシステムをクラウド基盤上に構築するため，満たすべき制約条件やシステムの性質を精査し，適切なクラウドサービスを選択する．大規模なシステムの構想では，複雑に関連しあった多数の制約条件を，ブローキングのために有用なボリュームで分解し，個々に独立した知識として記述することが重要である．また，明確な知識の記述は，顧客とクラウドブローカー間の構築するシステムの全体像の差異を減らせられる．本論文では，論理式によるシステム記述を検討しており，それはシステムの性質や制約条件を原子論理式により記述し，そのコンジャンクションによりシステムの全体像を抽象的に記述する方法である．我々は，その原子論理式のコンジャンクションを抽象システム記述式と呼ぶ．原子論理式においては，述語でクラウドサービスを表し，項でクラウドサービスに対するパラメータを表している．その他に必要に応じてネットワーク構造やシステムに要求する性能などの制約条件を記述する原子論理式が存在する．最後に，本論文では，論理式による抽象的なシステム記述の必要性を議論するとともに，具体例に基づき抽象システム記述式を作り出す過程を示す．

キーワード：クラウド, システム記述, 原子論理式, ブローカー

1. はじめに

IaaS のためのクラウド事業者の増加により，商用や学術研究など様々な用途の基幹システムがクラウド基盤上に仮想マシン (VM) として展開されている．Amazon Elastic Compute Cloud (EC2) [1] は，Amazon Web Service (AWS) のクラウドホスティングサービスであり，代表的なパブリック IaaS として知られている．学術研究の目的では，北海道大学などが研究者コミュニティ向けの IaaS を提供している．このように複数のクラウド基盤が存在する背景のもとで，地理的に分散するクラウド基盤を連携したインタークラウドが，構築するシステムに関する耐障害性の向上や柔軟なスケールアウトを実現するために注目されている．AWS や IBM SoftLayer [2] などのデータセンター (DC) は，世界中に分散配置されており，DC 間で連携したインタークラウドが可能である．クラウド事業者間でのインタークラウドの技術は，まだ十分に与えられていない．

[1] 北見工業大学, Kitami Institute of Technology
Kitami, Hokkaido 090–8507, Japan
[2] 北海道大学, Hokkaido University
Sapporo, Hokkaido 060–0811, Japan
[a)] k-miura@mail.kitami-it.ac.jp
[b)] a.saito@eis.hokudai.ac.jp
[c)] student-tt@ec.hokudai.ac.jp
[d)] munetomo@iic.hokudai.ac.jp

クラウド利用者には，複数のクラウド事業者の中からシステム要件に適したクラウドサービスを選択し，その上でシステムを構築し運用するクラウド活用の能力が求められる．大規模なシステムの構想では，システム上で稼働するアプリケーションやネットワーク構造，必要なシステム性能，システムの可用性，運用経費などの多数の制約条件が複雑に関連しあっている．クラウドブローカーは，顧客要求に適したシステムをクラウド基盤上に構築するために，満たすべき制約条件やシステムの性質を精査し，適切なクラウドサービスを選択する．その他に，クラウドブローカーは，システムのプロビジョニングや VM の管理などの保守も行う．

クラウドブローカーがシステム要件を十分に把握した上で，適切なクラウドサービスを選択しシステムをプロビジョニングするには，複雑に関連しあった多数の制約条件を，ブローキングのために有効なボリュームで分解し，個々に独立した知識として明確に記述することが重要である．本論文では，これを達成するために，論理式によるシステム記述を検討しており，それはシステムの性質や制約条件を原子論理式により記述し，そのコンジャンクションによりシステムの全体像を抽象的に記述する方法である．我々は，その原子論理式のコンジャンクションを抽象シス

テム記述式と呼ぶ．原子論理式 [3] は，その式中に部分論理式を持たない論理式であり，1 個の述語と 0 個以上の項から構成されている．抽象システム記述式の原子論理式においては，述語でクラウドサービスを表し，項でクラウドサービスに対するパラメータを表している．その他に必要に応じてネットワーク構造やシステムに要求する性能などの制約条件を記述する原子論理式を定義している．

本論文は次の通りに構成されている．2 章では，システムのプロビジョニングやクラウドサービスのブローキングなどのクラウドブローカー支援に関する研究を紹介する．3 章では，与えられたシステム要件を満たす全体の構造を抽象的に記述する必要性を議論する．4 章では，抽象システム記述式で扱われる 6 種類の原子論理式を定義し，それら原子論理式を使いシステムを記述する方法を説明する．さらに具体例に基づき抽象システム記述式を作り出す過程を示す．最後に 5 章で，抽象システム記述式によりシステム構造を定義する有用性を議論する．

2. 関連研究

2.1 AWS CloudFormation

AWS CloudFormation [4] は，AWS リソースである EC2 や Simple Storage Service (S3) などのクラウドサービスによるシステム構築とその管理を自動化するサービスである．AWS CloudFormation では，テンプレートと呼ばれる定義ファイルに，システムのプロビジョニングに必要な情報を記述する．記述される情報は，展開するクラウドサービスとインスタンスのタイプ，システム上で稼働するアプリケーション，展開と同時に実行する命令コマンドなどである．AWS CloudFormation は，クラウドサービス同士の依存関係からインスタンスの起動順を計算し，適切な順番でインスタンスを展開する．そのため AWS CloudFormation を使うと，人手によるインスタンス展開順の制御が不要となる．しかしながら，システム構築に関して情報を詳細に記述する必要があるため，実用性の高いシステムの構築では，テンプレートが長くなると共にデバックコストが高くなる．

ここで AWS CloudFormation の具体例を示す．以下のソースコードは，EC2 インスタンスを用いた Web サーバの展開に関して記述している．AWS CloudFormation では，Resources の中にインスタンスの展開に必要な情報が記述されている．FW::SSH と FW::HTTP は開放ポートなどのファイアウォールに関する記述で，WebServer は展開するクラウドサービスとインスタンスのタイプに関する記述である．WebServerEip には，インスタンスに Elastic IP アドレスを割り当てることを記述している．UserData に記述されている内容は，展開と同時に実行される命令コマンドである．

```
{"AWSTemplateFormationVersion":"2010-09-09",
 "Resources":{
  "FW::SSH":{
   "Type":"AWS::EC2::SecurityGroup",
   "Properties":{
    "GroupDescription":"Enable SSH access via port 22",
    "SecurityGroupIngress":[{
     "IpProtocal":"tcp", "CidrIP":"0.0.0.0/0",
     "FromPort":"22", "ToPort":"22"
    }]
   }
  },
  "FW::HTTP":{
   "Type":"AWS::EC2::SecurityGroup",
   "Properties":{
    "GroupDescription":"Enable HTTP access via port 80",
    "SecurityGroupIngress":[{
     "IpProtocal":"tcp", "CidrIP":"0.0.0.0/0",
     "FromPort":"80", "ToPort":"80"
    }]
   }
  },
  "WebServer":{
   "Type":"AWS::EC2::Instance",
   "Properties":{
    "ImageId":"ami-54cf5c3d",
    "InstanceType":"m3.medium",
    "SecurityGroupIds":[
     {"Ref":"FW::HTTP"}, {"Ref":"FW::SSH"}
    ],
    "UserData":{"Fn::Base64":{"Fn::Join":["",[
    "#!/bin/bash\n",
    "yum -y install httpd openssh-server\n",
    "chkconfig httpd on\n",
    "chkconfig sshd on\n",
    "service httpd start\n"
    "service sshd start\n"
    ]]}}
   }
  },
  "WebServerEip": {
   "Type": "AWS::EC2::EIP",
   "Properties": {
   "InstanceId": { "Ref": "WebServer" }
   }
  }
 }
}
```

2.2 CloudyMetrics

CloudyMetrics [5] は，Amazon EC2 または Rackspace Cloud [6] のインスタンスの中から，与えられた条件を満たすインスタンスを見つけ出す Web サービスである．検索条件には，

- インスタンスのタイプ名
- CPU のコア数，メモリ容量，ストレージ容量
- インスタンスのブート時間

- ファイルの読み書き込み速度
- インスタンスの価格

を与えることができる．CloudyMetrics には Web API が提供されており，REST 形式で問い合わせを与えると，検索結果を JSON 形式または HTML 形式で獲得することができる．例えば，以下の REST 形式の問い合わせを与えると，Amazon EC2 の M1 インスタンスのリストを JSON 形式で獲得することができる．

http://api.cloudymetrics.com/api/iaas/amazon/instances/m1

2.3 CSP-index によるクラウド事業者の選択

Cloud Service Provider (CSP) index [7] は，ビット列で表現されたクラウド事業者を B+木のデータ構造に従い整理したクラウド事業者データベースである．クラウド事業者を表すビット列は，いくつかの要素ビット列の排他的論理和により得られる．その要素ビット列は，クラウド事業者が提供する VM に割り当て可能なストレージ容量の範囲や VM にインストール可能な OS の種類，セキュリティレベルなどである．ストレージに関する要素ビット列は，

1 桁目: 0MB–500MB
2 桁目: 500MB–1GB
3 桁目: 1GB–10GB
4 桁目: 10GB–∞

の 4 桁で表されており，例えば 900MB–2GB のストレージ容量の範囲を提供するならば，これに関する要素ビット列は 0110 となる．CSP-index によるクラウド事業者の選択では，セキュリティレベル，メモリ容量，ストレージ容量，OS の種類の中のいくつかを決定すると，k-最近傍探索に従って CSP-index 上から条件を満たすクラウド事業者を選択する．検索条件もクラウド事業者と同様にビット列で表されている．

2.4 SMI/AHP によるクラウドサービスの順位付け

Service Measurement Index (SMI) / Analytical Hierarchical Process (AHP) によるクラウドサービスの順位付け [8] は，Amazon EC2 や Rackspace Cloud などのインスタンスを，インスタンスの計算性能，耐障害性のレベル，平均サービス応答時間などに付加したウェイトをもとに順位付けする方法である．SMI は，ウェイトが付加される項目のリストであり，カーネギーメロン大学などが参加する Cloud Service Measurement Initiative Consortium (CSMIC) により提唱されている．SMI ではアジリティの項目は，CPU やメモリなどの計算性能と弾力性の 2 項目から構成されている．ウェイトの付加は項目ごとに 1.0(100%) になるように割り振られる．そのためアジリティの項目では，計算性能と弾力性に割り振ったウェイトの和が 1.0 になる．クラウドサービスの順位付けは，AHP に基づく問題解決の方法に従って，項目に付加されたウェイトに基づいて実行される．

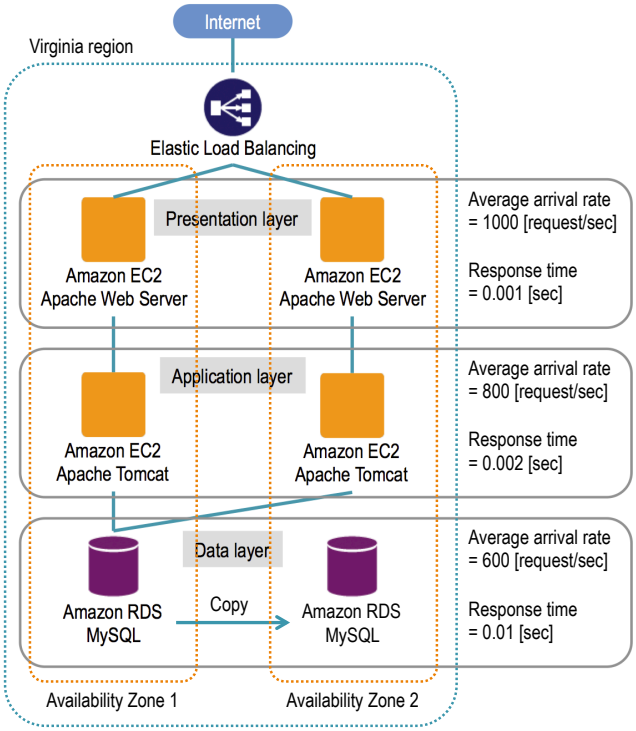

図 1 クラウドコンポーネントによるシステム記述例 [9]

2.5 関連研究の位置付け

クラウドブローカーがシステム要件に関してクラウドサービスを決定しインスタンスを展開するまでの作業の流れを，大まかに以下の通りに分類する．

作業 1: システム要件を満たす計算性能を具体化する
作業 2: 求める性能を提供するクラウドサービスを選択する
作業 3: クラウドサービス上にインスタンスを展開する

これらの 3 つの作業のうち，関連研究が主にどの作業に関して支援を行っているのかを示そう．AWS CloudFormation は，利用するクラウドサービスやインスタンスが具体的に確定した上で，インスタンスの展開や管理を自動化するサービスであるので，これは作業 3 を支援する研究といえる．次に，CloudyMetrics は，具体的な計算性能や価格コストなどの検索条件をもとに，その条件を満たすクラウドサービス上のインスタンスを見つけ出すためのサービスであるので，これは作業 2 を支援する研究といえる．CSP-index によるクラウド事業者の選択は，システム要件に基づく検索条件のもと，その条件を満たすインスタンスを提供するクラウド事業者を見つけ出すためのサービスであるので，CloudyMetrics と同様に，作業 2 を支援する研究といえる．最後に，SMI/AHP によるクラウドサービスの順位付けは，応用的な利用として，システム要件に適したクラウドサービスを見つけ出すのに有効であるので，作業 2 を支援する研究といえる．

3. 抽象的なシステム記述の必要性

クラウドブローカーは，システム要件を満足する計算性

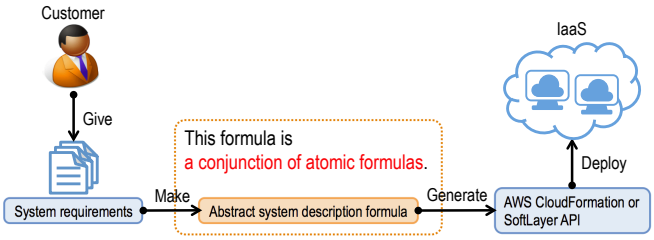

図 2 提案する枠組みによりシステムが展開されるまでの流れ

能やセキュリティを有するクラウドサービスを見つけ出し，そのクラウドサービスを構成要素として適切にシステム全体を構築しなければならない．一般にシステム要件は複雑に関連しあった多数の制約条件であり，システム要件から直接的に適切なクラウドサービスを推論しシステム全体を構築することは容易ではない．効率的に推論を行うために，制約条件を満たす全体像をシステム記述として一般的に与えることが重要である．

AWS CloudFormation は構築するシステムの全体像を記述するが，システム構造の推論のための枠組みとして扱うことができるだろうか．AWS CloudFormation は，クラウドサービスやインスタンスのタイプ，開放ポートなどのファイアウォールの設定，その他に多様なパラメータを，システム要件に合わせて詳細に記述できる．もし AWS CloudFormation が有効であるならば，与えられたシステム要件からシステムのプロビジョニングまでを自動的に実行することが可能になる．しかし，AWS CloudFormation は AWS リソースを具体的にどのように使うのかという手続き的な内容を記述しており，推論により解を導き出すという使い方を想定していない．

我々は，システム要件に関して正しく推論を行うために，複雑に関連しあった多数の制約条件を，個々に独立した知識として記述する必要があると考えている．これまでに知識の集合としてシステム記述を与える枠組みは与えられていないが，AWS は構成図によりシステム全体の構造を描画する方法を与えている．その構成図は，クラウドコンポーネント [9] と呼ばれる AWS リソースを抽象的に表現したアイコンを使って作成される．図 1 は，AWS リソースを使って，Web3 層モデルのシステム構成を描画した例である．Web3 層モデルは，クライアントサーバシステムをプレゼンテーション層，アプリケーション層，データ層の 3 層に分類し，各層を独立して設計する Web アプリケーションの開発体系である．本論文では，計算機によるシステム構造の推論を想定しており，図 1 で示されるようなシステム記述を，論理式により記述する方法を与える．

4. 抽象システム記述式

4.1 モデル

本論文で検討する枠組みは，図 2 で示すように，顧客より与えられるシステム要件をもとに，満たすべき制約条件やシステムの構造などのシステムの全体像を原子論理式 [3] のコンジャンクションにより記述する．厳密には，全称束縛付きのコンジャンクションである．我々は，その原子論理式のコンジャンクションを抽象システム記述式と呼ぶ．抽象システム記述式は，システム要件を満たすために必要な計算性能を推論するための入力であり，その推論から具体的なクラウドサービスやインスタンスのタイプ，その他のパラメータを決定する．そして，推論により得られた情報をもとに，AWS CloudFormation または SoftLayer API [10] などの実システムを展開するために使われる具象的なシステム記述を作成する．具象的なシステム記述が得られると，実システムは自動的に展開される．

原子論理式 [3] は，述語 p と項 t で構成されており，例えば $p(t_1, t_2)$ のように書かれる．述語 p は，原子論理式中の項の関係を定義しており，1 個の原子論理式に対して必ず 1 個現れる．項 t は変数または定数であり，0 個以上の項が原子論理式中に現れる．本論文において，変数はアスタリスク $(*)$ から始まる値とし，定数は数またはシンボルとする．述語 p は，確定節により意味を定義することで自由に増やすことができる．抽象システム記述式の中の原子論理式において，述語でシステムの構成要素であるクラウドサービスを表し，項でそのクラウドサービスに対するパラメータを表している．その他に必要に応じてネットワーク構造やシステムに要求する性能などの制約条件を記述する原子論理式が存在する．

4.2 抽象システム記述式を構成する原子論理式

4.2.1 原子論理式の定義

抽象システム記述式では AWS リソースの 3 つのクラウドサービスを扱っており，クラウドサービスとそれに対応する述語は次の通りである．

- *ec2* 述語 ··· Amazon Elastic Compute Cloud (EC2)
- *rds* 述語 ··· Amazon Relational Database Service (RDS)
- *elb* 述語 ··· Elastic Load Balancing (ELB)

これら述語に関する原子論理式は，一般的に次のように記述される．原子論理式の中の同じ名前の変数は同じ意味で解釈される．

$$ec2(*id,\ *layer,\ *name,\ *region,\ *az)$$
$$rds(*id,\ *layer,\ *name,\ *region,\ *az)$$
$$elb(*id,\ *name,\ *region,\ *az)$$

それぞれの変数には以下に示す数またはシンボルが代入される．

$*id$···個々の原子論理式に割り振る識別番号であり，原子論理式同士で値が重複しない自然数である

$*layer$···クラウドサービスを配置する Web3 層モデルのレイヤの識別子であり，{pres, appl, data} のいずれかで

ある

$*name$…AWS のインスタンスのタイプ名である

例えば，t2.micro，m3.medium，c3.xlarge などである

$*region$…クラウドサービスを展開するリージョンである

例えば，virginia，oregon，japan などである

$*az$…クラウドサービスを展開するアベイラビリティゾーン (AZ) である

例えば，us-virginia-a，us-virginia-b などである

抽象システム記述式は，Web3 層モデルによる Web アプリケーションのシステム構造を記述する．変数 $*layer$ に代入される識別子は，それぞれプレゼンテーション層，アプリケーション層，データ層を表している．上述の述語の他に，抽象システム記述式では，*layerRequest*，*connect*，*noteq* の 3 つの述語が与えられており，それらは一般的に次のように記述される．

$$layerRequest(*layer, *arrival, *response)$$
$$connect(*id1, *id2)$$
$$noteq(*v1, *v2)$$

layerRequest 述語の原子論理式は，Web3 層モデルの $*layer$ 層に対する 1 秒あたりのリクエスト数 ($*arrival$) と平均サービス応答時間 ($*response$) に関する制約条件を記述する．*connect* 述語の原子論理式は，クラウドサービス同士の繋がり方に関する制約条件を記述しており，変数 $*id1$ と変数 $*id2$ は，クラウドサービスを表す原子論理式に割り振られた識別番号である．*noteq* 述語の原子論理式は，与えられた 2 つの値が異なることを表しており，実際の使われ方は，4.3 章の抽象システム記述式の具体例により説明する．抽象システム記述式では上述の 6 種類の原子論理式が使われる．

4.2.2 原子論理式の例

原子論理式の項は常に定数であるとは限らず，変数と定数の両方が存在する場合がある．ただし，抽象システム記述式では，$*id$ および $*layer$ に必ず定数が与えられるとしている．

例えば，アプリケーション層のサーバとして EC2 インスタンスを virginia リージョンの us-virginia-a に展開することを決定したが，インスタンスのタイプが決定していないとしよう．この場合，*ec2* 述語の原子論理式の $*region$ と $*az$ の部分には virginia と us-virginia-a が記述されるが，$*name$ の部分には，例えば $*x$ のような変数が記述される．このパラメータのもとで，*ec2* 述語の原子論理式を作成すると，それは

$$ec2(1, appl, *x, virginia, us\text{-}virginia\text{-}a)$$

と記述される．我々は非決定的な項を変数で書くことで，その後ブローキングエンジンの推論により，その項の具体値を導出する．

その他の例として，Web3 層モデルのレイヤに対する制約条件を記述する *layerRequest* 述語の原子論理式を示そう．この原子論理式の項においては，必ず数やシンボルが与えられる．例えば，アプリケーション層に対して，1 秒あたり 1000 リクエストが与えられ，1 リクエストあたり平均サービス応答時間 0.001 秒で処理を行うとしよう．この場合，*layerRequest* の原子論理式を用いて，

$$layerRequest(appl, 1000, 0.001)$$

と記述される．

4.3 抽象システム記述式の作り方

本例で作成する抽象システム記述式が，どのようなシステム構造を記述しているのかを直感的に分かり易くするために，図 1 に示したクラウドコンポーネント [9] によるシステム記述例を基とする．図 1 のシステム記述では，EC2 インスタンスが 4 台，RDS インスタンスが 2 台そして ELB インスタンスが 1 台の合計 7 台のインスタンスが展開されている．EC2 インスタンスは，プレゼンテーション層とアプリケーション層に各 2 台が展開されている．RDS インスタンスは，データ層に展開されている．すべてのインスタンスは virginia リージョンに展開されている．AZ は具体的に明言されていないが，2 拠点の AZ により冗長化している．以上の制約条件より，*ec2* 述語，*rds* 述語そして *elb* 述語の原子論理式が以下の通り作り出される．

$$ec2(1, pres, *n1, virginia, *a1)$$
$$ec2(2, pres, *n2, virginia, *a2)$$
$$ec2(3, appl, *n3, virginia, *a1)$$
$$ec2(4, appl, *n4, virginia, *a2)$$
$$rds(5, data, *n5, virginia, *a1)$$
$$rds(6, data, *n6, virginia, *a2)$$
$$elb(7, virginia)$$

異なる 2 拠点の AZ が利用されるので，変数 $*a1$ と変数 $*a2$ が異なるという制約条件である

$$noteq(*a1, *a2)$$

を追加する．インスタンス同士の繋がり方に関する制約条件は，*connect* 述語の原子論理式により記述されるので，本例では以下の通り作られる．

$$connect(7, 1)\ connect(7, 2)\ connect(1, 3)$$
$$connect(2, 4)\ connect(3, 5)\ connect(4, 5)$$
$$connect(5, 6)$$

connect 述語の原子論理式の項は，*ec2*，*rds*，*elb* の 3 つの述語の原子論理式に与えられた識別番号である．抽象シ

ステム記述式では，Web3 層モデルの各レイヤに要求する性能数値も記述する．この制約条件を記述するために *layerRequest* 述語の原子論理式を導入している．本例では，*layerRequest* 述語の原子論理式は次の通り作成される．

$$layerRequest(\text{pres}, 1000, 0.001)$$
$$layerRequest(\text{appl}, 800, 0.002)$$
$$layerRequest(\text{data}, 600, 0.01)$$

結果として本例の抽象システム記述式は，18 個の原子論理式を構成要素とする全称束縛付きのコンジャンクションとなる．

$$\begin{aligned}\forall\{\ & ec2(1, \text{pres}, *n1, \text{virginia}, *a1),\\ & ec2(2, \text{pres}, *n2, \text{virginia}, *a2),\\ & ec2(3, \text{appl}, *n3, \text{virginia}, *a1),\\ & ec2(4, \text{appl}, *n4, \text{virginia}, *a2),\\ & rds(5, \text{data}, *n5, \text{virginia}, *a1),\\ & rds(6, \text{data}, *n6, \text{virginia}, *a2),\\ & elb(7, \text{virginia}),\\ & noteq(*a1, *a2),\\ & connect(7, 1), connect(7, 2), connect(1, 3),\\ & connect(2, 4), connect(3, 5), connect(4, 5),\\ & connect(5, 6),\\ & layerRequest(\text{pres}, 1000, 0.001),\\ & layerRequest(\text{appl}, 800, 0.002),\\ & layerRequest(\text{data}, 600, 0.01)\ \}\end{aligned}$$

5. 議論とまとめ

5.1 AWS CloudFormation との比較

システム記述の有用性に関する議論として，システム記述の内容の読み易さや誤りの発見し易さが指標のひとつとして挙げられる．この観点から抽象システム記述式と AWS CloudFormation に関して考察してみよう．抽象システム記述式は，ひとつの制約条件がひとつの原子論理式により記述される．そのため原子論理式を見ることで，どのような制約条件が記述されているのかを確認することができる．例えば，利用されているクラウドサービスは原子論理式の述語により確認でき，展開するインスタンス数は原子論理式の数により確認できる．その他に，システムのネットワーク構造は *connect* 述語の原子論理式により確認できる．一方，AWS CloudFormation は，ひとつの制約条件をプログラミング言語のメソッドのような断片で記述している．その断片ひとつのボリュームは一定でなく複数行の長いものになることがある．そのため，システム記述のボリュームが増すほど，解読や解析のコストが増してしまう．

抽象システム記述式は，AWS CloudFormation よりもシンプルにシステム構造を記述できる．例えば，4.3 章で示した抽象システム記述式が定義している図 1 のシステム構造を，AWS CloudFormation で記述すると 500 行超のソースコードが必要になると推測できる．なぜなら，AWS CloudFormation は EC2 インスタンスひとつの記述に，2 章で示した 50 行ほどのソースコードが必要である．

5.2 まとめ

本論文では，システムの性質や満たすべき制約条件を原子論理式により定義し，そのコンジャンクションによりシステムの全体像を記述する方法を検討した．この方法により，ネットワーク構造や利用するクラウドサービスなどのハードウェア仕様に関して明確に記述できるようになった．原子論理式によるシステム記述は，関連しあった多数の制約条件を，それらの関係性を保持したまま分解し記述することを可能にした．

今後は，システム内のデータの流れや実行されるアプリケーションなどのソフトウェア仕様に関して記述可能な方法に拡張すると共に，システム構成を具体化するための推論に関する方法を検討する．

参考文献

[1] Amazon web services: Amazon EC2, Amazon.com (online), available from ⟨http://aws.amazon.com/ec2/⟩ (accessed 2015-02-18).
[2] SoftLayer an IBM Company: SoftLayer, IBM (online), available from ⟨http://www.softlayer.com/⟩ (accessed 2015-02-18).
[3] 森田茂行：論理学 -意味とモデルの理論-，東京電機大学出版局 (1999).
[4] Amazon web services: AWS CloudFormation, Amazon.com (online), available from ⟨http://aws.amazon.com/cloudformation/⟩ (accessed 2015-02-18).
[5] Smit, M., Pawluk, P., Simmons, B. and Litoiu, M.: A Web Service for Cloud Metadata, *The 8th IEEE World Congress on Services*, IEEE, pp. 361–368 (2012).
[6] Rackspace: Rackspace, Rackspace Cloud (online), available from ⟨http://www.rackspace.com/⟩ (accessed 2015-02-23).
[7] Sundareswaran, S., Squicciarini, A. C. and Lin, D.: A Brokerage-Based Approach for Cloud Service Selection, *The 5th IEEE International Conference on Cloud Computing*, IEEE, pp. 558–565 (2012).
[8] Garg, S. K., Versteeg, S. and Buyya, R.: A framework for ranking of cloud computing services, *Future Generation Computer Systems*, Vol. 29, No. 4, pp. 1012–1023 (2013).
[9] Amazon web services: CDP:クラウドコンポーネント-AWS-CloudDesignPattern，Amazon.com（オンライン），入手先 ⟨http://aws.clouddesignpattern.org/index.php/CDP:クラウドコンポーネント⟩（参照 2015-03-26）.
[10] SoftLayer an IBM Company: SoftLayer API, IBM (online), available from ⟨http://sldn.softlayer.com/reference/softlayerapi/⟩ (accessed 2015-02-18).