Vol.2015-MPS-105 No.6
2015/9/29

述語論理式による仕様記述に基づくクラウドブローキングの提案

三浦 克宜[1,a)]　齋藤 篤志[2,b)]　棟朝 雅晴[2,c)]

概要：本論文では，システム構成や要求性能値などハードウェアに関する顧客要求を記述する述語論理式を新たに定義し，その述語論理式を基にシステム構成を検証しアプリケーション環境上で利用する仮想マシンの性能スペックを選定する方法を提案する．述語論理式は原子論理式の連言であり，原子論理式の述語でクラウドサービスを表し，項でパラメータを表す．

キーワード：クラウドブローキング, 述語論理式, クエリ節, 等価変換理論

1. はじめに

IaaS (Infrastructure as a Service) のためのクラウド事業者の増加は，商用や学術研究など様々な用途の基幹システムを，仮想マシン (VM : Virtual Machine) として，クラウド基盤上に展開することを促進させた．IaaS は仮想化された計算基盤を提供するサービス形態である．Amazon Web Service (AWS) のクラウドサービスである Amazon Elastic Compute Cloud (EC2) [3] は，代表的なパブリック IaaS である．学術研究の用途では，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．

クラウドサービスの拡大に伴い，クラウド利用に係る作業が複雑で高コストになり，近年，顧客に代わり，顧客要求に合うサービスの選定や新たなサービスの構築を行うクラウドブローカー [6] が注目されている．本論文では，主に，IaaS 活用におけるクラウドブローキングを議論している．顧客要求はいくつかの制約条件の集まりである．多くの場合，顧客要求は相互依存した多数の制約条件となり，大規模なアプリケーション環境のためのシステム構成であるならば，顧客要求は複雑さが増し，クラウドブローキングは高度な制約充足問題となる．

これまでのクラウドブローキングの研究 [4], [11], [12] は，顧客要求を基に，システム構成を描画したりクラウドサービスを検索や展開したりする方法を議論している．その他に，円滑なクラウドブローキングのために，アプリケーション環境のシステム仕様に関する記述法が議論されており，それは手続き的な仕様記述と数学的な仕様記述の 2 種類に大きく分けられる．手続き的な仕様記述としては，Amazon や Microsoft，IBM などのクラウド事業者の多くが提供する，自社のクラウドサービスを基にアプリケーション環境を構築したり，その環境を管理したりするための支援サービスが挙げられる．例えば，Amazon のサービスである AWS CloudFormation[2] では，テンプレートと呼ばれる定義ファイルに，“どのクラウドサービスをどのように使うか” というシステムのプロビジョニングに必要な情報を記述する．一方，数学的な仕様記述としては，命題論理式による顧客要求の記述法が挙げられる．Chen らによる研究 [5] は，命題論理式で定義されたクラウド事業者が持つ企業ポリシーに対して，顧客要求を制約条件として与えたときの充足可能性を自動的に決定する方法を提案している．

本論文では，数学的な仕様記述の新たな方法として，述語論理式による顧客要求の記述法を提案する．三浦らによる研究 [13] では，述語論理式により顧客要求を定式化する重要性を議論している．述語論理式は，その式中に非基礎項を扱うことで抽象的に無限の基礎例を定義できるので，常に有限の基礎例で定義する命題論理式と比べ，問題の記述力が高いことから，制約充足問題や最適化問題など多様な問題でしばしば適用される．しかし，これまでに述語論理式を問題記述とし推論や証明を行う有効な計算モデルが与えられていなかったため，クラウドブローキングに関する研究では，述語論理式を扱うアプローチは提案されてい

[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)] munetomo@iic.hokudai.ac.jp

ⓒ 2015 Information Processing Society of Japan

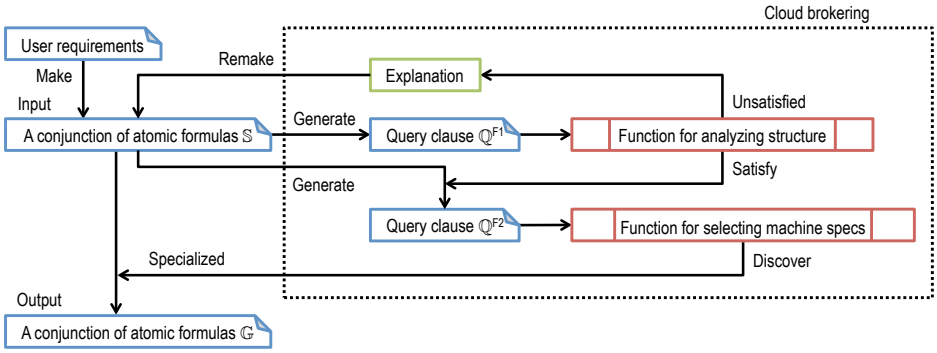

図 1 提案方法によるクラウドブローキングの処理フロー

なかった．本論文では，等価変換計算モデル [1] をクラウドブローキングに応用することで，述語論理式に基づくクラウドブローキングを可能にしている．

また本論文では，述語論理式に基づくクラウドブローキングとして，ユーザが定義したシステム構成を検証し，アプリケーション環境上で利用する仮想マシンの性能スペックを選定する方法を提案する．マシンスペックの選定では，制約条件として与えられる想定する 1 秒あたりの Web リクエスト数と 1 リクエストあたりの平均サービス応答時間を基に，M/M/s モデルの待ち行列理論 [7] に従って計算が実行される．提案方法では計算は，等価変換ルール [1] による宣言的な意味を保存した確定節の置き換えであり，その置き換えの起点となる確定節は，述語論理式を基に作られるクエリ節と呼ばれる確定節である．

本論文は次の通りに構成されている．2 章では，提案方法における入出力およびクエリ節を定義し，クラウドブローキング機能を説明する．3 章では，述語論理式により顧客要求を記述する方法を説明し，入力となる述語論理式を例示する．4 章では，クラウドブローキング機能に関する数理モデルを示す．5 章では，開発したクラウドブローキングツールによる動作実験の結果を示す．6 章では，形式手法の観点から提案方法の有効性を議論する．

2. クラウドブローキングのための処理モデル

2.1 処理モデルの概要

提案方法では，顧客要求を満たすための制約条件を述語論理式により記述する．述語論理式は原子論理式の全称束縛付き連言であり，原子論理式によりシステムの性質や要求性能値などハードウェアに関する制約条件が定義される．

本論文で検討する枠組みでは，図 1 で示すように，顧客より与えられる要求要件を基に原子論理式の連言 S が作成される．連言 S を次の通り定義する．

$$\mathbb{S} = \forall\{A_1 \wedge A_2 \wedge A_3 \wedge \cdots \wedge A_n\} \qquad (1)$$

原子論理式 A は顧客要求に関する制約条件であり，述語で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す．項には変数または定数が与えられる．変数はアスタリスク $(*)$ から始まる値であり，定数は数またはシンボルである．次に，その連言 S から 2 つのクエリ節 $\mathbb{Q}^{F1}$ と $\mathbb{Q}^{F2}$ が生成され，クラウドブローキングが実行される．本処理では，クエリ節 $\mathbb{Q}^{F1}$ を入力としたシステム構成の検証とクエリ節 $\mathbb{Q}^{F2}$ を入力とした仮想マシンの性能スペックの選定が実行される．最後に，連言 S 上の変数に対する基礎代入 δ が得られ，連言 S に基礎代入 δ を適用した原子論理式の連言 G がクラウドブローキングの結果として出力される．連言 G を以下の通り定義する．

$$\mathbb{G} = \forall\{A_1\delta \wedge A_2\delta \wedge A_3\delta \wedge \cdots \wedge A_n\delta\} \qquad (2)$$

2.2 クエリ節

クエリ節は，提案方法における機能を実行するにあたり，提案方法の入力から作られる確定節であり，厳密には，原子論理式の連言 $\mathbb{C}$ から $ans(*x)$ への確定節 $(ans(*x) \leftarrow \mathbb{C}.)$ である．連言 $\mathbb{C}$ は式 (1) から作られ，そして $ans(*x)$ の項である変数 $*x$ は連言 $\mathbb{C}$ 中の変数から作られる．ans 述語はクエリ節の左辺にのみ現れる特別な述語である．連言 $\mathbb{C}$ の中で使われる原子論理式は，2 つのクエリ節 $\mathbb{Q}^{F1}$ と $\mathbb{Q}^{F2}$ で異なる．4 章で 2 つのクエリ節を定義する．

2.3 機能の概要

2.3.1 システム構成の検証

式 (1) 中には，アプリケーション環境上で活用するクラウドサービスおよびネットワーク構造を定義する原子論理式および顧客要求として保証したいサービス稼働率を定義する原子論理式が含まれている．本機能では，システム構成とサービス稼働率の間の不一致が判定され，もし不一致が発生した場合，本機能では式 (1) の修正を要求する．

2.3.2 マシンスペックの選定

式 (1) 中には，アプリケーション環境に対し想定する 1 秒あたりの Web リクエスト数と 1 リクエストあたりの平均サービス応答時間を定義する原子論理式が含まれている．本機能では，VM の構成台数と Web リクエスト数，そして平均サービス応答時間から，それら制約を満たすため

ⓒ 2015 Information Processing Society of Japan

表 1 原子論理式の連言およびクエリ節を構成する原子論理式

原子論理式	述語の意味	変数の意味
$ec2(*id, *layer, *name, *region, *az)$ $rds(*id, *layer, *name, *region, *az)$ $elb(*id, *region)$	Amazon EC2 インスタンス Amazon RDS インスタンス ELB インスタンス	$*id\cdots$ 原子論理式に割り振るユニークな識別番号 $*layer\cdots$Web3 層モデルのレイヤ識別子 {pres, appl, data} $*name\cdots$AWS のインスタンスタイプ名 $*region\cdots$ リージョン名 $*az\cdots$ アベイラビリティゾーン (AZ) 名
$layerRequest(*layer, *request, *response)$	対象レイヤに対する要求処理性能	$*request\cdots$1 秒あたりのリクエスト数 $*response\cdots$ 平均サービス応答時間 [秒]
$sla(*layer, *sla)$	対象レイヤに対するサービス稼働率	$*sla\cdots$ サービス稼働率
$connect(*id1, *id2)$	クラウドサービス同士の接続	$*id1, *id2\cdots$ クラウドサービスの原子論理式の識別番号
$noteq(*v1, *v2)$	2 つの変数の非同一性	$*v1, *v2\cdots$ 任意の変数
$structure(*vms, *request, *response)$	要求処理性能とマシンスペックの関係	$*vms\cdots$ 同一レイヤに属する $*name$ を要素とするリスト
$checkSLA(*sla, *num, *result)$	サービス稼働率とシステム構成の関係	$*num\cdots$ 同一レイヤ上で並列化された VM 台数 $*result\cdots true$ または $false$

に必要なマシンスペックを，M/M/s モデルの待ち行列理論に従い算出する．ここではクラウド事業者が提供するインスタンスタイプの中で選定を行う．マシンスペックの選定では，ApacheBench によるベンチマークの測定値である “Requests per second” を，各インスタンスに関する平均サービス率として扱う．この測定値は 1 秒あたりに処理可能なリクエスト数を表している．

3. 述語論理式によるシステム記述

3.1 原子論理式の定義

本論文では AWS の 3 つのクラウドサービスを扱う．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の一般形は，表 1 に記述されており，原子論理式中の同じ名前の変数を同じ意味で解釈する．ここでは，Web3 層モデル [8] による Web アプリケーションのシステム構成を対象としている．そのため，変数 $*layer$ に代入される識別子は，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．さらに上述の述語の他に，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定義するために，*layerRequest*，*sla*，*connect*，*noteq* の 4 つの述語を導入する．これら述語の意味および原子論理式の一般形は，表 1 に記述されている．

3.2 具体例

本例では，どんなシステム構成を記述しているかを直感的に分かり易くするために，図 2 に示す AWS クラウドコンポーネント [4] によるシステム記述例を基本とする．図 2 では，EC2 インスタンスが 4 台，RDS インスタンスが 1 台そして ELB インスタンスが 2 台の合計 7 台のインスタンスが展開されている．さらに全てのインスタンスが Virginia リージョンに展開されており，AZ は明確にされていないが，2 拠点の AZ により冗長化している．以上より，*ec2*，*rds*，*elb* の述語の以下の原子論理式が作られる．

$$ec2(1, \text{pres}, *n1, \text{us-east-1}, *a1)$$
$$ec2(2, \text{pres}, *n2, \text{us-east-1}, *a2)$$
$$ec2(3, \text{appl}, *n3, \text{us-east-1}, *a1)$$
$$ec2(4, \text{appl}, *n4, \text{us-east-1}, *a2)$$
$$rds(5, \text{data}, *n5, \text{us-east-1}, *a1)$$
$$elb(6, \text{us-east-1})\ elb(7, \text{us-east-1})$$
$$noteq(*a1\ *a2)$$

$noteq(*a1, *a2)$ は，変数 $*a1$ と変数 $*a2$ が異なる AZ であると定義している．クラウドサービス同士の繋がり方は，*connect* 述語の原子論理式により，次の通り作られる．

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

connect 述語の原子論理式の項は，*ec2*，*rds*，*elb* の述語の原子論理式に与えられた識別番号である．式 (1) では，Web3 層モデルのレイヤに対する性能数値を *layerRequest* 述語と *sla* 述語を使って定義する．本例では次の性能数値が与えられていると仮定する．

$$layerRequest(\text{pres}, 50, 0.03)$$
$$layerRequest(\text{appl}, 30, 0.04)$$
$$layerRequest(\text{data}, 20, 0.05)$$
$$sla(\text{pres}, 99.9999)\ sla(\text{appl}, 99.99)\ sla(\text{data}, 99.99)$$

以上の 22 個の原子論理式から式 (1) が作られる．

4. クラウドブローキングのための機能

4.1 2 つのクエリ節の定義

クエリ節 Q^{F1} は，システム構成の検証機能を実行するにあたり，式 (1) から作り出される．このクエリ節の右辺

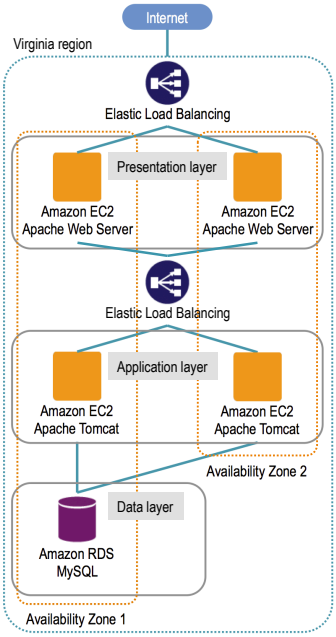

図 2 AWS クラウドコンポーネント [4] によるシステム記述例

は *checkSLA* 述語の原子論理式の連言であり，一方，左辺は $ans(*results)$ である．*checkSLA* 述語の意味とその原子論理式の一般形は，表 1 に記述されている．変数 $*result$ には，システム構成がサービス稼働率を満たすとき *true* が代入され，それ以外は *false* が代入される．変数 $*results$ は変数 $*result$ を要素とするリストである．

クエリ節 Q^{F2} は，マシンスペックの選定機能を実行するにあたり，式 (1) から作り出される．このクエリ節の右辺は *structure* 述語の原子論理式の連言であり，一方，左辺は $ans(*names)$ である．*structure* 述語の意味とその原子論理式の一般形は，表 1 に記述されている．変数 $*names$ はリスト $*vms$ 中の要素からなるリストである．

4.2 2 つの機能のメカニズム

4.2.1 システム構成の検証機能

クエリ節 Q^{F1} が与えられると，並列システム構成における信頼度の計算モデルに基づき，*checkSLA* 述語の原子論理式の第 3 引数の変数 $*result$ に *true* または *false* を代入するための処理が実行される．サービス稼働率 r の VM を n 台で並列に構成した場合の全体の稼働率 R は，$R = 1-(1-r)^n$ で求められる．稼働率 R の計算は，クエリ節 Q^{F1} 中の全ての *checkSLA* 述語の原子論理式に対して実行され，等価変換ルールとして，以下のようにコーディングされている．

$$
\begin{aligned}
&checkSLA(*sla,\ *n,\ *result),\\
&\quad \{multiplier(*n,\ *r,\ *mul),\ sub(1,\ *mul,\ *R)\}\\
&\qquad \Rightarrow greaterEq(*R,\ *sla),\ eq(*result,\ true);\\
&\qquad \Rightarrow greater(*sla,\ *R),\ eq(*result,\ flase).
\end{aligned}
$$

$multiplier(*n,\ *r,\ *mul)$ で $(1-r)^n$ の結果が $*mul$ に与えられ，$sub(1,\ *mul,\ *R)$ で 1 と $(1-r)^n$ の差が $*R$ に与えられる．$greaterEq(*R,\ *sla)$ は $*R \geq *sla$ を表しており，$greater(*sla,\ *R)$ は $*sla > *R$ を表している．

4.2.2 マシンスペックの選定機能

クエリ節 Q^{F2} が与えられると，M/M/s モデルの待ち行列理論 [7] に基づき，*structure* 述語の原子論理式の第 1 引数の変数 $*vms$ にインスタンスタイプを代入するための処理が実行される．本機能では，まず変数 $*vms$ に対して具体的なインスタンスを代入し基礎原子論理式を作る．待ち行列理論により，s 個のサービス窓口に関して，単位時間あたりのリクエストの平均到着率 λ と平均サービス率 μ から平均サービス応答時間 W が求められる．*structure* 述語の原子論理式に関して，変数 $*request$ はリクエストの平均到着率 λ であり，変数 $*response$ は要求する平均サービス応答時間 W_{req} である．平均サービス率 μ は，変数 $*vms$ に与えられるインスタンス全体の平均サービス率である．個々のインスタンスの平均サービス率は，ApacheBench の測定値 “Requests per second” により与えられるため，その値を基に平均サービス率 μ を求める．平均サービス応答時間 W は式 (3) より求められるので，$W_{req} \geq W$ を満たすインスタンスの組み合わせを見つければ良い．式 (4，5，6) は式 (3) の内部式である．

$$W = \frac{L}{\lambda} + \frac{1}{\mu} \tag{3}$$

$$\rho = \frac{\lambda}{s\mu} \tag{4}$$

$$L = \frac{s^{s-1}\rho^{s+1}}{(s-1)!(1-\rho)^2}P_0 \tag{5}$$

$$P_0 = \frac{1}{\sum_{n=0}^{s-1}\frac{(s\rho)^n}{n!} + \frac{(s\rho)^s}{s!(1-\rho)}} \tag{6}$$

$W_{req} \geq W$ を満たすインスタンスを求める計算は，等価変換ルールとして，以下のようにコーディングされている．

$$
\begin{aligned}
&structure(*names,\ *lambda,\ *response),\\
&\quad \Rightarrow length(*names,\ *s),\ setInstance(*names),\\
&\qquad comMu(*names,\ *s,\ *mu),\\
&\qquad comL(*s,\ *lambda,\ *mu,\ *L),\\
&\qquad div(*L,\ *lambda,\ *C),\ div(1,\ *mu,\ *D),\\
&\qquad add(*C,*D,*W),\ greaterEq(*response,\ *W).
\end{aligned}
$$

$length(*names,\ *s)$ でリスト $*names$ の要素数から VM 台数 $*s$ を計算する．$setInstance(*names)$ でインスタンスのタイプを $*names$ 中の要素に代入する．$comMu(*names,\ *s,\ *mu)$ で平均サービス率 μ を計算する．$comL(*s,\ *lambda,\ *mu,\ *L)$ で式 (5) の計算であるリクエストが処理されるまでの平均待ち時間 L が求められる．式 (4，5，6) に関する等価変換ルールは省略する．$div(*L,\ *lambda,\ *C)$ で式 (3) 中の L/λ が計算され，$div(1,\ *mu,\ *D)$ で $1/\mu$ が計算され，$add(*C,\ *D,\ *W)$ でそれらの和である平均サービス応答時間 W が計算される．$greaterEq(*response,*W)$ は $W_{req} \geq W$ に関する条件を判定している．

ⓒ 2015 Information Processing Society of Japan

表 2 インスタンスタイプと平均サービス率 μ

インスタンスタイプ	平均サービス率 μ
t2.micro	21
t2.small	25
t2.medium	10
t2.large	15
m4.large	8
m4.xlarge	20
m4.2xlarge	35
m4.4xlarge	49
m4.10xlarge	70
m3.medium	2
m3.large	8
m3.xlarge	17
m3.2xlarge	32

5. 動作実験

5.1 前提

クエリ節 Q^{F1} の置き換えでは，VM1 台あたりのサービス稼働率を定めなければならないので，Amazon EC2 で与えられている信頼性を基に，VM1 台あたりのサービス稼働率を 99.95%とする．クエリ節 Q^{F2} の置き換えでは，*structure* 述語の原子論理式中の変数 $*vms$ に代入されるインスタンスを定めなければならないので，Amazon EC2 で利用可能なインスタンスタイプのうち，表 2 に示す汎用タイプとして定義されている T2 ファミリー，M4 ファミリーおよび M3 ファミリーを用いる．提案方法では，平均サービス率 μ の精度の高さを求めていないので，平均サービス率 μ は ApacheBench の実測値を基に現実的な値を設定した．

5.2 実験内容

5.2.1 システム構成の検証機能の処理内容

3.2 章で示した原子論理式の連言 S を入力とする．まずクエリ節 Q^{F1} を示す．*checkSLA* 述語の原子論理式は，Web3 層モデルの各レイヤごとに作られる．例えば，プレゼンテーション層に関連する原子論理式に注目する．*elb* 述語と *ec2* 述語の原子論理式の繋がり方より，プレゼンテーション層は，EC2 インスタンス 2 台で処理を並列化していることが分かる．そのため $sla(\mathrm{pres}, 99.9999)$ から，VM2 台構成で 99.9999% のサービス稼働率があるかを制約条件として与えることになる．この制約条件は，*checkSLA* 述語の原子論理式により $checkSLA(99.9999, 2, *r1)$ と記述される．他のレイヤに関しても *checkSLA* 述語の原子論理式を作成すると，以下のクエリ節 Q^{F1} が作り出される．

$$ans([*r1, *r2, *r3]) \leftarrow checkSLA(99.9999, 2, *r1), checkSLA(99.99, 2, *r2), checkSLA(99.99, 1, *r3).$$

次にクエリ節 Q^{F1} の置き換えを説明する．クエリ節 Q^{F1} に等価変換ルールが適用されることで，3 つの変数が具体化される．$checkSLA(99.9999, 2, *r1)$ に注目して説明する．$multiplier(2, 99.95, *mu)$ は $*mu = 0.00000025$ であり，$sub(1, 0.00000025, *R)$ は $*R = 0.99999975$ である．つまり，99.999975% $\geq$ 99.9999% なので，$*r1 = true$ となる．その結果，クエリ節 Q^{F1} は以下の状態になる．

$$ans([true, *r2, *r3]) \leftarrow checkSLA(99.99, 2, *r2), checkSLA(99.99, 1, *r3).$$

クエリ節の変換を繰り返すと，以下の状態が得られる．

$$ans([true, true, false]) \leftarrow$$

データ層のシステム構成が制約条件を満たさないため，連言 S の修正が求められる．制約条件を満たすためには，少なくとも VM2 台で構成しなければならないので，ここではデータ層も VM2 台の並列構成とした式に書き換えた．

5.2.2 マシンスペックの選定機能の処理内容

まずクエリ節 Q^{F2} を示す．クエリ節 Q^{F2} は 3.2 章で示した式からではなく，前章で新たに作られた原子論理式の連言から作られる．例えば，アプリケーション層に関連する原子論理式に注目する．*elb* 述語と *ec2* 述語の原子論理式の繋がり方より，アプリケーション層は，EC2 インスタンス 2 台で処理を分散していることが分かる．そのため $layerRequest(\mathrm{appl}, 30, 0.04)$ から，VM2 台のサービス窓口で 0.04 秒の平均サービス応答時間があるかを制約条件として与えることになる．この制約条件は，*structure* 述語の原子論理式により $structure([*n3, *n4], 30, 0.04)$ と記述される．他のレイヤに関しても *structure* 述語の原子論理式を作成すると，以下のクエリ節 Q^{F2} が作り出される．

$$ans([*n1, *n2, *n3, *n4, *n5, *n8]) \leftarrow structure([*n1, *n2], 50, 0.03), structure([*n3, *n4], 30, 0.04), structure([*n5, *n8], 20, 0.05).$$

次にクエリ節 Q^{F2} の置き換えを説明する．クエリ節 Q^{F2} に等価変換ルールが適用されることで，6 つの変数が具体化される．$structure([*n1, *n2], 50, 0.03)$ に注目して説明する．$length([*n1, *n2], *s)$ は，リストの要素数が 2 個なので，$*s = 2$ となる．$setInstance([*n1, *n2])$ の変数 $*n1$ と $*n2$ は，インスタンスが 13 個存在するので，$169 (= 13^2)$ 個に分けられる．169 個のパターンの中から制約条件を満たす組み合わせを見つけ出す．ここから $*n1 = \mathrm{m3.2xlarge}$ と $*n2 = \mathrm{m4.4xlarge}$ の場合で説明する．$comMu([\mathrm{m3.2xlarge}, \mathrm{m4.4xlarge}], 2, *mu)$ は，m3.2xlarge の平均サービス率が 32 で m4.4xlarge の平均サービス率が 49 なので，$*mu = 40.5$ となる．$comL(2, 50, 40.5, *L)$ は，式 (5) に従い計算されるため，$*L = 0.760015$ となる．$div(0.760015, 50, *C)$ の変数 $*C$ は 0.015200 となり，

$div(1, 40.5, *D)$ の変数 $*D$ は 0.024691 となる．従って，平均サービス応答時間 $*W$ は 0.039892 となる．$0.03 < 0.039892$ であり *greaterEq* 述語の制約を満たさないので，$*n1 =$ m3.2xlarge と $*n2 =$ m4.4xlarge の組み合わせは，顧客要求を満たさないとなる．その他 168 個の計算は省略するが，169 個のうち 10 個の組み合わせが制約を満たす．他の *structure* 述語の原子論理式に関して同様に処理を実行すると，*ans* 述語の全ての基礎原子論理式 4826809$(= 13^6)$ 個の中から 31240 個の基礎原子論理式が得られる．

5.2.3 結果

クエリ節 $\mathbb{Q}^{F2}$ の *ans* 述語の基礎原子論理式を基に，原子論理式の連言 $\mathbb{S}$ 中の変数集合 $\{*n1, *n2, *n3, *n4, *n5, *n8\}$ に関する基礎代入 δ が 31240 個得られる．その結果，31240 個の原子論理式の連言 $\mathbb{G}$ が出力される．

6. 議論とまとめ

6.1 命題論理式による仕様記述との違い

本論文と同様に論理式により制約条件を記述する方法として，命題論理式に基づく方法 [5], [10] が議論されている．命題論理式において問題は，必ず基礎例の有限集合で記述しなければならない．その基礎例数をクエリ節で例えると，変数に対して m 個の基礎代入が与えれていて，変数が n 個のとき，n^m 個の基礎クエリ節が存在する．この n^m 個の基礎クエリ節を命題論理式で作らなければならない．そして計算量は，n^m 個の基礎例に関する全ての真偽の組み合わせから，全体が真になる基礎例を見つけ出すので，$O(n^{2m})$ となる．一方，述語論理式では，クエリ節中に変数が含まれている状態で計算を進められるため，命題論理式よりも無駄な計算を減らすことができる．

6.2 形式手法としての有効性

形式手法 [9] は，論理学やプログラム意味論などの数学を基礎とするシステム開発手法であり，数学的にシステムの正当性を保証する枠組みである．形式手法では次の 2 つの要素を満たすことが重要とされている．

- 数学的に仕様を記述する【形式仕様記述】
- 仕様を基にシステムを検証する【形式検証】

提案方法では，顧客要求に関する制約条件を原子論理式の連言により定義している．原子論理式の正しさは，確定節により宣言的に意味付けられた述語を基に，明確に定義されており，その連言の正当性を数学的に証明できる．そのため提案方法は形式仕様記述である．形式検証に関して，提案方法では，クエリ節の置き換えによる計算により，システム構成の検証やマシンスペックの選定を行っている．クエリ節は，原子論理式の連言に基づき作り出され，その計算の正当性は，同じ述語の意味に依存している．そのため提案方法の機能は形式仕様に基づいた形式検証である．

6.3 まとめ

本論文では，顧客要求に基づき作られる原子論理式の連言を入力としたクラウドブローキング方法を提案し，その方法に基づくブローキングツールを開発した．そして，そのツールの動作実験として，AWS の汎用インスタンスタイプを対象する VM の性能スペックの選定を行った．

今後は，システム内のアプリケーションのデータの流れや準拠法，クラウドサービス利用費用などのソフトウェア仕様を記述するために原子論理式の連言の拡張を行う．

参考文献

[1] Akama, K. and Ekawit, N.: Formalization of the Equivalent Transformation Computation Model, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 10, No. 3, pp. 245–259 (2006).
[2] Amazon web service: AWS CloudFormation, Amazon.com (online), available from ⟨http://aws.amazon.com/cloudformation/⟩ (accessed 2015-07-21).
[3] Amazon web services: Amazon EC2, Amazon.com (online), available from ⟨http://aws.amazon.com/ec2/⟩ (accessed 2015-07-21).
[4] 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/⟩（参照 2015-07-21）.
[5] Chen, C., Yan, S., Zhao, G., Lee, B.-S. and Singhal, S.: A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises, *The 5th International Conference on Cloud Computing (CLOUD 2012)*, IEEE, pp. 883–890 (2012).
[6] Christy Petty and Rob van der Meulen: Gartner Says Cloud Consumers Need Brokerages to Unlock the Potential of Cloud Services, Gartner, Inc. (online), available from ⟨http://www.gartner.com/newsroom/id/1064712⟩ (accessed 2015-08-3).
[7] Kleinrock, L.: *Theory, Volume 1, Queueing Systems*, Wiley-Interscience (1975).
[8] Liu, X., Heo, J. and Sha, L.: Modeling 3-Tiered Web Applications, *The 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Washington, DC, USA, IEEE Computer Society, pp. 307–310 (2005).
[9] Monin, J. F.: *Understanding Formal Methods (FACIT)*, Springer (2003).
[10] Sirbu, A. and Hoffmann, J.: Towards Scalable Web Service Composition with Partial Matches, *2008 IEEE International Conference on Web Services (ICWS 2008)*, IEEE Computer Society, pp. 29–36 (2008).
[11] 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).
[12] Sundareswaran, S., Squicciarini, A. C. and Lin, D.: A Brokerage-Based Approach for Cloud Service Selection, *The 5th IEEE International Conference on Cloud Computing (CLOUD 2012)*, IEEE, pp. 558–565 (2012).
[13] 三浦克宜，齋藤篤志，玉家武博，棟朝雅晴：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，情報処理学会第 103 回 MPS 研究会， Vol. 2015-MPS-103, No. 18, pp. 1–6 (2015).

ⓒ 2015 Information Processing Society of Japan