クラウドの信頼性評価のためのモデル化手法

町田　文雄

クラウドの信頼性を解析的に評価するアプローチについて紹介する．マルコフ連鎖や確率ペトリネットなどの状態空間モデルを用いてクラウドの信頼性を解析する上では，クラウドのスケーラビリティと複雑性への対処が課題となる．スケーラビリティに対しては，大きなモデルを複数のサブモデルに分割して解析するモデル分解手法が注目される．クラウドの品質評価にモデル分解手法を用いた研究例を紹介する．また，複雑なシステムのモデルをいかに生成するかという課題に対し，筆者が取り組むSystems Modeling Language（SysML）を用いたコンポーネントベースモデリング手法の研究を紹介する．

キーワード：クラウド，信頼性評価，モデル分解（decomposition），確率報酬ネット（SRN），SysML

1. はじめに

クラウドは企業や個人に対する安価で容易なIT資源調達手段としてだけではなく，政府機能や社会基盤を担う重要なインフラストラクチャとして注目され始めている．米国政府は“Cloud First”ポリシーの下，IT支出の4分の1にあたる200億ドルをクラウドへの移行対象とし，データセンタのクラウド化を進めている[1]．我が国においても霞が関クラウドや自治体クラウドの構想が徐々に具体化し，クラウドを利用した地域行政サービスの統合化，集約化の事例が出始めている[2]．企業活動を含め，我々の生活を支える多くの情報システムがクラウドに依存しつつある．

クラウドの利用が広まるにつれ，クラウドの信頼性に対する関心も高まっている．2011年3月のGmail障害，同4月のAmazon EC2の障害時には，多くのユーザやWebサイトが影響を受け，各種メディアでも大きく取り上げられた．GoogleやAmazonなどのパブリッククラウドではSLAで99.5%～99.95%の可用性を保障しているが[3][4]，ミッションクリティカル性の高いアプリケーションの運用には十分とはいえない．SLA違反によるペナルティはユーザの損失をカバーするものではないため，パブリッククラウド利用に伴って発生した損害は自己責任となる．一方，ITサービスベンダが提供するプライベートクラウドでは，高信頼性を付加価値の一つとして訴求しているが，多くの場合詳細が明らかではない．クラウドは様々なハードウェアやソフトウェア，運用技術が組み合わさって構成されており，冗長構成や障害検出法などの技術一つでクラウドの信頼性が保障されるわけではない．特に，大規模な情報システムの信頼性は，運用や保守フェーズに大きく依存するが[5]，クラウドサービスプロバイダにおける運用管理はユーザからは通常見えない．この不可視性は，クラウドの信頼性に対するユーザの不安を助長する．クラウドサービスプロバイダはクラウドのアーキテクチャや運用管理に基づいて信頼性の定量的な評価を行い，ユーザに対して開示していくことが求められる．

クラウドのアーキテクチャや運用を反映して品質を定量的に評価する手段として，解析モデルに基づく品質評価手法が有用である．解析モデルによる評価手法はシステム設計時の品質評価に利用されるが，特に実システムを使った評価が難しい信頼性評価においてはその有用性が高い．複雑なシステム構成や依存関係のある運用手順をモデル化することにより，システムを動かすことなくその構成や振る舞いを評価できる利点がある．モデルを解析することにより可用性やMTTFなどのシステムの信頼性指標を見積もり，アーキテクチャや運用手順の有効性を定量的に示せる．また，システム構成上のボトルネック特定や，システム構成変更や運用ポリシの変更による効果の推定など，システムの高信頼化に向けた改善にも応用できる．

本稿では，クラウドの信頼性評価のための解析モデルの応用について述べ，大規模なシステムのモデルを解析するためのモデル分解手法，および，システムの運用手順を反映した解析モデルを効率よく生成するモデル合成手法について紹介する．

まちだ　ふみお
NEC Service Platforms Research Laboratories
〒211-8666 川崎市中原区下沼部1753

2. システム信頼性評価のための解析モデル

システム信頼性評価のための解析モデルを大きく分類すると，組み合わせモデルと状態空間モデルに分けられる．故障木や信頼性ブロック図などの組み合わせモデルは，システムの構成要素の組み合わせをモデル化し，システム全体の信頼性やボトルネック解析に利用する．複数の部品やデバイスで構成されるハードウェア製品の信頼性評価に広く利用されており，Relex[6]やBlockSim[7]など，多数の商用製品でサポートされている．一方，マルコフ連鎖や確率ペトリネットなどの状態空間モデルはシステムの動的な振る舞いを確率過程で捉え，復旧処理のカバレッジ，異なる故障状態，フェイルオーバーなど，組み合わせモデルでは表現の難しい状態変化を表現して信頼性評価に反映できる．状態空間モデルによる信頼性解析は一部の商用製品でもサポートされているが，大学で開発された研究用のツールが多く知られている．組み合わせモデルと比較してモデルの生成が容易ではないため，大規模で複雑なシステムへの適用にはまだ課題がある．

クラウドの信頼性解析では，以下のクラウドの性質から，状態空間モデルの応用が必須になると考えられる．

- 仮想マシンの生成や削除など，システムの構成や状態が動的に変化する
- 仮想マシンやアプリケーション，ミドルウェアなど，様々なソフトウェアが相互に依存している
- クラウドのインフラストラクチャを維持するための運用管理が信頼性に大きな影響を与える

特に，運用管理の有効性を定量的に評価するためには，運用手順に伴って変化するシステムの状態を状態空間モデルで適切に捉えることが重要である．例えば，夜間に計画的にサーバを停止してメンテナンス作業を行う運用が行われている場合，計画停止によるダウンタイムやメンテナンス作業後の状態などはシステムの可用性に影響を与える重要なファクタである．これらを単純な組み合わせモデルで捉えることは難しい．マルコフ連鎖やセミマルコフ過程などの状態空間モデルを用いれば，定期的なシステム停止に伴う状態変化を捉えてシステムの信頼性評価に反映できる[8]．

上記に挙げたクラウドの性質は，いずれも状態空間モデルを用いて表現可能であるが，クラウドのスケーラビリティへの対応と，大規模な状態空間モデルの生成手法が大きな課題として残る．クラウドのインフラストラクチャは数千，数万台規模のサーバで構成されるため，信頼性解析のための解析モデルにもスケーラビリティが求められる．状態空間モデルでは，システムが大規模になると状態爆発の問題に直面してしまう．状態空間モデルをクラウドの規模に対応させるためには，より抽象度が高く，より小さなモデルを用いていかに近似するかが課題になる．

一方で，より精度の高い信頼性評価のためには，システムの詳細な構成や運用手順を状態遷移モデルに反映させなければならない．通常システムの詳細に関する専門的な知識はシステムを熟知したシステム管理者が持っているが，システム管理者が状態遷移モデルを使いこなすことは難しい．正確な解析モデルを作成するためには，解析モデルに関する専門知識が求められる．さらに，専門家であっても，クラウドのような大規模なシステムに対するモデルをゼロから作りあげることは容易ではない．

これらの課題に対し，現在，状態空間モデルをいくつかの部分に分割してモデル化・解析する手法の研究が進んでいる．状態空間モデルのスケーラビリティに関しては，システム全体のモデルを互いに連携するサブモデルの組み合わせで表現する手法が注目される．また，解析モデルを効率良く生成する手法としては，我々が現在取り組んでいるコンポーネントベース可用性モデリング手法の研究が有用と考えられる．次節では，状態空間モデルのスケーラビリティを改善するモデルの分解手法を紹介し，4節で筆者が取り組むコンポーネントベース可用性モデリング手法について紹介する．

3. モデル分解手法

状態空間モデルでは対象とするシステムの規模が大きくなると状態爆発によって，解析やモデル化が困難になる．そこで，一枚岩の状態空間モデルではなく，システムの機能や性質単位にモデルを分解して解析する手法が取られる．

Ghoshらは，Infrastructure as a Service（IaaS）の品質評価のための連続時間マルコフ連鎖（CTMC）を互いに関連しあうサブモデルに分割し，解析の複雑さを解消する手法を提示している[9]．この研究ではIaaSにおけるサーバプロビジョニング時の振る舞いをランタイムモデル，ホットプールモデル，ウォーム

プールモデル，コールドプールモデル，プロビジョニング決定モデルに分割し，各サブモデル間でパラメタ値を交換し，定点反復法（Fixed point iteration）を用いて定常状態における性能指標を導出している．定点反復法は解の収束性についての証明が必要となるが，相互に依存するサブモデル間の定常状態を解析する際に役に立つ．この手法により，解析対象の状態空間を大幅に削減できるため，一枚岩のモデルと比較して高いスケーラビリティを達成できる．

通常サブモデルへの分解には近似を伴うため，いかに適切なサブモデルを定義するかがこの手法の鍵になる．多くの場合サブモデルへの分割はモデル化する対象のドメイン知識が必要だが，Lolliniらはモデル分解のための一般的なアプローチとして3段階からなる分解手法を提示している[10]．Lolliniらによる手法では，まずシステムを機能単位で分割し，次にシステムのライフタイムを複数のフェーズに分割する．最後に各フェーズにおける各機能に対し，モデルレベルの分割を行う．モデルレベルの分割ではモデルに内在する依存関係を互いに連携する複数のサブモデルに分割する．得られたサブモデルに対してはそれぞれ別々に解析を行い，解析結果をサブモデル間で交換することにより，モデル全体の解析結果を求める．モデル全体の解析には定点反復法が適用できる．

相互に依存するサブモデルへの分割は，サブモデル間の関係を維持するため，一枚岩モデルの良い近似を与える．しかし，モデルレベルの分割や定点反復法の設計は必ずしも容易ではない．これに対し，分割したサブモデルに順序付けができる（依存関係のグラフに閉路を含まない）場合，サブモデル間の関係を組み合わせモデルで表現した階層型確率モデルで近似ができる．

Smithらは複数のハードウェアとソフトウェアから構成されるブレードサーバシステムの可用性を階層型確率モデルを用いて解析している[11]．CPUやメモリなどのハードウェアデバイスやOSなどのソフトウェアを構成要素単位でCTMCでモデル化し，システムの構成を故障木を使って表現している．はじめにサブモデルを解析することでコンポーネント単位の可用性を求め，故障木に従ってシステム全体の可用性を見積もる．基本的には各コンポーネントの故障は互いに独立であることが前提となるが，ハードウェアの共有などの依存関係は故障木の機能を使って表現できる．

HaiyangらはIaaSの体感品質（QoE）を解析的に評価するため，複数のモデルを階層型に組み合わせたモデルを用いている[12]．クラウドではユーザ視点での品質評価も重要であるが，プロバイダ内で発生する仮想マシンの障害や，それに伴って発生するユーザリクエストのリダイレクト方針など，インフラの管理手法によってその品質は左右される．この研究では仮想マシンの可用性モデル，外部帯域モデル，応答時間モデル，遅延モデル，リダイレクトモデルなどを階層型に組み合わせてQoEの評価に利用している．

階層型確率モデルでは，組み合わせモデルで表現できないサブモデル間の依存関係の情報は失われるため，相互に連携するモデルと比較して近似性能は劣る．しかし，モデリングツールなどによる実装が容易であり，解析も自動化できる利点があるため，実用性の観点で優れているといえる．

4. モデル合成手法

前節で示したモデルはいずれも研究段階にあり，特定のシステムの構成情報に基づいて研究者によって設計されている．しかし，実際のクラウドのインフラストラクチャは，プロバイダによってそのアーキテクチャや運用が異なる．プロバイダごとに異なるサービス，システム構成や依存関係，運用手順を反映したモデルが必要になる．しかし，解析モデルの専門知識を持たない情報システムのエンジニアが，複雑な情報システムの解析モデルを作成することは容易ではない．

この課題に対し，現在筆者らはシステムの特定の構成要素や振る舞いに対して再利用可能なサブモデルを定義し，それらの組み合わせによってシステム全体の近似モデルを合成して可用性を評価する，コンポーネントベース可用性モデルの研究に取り組んでいる．コンポーネントを組み合わせるモデル化手法自体は決して新しいものではない．ハードウェアシステムの信頼性評価では，ハードウェア部品の信頼性を評価するためのサブモデルが定義されており，それらの組み合わせで構成されるシステムの信頼性を評価することが一般的に行われている．ハードウェア部品に対するサブモデルはハードウェアベンダによって提供されるものもあれば，米国の標準で定義されているものもある．筆者らは同様なアプローチを大規模な情報システムの可用性解析に応用することを目指している．

クラウドを構成する要素に対するモデルのコンポーネント（以下モデルコンポーネントと呼ぶ）が定義されたとき，モデルコンポーネントに基づいて対象とす

るシステムやサービスのモデルを生成するためには，システムのアーキテクチャや構成要素間の依存関係の詳細な情報が必要である．システムの詳細な情報はシステム管理者が最も良く把握しており，多くの場合，システム設計書や運用手順書などに記述されている．システム設計書や運用手順書はシステム設計者や管理者が読んで理解できることを目的とし，自然言語やブロック図，フローチャートなどを使って書かれる．定型化された文書ではないため，設計や運用に関する詳細な情報を自動的に抽出してコンポーネントの合成に利用することは難しい．しかし，システム設計や管理の効率化の観点から，近年これらの設計をUMLやSysML[13]などのモデリング言語を用いて形式的に記述する動きがある．NECではSysMLを用いたシステム設計に力を入れており，設計情報の共有によりシステム設計の効率化，コスト削減，品質改善を目指している[14]．SysMLによってシステム構成や運用が形式的に記述されれば，これらのモデルを入力として可用性モデルの合成をより効率化できると考えられる．以下，Webアプリケーションシステムで，SysMLを用いたコンポーネント可用性モデリングのプロセスを示す．Webアプリケーションシステムはクラウドでホスティングされる典型的なアプリケーションの一つである．可用性モデルには，表現力が高く，解析のためのツールが存在する確率報酬ネット（SRN）を用いる．

図1はSysMLの内部ブロック図を使って描かれたWebアプリケーションシステムの構成を示している．システムはロードバランサプロセス，Webサーバプロセス，DBサーバプロセスから構成され，構成要素がブロック（長方形）で表現されている．各プロセスはクラスタ構成やスタンバイ構成を持つことができ，クラスタ構成はブロックの多重度を使って表現される．図1の例ではWebサーバプロセスが多重度4を持つ，すなわち，4つのサーバプロセスからなるクラスタ構成であることを表している．サーバプロセスの仮想マシン割り当てや，ハードウェアの構成なども内部ブロック図を使って表現できるが，ここでは図1の構成を中心に説明する．

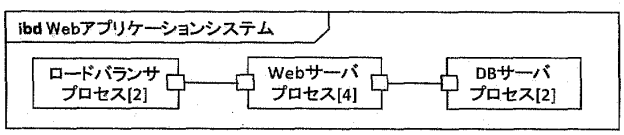

図1 Webアプリケーションシステムのプロセス構成を表すSysML内部ブロック図

内部ブロック図で表現されたプロセス構成に基づいて可用性モデルを生成するため，各ブロックに対するモデルコンポーネントを定義する．少なくとも各プロセスは稼動状態と停止状態を持つため，図2に示すSRNを基本モデルコンポーネントとして定義する．

SRNはペトリネットを使って表現される．図2のSRNではプレースP_{up}が稼動状態，P_{down}が停止状態に対応している．トークンの数はプロセスの多重度に対応し，P_{up}にあるトークンの数nが稼動状態にあるプロセスの数を表している．トランジションT_{fail}の発火によって故障が起こり，T_{recv}の発火によって復旧する．この基本モデルコンポーネントを利用することにより，図1のプロセス構成に対応する最も単純な可用性モデルが得られる．構成要素のタイプごとにモデルコンポーネントを定義することにより，より詳細な可用性モデルを内部ブロック図からの変換によって生成できる．

内部ブロック図だけでは表現の難しい詳細な状態遷移過程は，SysMLの状態遷移図を使って表現する．図3の例ではサーバの電源オフ状態，稼動状態，劣化状態，故障状態，故障検出状態，復旧状態の6状態からなるサーバプロセスの状態遷移を示している．

詳細な状態遷移の情報が与えられれば，より正確なシステムのモデル化が行える．状態遷移図からSRNへの変換は比較的容易であり，状態ノードや遷移に対

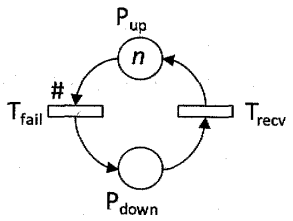

図2 基本モデルコンポーネント

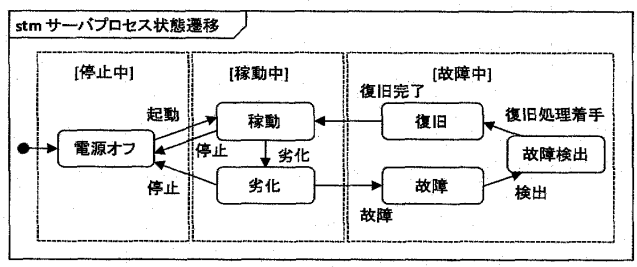

図3 サーバプロセスの詳細な状態遷移を表す状態遷移図

応した変換ルールを定義して適用することで，図4に示すSRNが得られる．

さらに，システムの状態はシステム管理のための運用操作によっても変化する．システム運用手順はフローチャートなどを使って表現されることが多いが，SysMLのアクティビティ図でも代用できる．アクティビティ図で記述された運用手順が得られれば，状態遷移図と同様な変換処理によりSRNへ変換できる．例えば，図5は毎日深夜2時にWebサーバプロセスの稼動状態をチェックし，劣化状態にあるサーバプロセスがあれば，リブートして劣化状態を解消する処理のアクティビティ図，および変換されたSRNを示している．アクティビティ図の各ノードに対してSRNへの変換ルールを定義し，ルールに基づいて自動的にSRNへ変換する．

SysMLの内部ブロック図，状態遷移図，アクティビティ図を規則に基づいて変換することで，システムの構成，構成要素の状態変化，運用手順などの側面からシステムを捉えた可用性モデルのコンポーネントが得られる．しかし，得られた各モデルコンポーネントはシステムの一側面しか表現しないため，サービスやシステム全体の可用性を解析する際には，これらのモデルを合成する必要がある．モデルコンポーネントを合成するためには，まずモデルコンポーネント間の依存関係を特定しなければならない．ここで，SysMLのAllocationの情報が活用できる．AllocationはSysMLで記述されたモデルの要素間に関連があることを示す記述規則である．依存関係が存在することのみを表し，制約条件などの詳細な記述は要求されない．したがって，Allocationの情報を元に関連するモデルコンポーネントを特定できるが，関連するモデルコンポーネントの合成には依存関係の詳細に関する追加の情報が必要である．

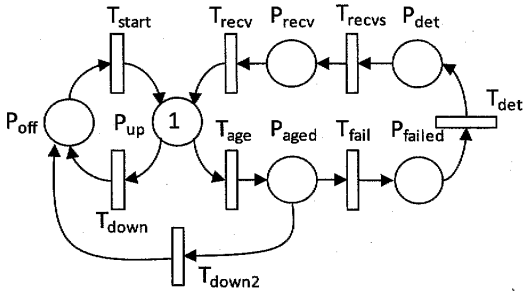

図4 サーバプロセスの状態遷移図から得られたSRN

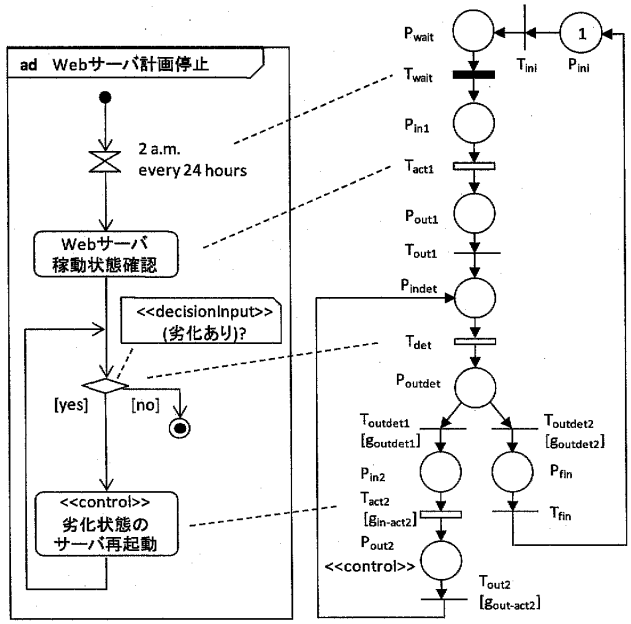

図5 夜間サーバ計画停止処理を示すアクティビティ図（左）と対応するSRN（右）

依存関係の詳細な情報を与える手段として以下の二つが考えられる．一つは，Object Constraint Language（OCL）[15]などの制約記述言語を用いて，形式的に依存関係の情報をSysML上に定義する方法である．依存関係の詳細が形式的に記述されれば，この情報に基づいてモデルコンポーネントの合成を自動化できる．しかし，状態空間モデルへの自動的な変換を実現するためには，厳密なSysMLモデルと誤りのない制約記述が必要となり，SysMLによるシステム記述の柔軟性を著しく損なう．複雑なシステムのラフスケッチを得るための手段としてのAllocationの機能が活かせない．もう一つは，モデルを合成する過程でシステム管理者から直接依存関係に関する情報を獲得する方法である．定型化されていない構成要素間の依存関係やシステム固有の制約などを，システム管理者とのインタラクションによって個々に解決していく．状態空間モデルの合成は半自動化できるが，システム管理者にはモデルコンポーネントの依存関係を解決するための知識と労力が求められる．

二つの方法はSysML上ですべての情報を与えるか，必要に応じて情報を追加していくか，という違いがあり，SysML設計の労力と，モデル合成の労力がトレードオフの関係にある．どちらの手法を取るかは，システムの性質やシステム管理者のスキルレベルに応じて判断する必要がある．

モデルコンポーネントの合成によってシステム全体を表現する可用性モデルが得られれば，可用性指標を

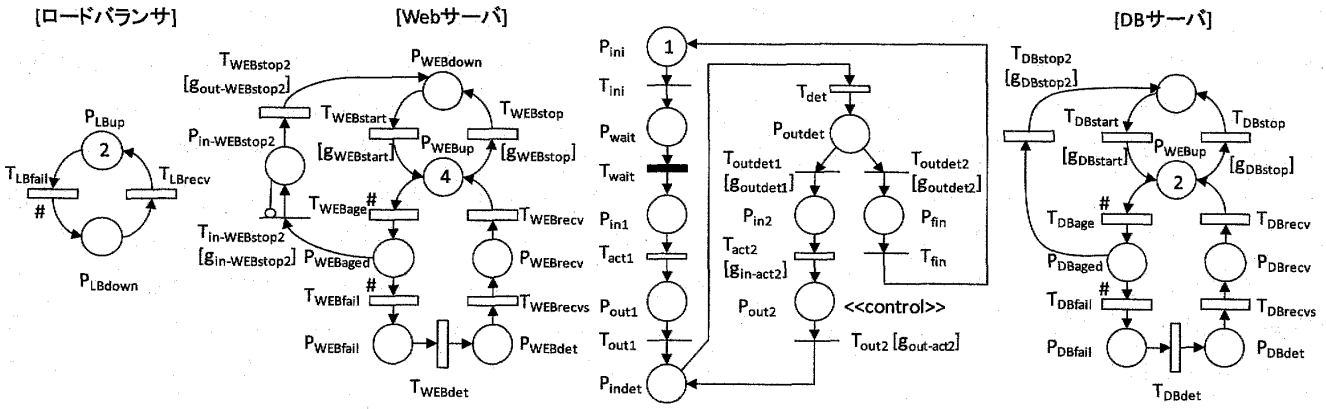

図6 Webアプリケーションシステムの SysML 記述から合成された SRN

表1 ガード関数・報酬関数

関数	定義
$g_{outdet1}$	if (#$P_{WEBaged}$>0) then 1 else 0
$g_{outdet2}$	if (#$P_{WEBaged}$==0) then 1 else 0
$g_{in\text{-}act2}$	if (#$P_{in\text{-}WEBstop2}$==1) then 1 else 0
$g_{out\text{-}act2}$	if (#$P_{in\text{-}WEBstop2}$==0) then 1 else 0
$g_{in\text{-}WEBstop2}$	if (#P_{in2}==1) then 1 else 0
$g_{out\text{-}WEBstop2}$	if (#P_{out2}==1) then 1 else 0
$g_{WEBstart}$, $g_{DBstart}$	1
$g_{WEBstop}$, g_{DBstop}, $g_{DBstop2}$	0
報酬関数	if (#(P_{LBup})>=1 and (#(P_{WEBup})+#($P_{WEBaged}$))>=3 and (#(P_{DBup})+#(P_{DBaged}))>=1) 1 else 0 end

表2 デフォルトパラメタ値

パラメタ	トランジション	値
ロードバランサ MTTF	T_{LBfail}	1year
ロードバランサ MTTR	T_{LBrecv}	2hours
Web サーバ平均劣化時間	T_{WEBage}	1week
劣化 Web サーバ MTTF	$T_{WEBfail}$	2weeks
サーバ障害平均検出時間	T_{WEBdet}, T_{DBdet}	5mins
サーバ平均復旧着手時間	$T_{WEBrecvs}$, $T_{DBrecvs}$	30mins
Web サーバ平均復旧処理時間	$T_{WEBrecv}$	1hour
Web サーバ平均起動/停止時間	$T_{WEBstart}$, $T_{WEBstop}$, $T_{WEBstop2}$	1min
Web サーバ計画停止間隔	T_{wait}	1day
平均アクション起動時間	T_{act1}, T_{det}, T_{act2}	1sec
DB サーバ平均劣化時間	T_{DBage}	2months
劣化 DB サーバ MTTF	T_{DBfail}	4months
DB サーバ平均復旧処理時間	T_{DBrecv}	2hours
DB サーバ平均起動/停止時間	$T_{DBstart}$, T_{DBstop}, $T_{DBstop2}$	5mins

評価する報酬関数を定義し，必要なモデルのパラメタを与えてモデルを解析することで可用性を評価できる．図6は図2，図4，図5のモデルコンポーネントをすべて合成して得られた SRN の一例を示している．表1に関連するガード関数，および報酬関数の定義を示す．報酬関数では三つ以上の Web サーバプロセス，

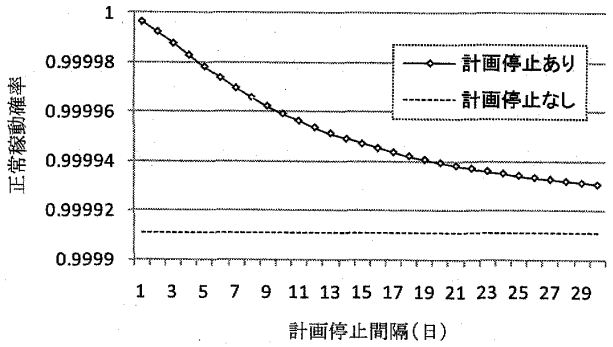

図7 計画停止間隔を変化させた際の正常稼働確率の変化

一つ以上のロードバランサプロセスと DB サーバプロセスがすべて動作している状態を正常な稼動状態と定義している．ガード関数の定義はモデルコンポーネント合成の過程でシステム管理者とのインタラクションによって獲得することを前提としている．表2に示すモデルのパラメタ値を与えて定常状態での報酬関数の期待値を求めることにより，Web アプリケーションシステムが正常に機能している確率を 99.99966% と見積もれる（SPNP[16]を使って評価）．

パラメタ値を変化させて結果の違いを比べることにより，システム設計や運用ポリシの改善に役立てることができる．図7は計画的な Web サーバの再起動を1日から1カ月の間で変化させた際の正常稼働確率の変化を示している．より頻繁にサーバの劣化状態をチェックし，サーバプロセスを再起動することで，障害を事前に回避して正常稼動確率を高めることができる．

上記の例で示したように，SysML で記述されたシステムの設計情報を再利用することにより，複雑な可用性モデルを効率よく生成し，システムの品質改善に役立てることができる．UML などから解析モデルを生成してシステムの品質評価に応用する研究例は多数

存在するが，多くの場合，モデルの完全な自動生成を目的とし，UMLなどに追加の表記法を導入している．しかし，このような追加表記法は，UMLなどによるシステム記述の柔軟性を損なうだけでなく，解析可能な対象システムを制限してしまう．我々の研究では，解析モデルの完全な自動生成ではなく，入力となるSysMLの柔軟性を維持し，解析モデルの生成に必要な情報をインタラクティブに獲得していく方式の設計を目指している．このアプローチにより，定型化されていない複雑なシステム構成や運用にも柔軟に対応した解析モデルの生成が行えると考えている．

5. おわりに

クラウドのインフラストラクチャは大規模で複雑なシステム構成と振る舞いを持つが，複数の共通する構成要素や状態変化，振る舞いで構成される特徴がある．解析モデルをクラウドの品質評価に応用する際には，いかにこの共通する構成要素に基づいてモデルを作成し，いかに共通する計算処理を省いて解析結果を得るかが課題である．本稿では，大規模なシステムを表現するモデルをサブモデルの集合に分割して解析するモデル分割のアプローチと，システムの構成要素単位でモデルをコンポーネント化し，モデルコンポーネントの合成によってシステム全体のモデルを獲得するアプローチについて紹介した．これらのアプローチはトップダウンかボトムアップかの違いはあるが，複雑なモデルを複数のサブモデルを使って扱うことによりスケーラビリティに対処するという点で共通している．モデルの合成，効率的な解析のためのモデル分解，双方の観点で扱いやすいモデルコンポーネントの設計は今後の興味深い研究テーマの一つと考えられる．

参考文献

[1] V. Kundra, Federal Cloud Computing Strategy, http://www.cio.gov/documents/Federal-Cloud-Computing-Strategy.pdf, 2011.

[2] 奈良県下7市町に基幹システムのクラウドサービスを提供，http://www.nec.co.jp/press/ja/1012/0701.html, 2010.

[3] Amazon EC2 SLA, http://aws.amazon.com/ec2-sla/

[4] App Engine for Business SLA, http://code.google.com/intl/ja/appengine/business/sla.html

[5] 経済産業省，情報システム・ソフトウェアの信頼性及びセキュリティの取組強化に向けて（中間報告書），http://www.meti.go.jp/press/20090528001/20090528001-2.pdf, 2009.

[6] Relex, http://www.relex.com

[7] Reliasoft Blocksim, http://www.reliasoft.com/BlockSim

[8] T. Dohi, K. Goseva-Popstojanova and K.S. Trivedi, Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule, In Proc. of Pacific Rim Int. Symp. on Dependable Computing, pp. 77-84, 2000.

[9] R. Ghosh, K.S. Trivedi, V.K. Naik and D.S. Kim, Performability analysis for Infrastructure-as-a-Service cloud : An interacting stochastic models approach, In Proc. of Pacific Rim Int. Symp. on Dependable Computing, 2010.

[10] P. Lollini and A. Bondavalli, A Decomposition-Based Modeling Framework for Complex Systems, IEEE Transactions on Reliability, Vol. 58, No. 1, 2009.

[11] W.E. Smith, K.S. Trivedi, L.A. Tomek and J. Ackaret, Availability analysis of blade server systems, IBM System J., Vol. 47, No. 4, 2008.

[12] H. Qian, D. Medhi and K.S. Trivedi, A hierarchical model to evaluate quality of experience of online services hosted by cloud computing, In Proc. of Int. Symp. on Integrated Network Management, 2011.

[13] OMG Systems Modeling Language (OMG SysML) Version 1.2, http://www. omg. org/spec/SysML/1.2/

[14] 伊豆倉さやか，向剣文，榊啓，木村大地，矢野尾一男，システムモデルベースSI支援環境による性能・可用性評価，情報処理学会研究報告 EVA, pp. 1-8, 2010.

[15] OMG Object Constraint Language (OCL), http://www.omg.org/spec/OCL/2.2

[16] C. Hirel, B. Tuffin and K.S. Trivedi, "SPNP : Stochastic Petri Nets. Version 6.0," In Proc. of TOOLS 2000.