クラウドリソース監視情報の提供量に応じた動的リソース融通の提案

坂口和也[†1]　前田香織[†1]　近堂徹[†2]　相原玲二[†2]

クラウドサービスの提供が増加する中，障害対応やサービス処理の負荷分散に対応するため，単一クラウド内やクラウドをまたがってリソースを融通するニーズが高まっている．リソース融通においてはリソース融通に必要なリソース監視情報を交換することが前提になっているが，実際には事業者ごとに提供するリソースの種類が異なることが多く，障害，監視装置の機能の制限，運用ルールなどで必要なリソース監視情報を相互に得られないことが多い．そのため，クラウド間連携の目的であるユーザへのサービス品質を維持しつつ，各クラウドの条件の元クラウド全体のリソース使用の効率化が図られなくなる．そこで本研究ではリソース融通に必要なリソース監視情報をすべて収集できない場合にどのくらい非効率な状態になるかを示し，本来のクラウド間連携の目的をできるだけ達成するよう，複数のリソース群間で動的にリソース配置をする方法を提案する．最初に，利用できるリソース監視情報が限られていることを前提としてクラウドリソースを融通しあうシステムをモデル化する．次に，リソース監視情報の種類をしぼり，具体的な例を用いてリソース監視情報の提供度合いに応じてリソース融通をするアルゴリズムを提示し，シミュレーションによりリソース融通のプロセスを示す．さらに，シミュレーション結果から，提案方法が提案のアルゴリズムがないときに比べてどれくらいリソース融通が有効かを示す．

A Proposal of a Cloud Resource Collaboration Depending on Amount of Monitoring Resource Information

KAZUYA SAKAGUCHI[†1]　KAORI MAEDA[†1]　TOHRU KONDO[†2]
REIJI AIBARA[†2]

Demand for resource collaboration in a single cloud or inter clouds will increase due to disaster recovery and load balancing of service processing of cloud services. Exchange of resource monitoring information is assumed to be necessary for the resource collaboration. However, necessary monitoring resource information is not always exchanged in fact because of difference between kind of resource information, restriction of monitoring devices, system troubles, operational policies and so on. Consequently, resource optimization of all resources over clouds is difficult with maintaining service quality to users. In this research, we show inefficiency in resource allocation if all of the monitoring resource information is not available. Also, we propose a model of dynamic resource allocation over multiple resource groups. First, we propose a model of a system that collaborates cloud resources in the case that all monitoring resource information are available. Second, we show a process of resource allocation depending on availability of resource information in the case of specific resource information by a simulation. Moreover, we show an effectiveness of our proposed method by some simulation results on comparison without the proposal.

1. はじめに

多様なシステムの柔軟性，俊敏性，安定性をエンドユーザの要求に応じて提供できることはクラウドサービスのメリットでもあるが，実際にそれを可能するためにはクラウド内の拠点間またはクラウド間でのリソースの融通が不可欠となる．近年，特に大規模災害に対応することや社会基盤として信頼性の高いシステムとして稼働すること，また，クラウド利用者の多様なニーズに対して経済的なシステムを構築したいという要求が高まっており，複数のクラウドを用いるシステムづくりが求められる．

このようなクラウド間の連携によりリソースを融通する技術に関して，グローバルクラウド連携基盤技術フォーラム（GICTF）などでインタークラウドの標準化が始まった[1]．標準化ではクラウド間でリソースを融通するための連携インタフェースが共通的な仕様を規定している．一般的なクラウド間連携の目的はユーザへのサービス品質を維持しつつ，各クラウド事業者の条件の下クラウド全体のリソース使用の効率化である．この目的達成のために，リソース融通に必要なリソース監視情報（通信遅延やサーバのCPU 使用率など）は相互に情報交換が必要となる．GICTF においてもすべての情報を交換しあうことを前提としている．しかし，対象とされるリソース監視情報は多岐に渡り，各事業者でリソース監視情報を独立に収集や監視している実態から，必ずしも必要な情報が収集できるとは限らない．ネットワークインフラを提供するプロバイダとそれを利用してサーバやストレージを提供する SaaS クラウド事業者のリソース監視情報は異なるなど事業者ごとに扱うリソー

†1 広島市立大学大学院情報科学研究科
Graduate School of Information Sciences, Hiroshima City University
†2 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University

ス監視情報は異なり，共通仕様を前提とするも，運用ポリシーや監視システムの都合でクラウド間連携に必要な情報すべてを各クラウド事業者で収集することができない場合もある．これは単一クラウド内においても同様で，拠点ごとのリソース監視情報の収集が困難な場合がある．このとき，情報が不足する場合も全情報が得られることを前提としたリソース配置のアプローチでは，クラウド間連携の目的となるリソース使用の効率化が図られなくなる場合が出てくる．

そこで本研究ではリソース融通に必要なリソース監視情報をすべて収集できない場合にどのくらい非効率な状態になるかを示し，本来のクラウド間連携の目的をできるだけ達成するよう，複数のリソース群間で動的にリソース配置をする方法を提案する．最初に，利用できるリソース監視情報が限られていることを前提としてクラウドリソースを融通しあうシステムをモデル化する．次に，リソース監視情報の種類をしぼり，具体的な例を用いてリソース監視情報が得られない場合のリソース融通をするアルゴリズムを提示し，リソース融通のプロセスを示す．シミュレーションにより，提案方法が全リソース監視情報を収集できるときに比べてどれくらい有効かを示す．

リソース連携に必要な監視情報の有無によらず，複数のリソース群のリソース連携の効率化の定義は連携するリソース全体なのか，提供するクラウド事業者にとってなのか，エンドユーザにとってなのかで一意には決まらないことが一般的である．本研究では，一般的な効率化に対してその解決策を求めることを対象としているのではなく，ある（特定の）リソース使用の効率化を求めるという条件が与えられた場合，リソース監視情報が不足する際に全情報が得られた場合とに比べて，効率化の精度を下げないようなリソースの組み合わせを示すことが目的である．

以降，２章では関連研究を，３章では限られたリソース監視情報でリソース融通をするリソース配置のモデル化について述べる．４章で提案したモデルを用いてユーザの要求を満たしつつ，リソースの配置をするシミュレーションにより，提案モデルとリソース配置の有効性を示す．最後に６章でまとめと今後の課題について示す．

2. 関連研究

2.1 インタークラウド連携

大規模災害対応など社会的情報基盤の維持や優先度の高いサービスの処理を集中的に行うなど複数クラウドが相互にリソースを融通する需要が高くなると，クラウド間で技術的な仕様の共通化が必要となる．これに対して，グローバルクラウド連携基盤技術フォーラム（GICTF）[1]でインタークラウドの標準化が始まり，2011 年に情報通信技術委員会のクラウドコンピューティングアドバイザグループにおいて，ITU-T SG13 での勧告化が進んでいる[2].

クラウド間のリソース融通に関して，[3]では動的配置法を用いてプライベートクラウドの使用率を一定とし，ハイブリッドクラウドの運用管理コストを最小とする手法を提案している．別の既存研究において，理論的な検討だけでなく，複数のクラウドでリソースを融通するシステムの提案や実装もある．例えば，[4]は複数クラウド間でディザスタリカバリを行う機構を提案し，それを実装して評価実験によりリソースの配置やスケールアウト時間について評価している．[5]はサービス需要のピーク時に他のクラウド事業者からリソースを得て負荷をオフロードするシステムの提案である．[6]では選択ポリシーに従って Web サーバ用の仮想マシン（VM）を複数のクラウドにまたがって増設するシステムを構築し，評価している．

このように複数のクラウド間でリソースを融通するためのリソースの配置方法の提案や実験的なシステムの構築は行われているものの，リソース配置の判断に必要なリソース監視情報が得られることが前提となっている．GICTF の技術仕様においても，リソース監視情報，融通しあえるリソースの仕様や条件をクラウド事業者間で必要時に交換し合うことを前提としている．しかし，実際には各事業者で提供できるリソース監視情報に制限があったり，障害等でリソース監視情報が収集できないケースもある．これは単一クラウドの複数拠点間のリソース配置においても同様のことが言える．このようにすべてのリソース監視情報が得られない場合におけるリソース融通のための技術的な解決方法が必要である．

2.2 クラウドリソースの再配置頻度の増加

IaaS 等で提供されるクラウドサービスでは VM をサーバとしてアプリケーションを提供することがほとんどであるが，さらに柔軟性や冗長性をあげるためにアプリケーションを構成する機能ブロック（モジュール）の単位で VM 化し，VM の分割や連結などでアプリケーション全体を構成することが可能である[7][8]. 著者らも先行研究でストリーム配信を VM の組み合わせで構成するプラットフォームの提案を行なっている[9].このように VM が担う処理単位が小さくなると，機能単位で VM を分散配置もしやすくなり，拠点やクラウドをまたがって配置される場合も増える．

一方，端末が移動しながら使われる場面も多くなっており，アプリケーションが保持する状態情報を保持しながらサービス実行処理を別のサーバに移動するようなケースも増えていく[10].

これらの背景から VM の分散配置やマイグレーションなどの頻度が高くなり，リソースの動的配置の頻度は増加していくことが想定される．

3. リソース配置のモデル化

3.1 レイヤ間リソース連携アーキテクチャ

クラウド間のリソース連携にはクラウド間を接続するネ

ットワークリソース，配置し合うサーバやストレージなどハードウェアリソース，さらに上位のミドルウェアやアプリケーションに関する連携というように，複数のレイヤ同士の横の連携やレイヤ間にまたがる縦の連携が必要となる．これを GICTF では多階層のインタフェースモデルとして示している[11]．これをベースに[4]でも階層的なフレームワークを構成している．本研究でも同様に階層的な連携の概念を用い，図 1 のようなアーキテクチャでリソースが配置されるものとする．このアーキテクチャでは，各リソースとそれに関するリソース監視情報が階層的に構成され，レイヤごとの横方向に，また，レイヤをまたがって縦方向にリソース監視情報を互いに交換することによって，理想的なリソース群間のリソースを配置する．GICTF の階層化モデルでは図 1 のリソース群が 1 つのクラウドを表すが，ここでは単一クラウド内の拠点（データセンタなど）もリソース群として対象とし，クラウド間，クラウド内のリソース群の連携を対象とする点が GICTF のモデルと異なる．

運用上のルールや監視機器の都合，さらに障害などによって，レイヤごとやレイヤをまたがって交換されるリソース監視情報は必ずしも相互に交換できない場合もある．本研究では以降で情報が不足する場合をモデル化し，その制約の中でユーザのサービス品質を維持するようにリソースを配置するアルゴリズムを示す．

3.2 対象のモデル化

エンドユーザの品質要求を維持しつつ，階層化されたリソースに対するリソース監視情報が欠損する場合でもリソース群間でできるだけ効率的な使用リソースの配分，つまりリソース配置を行うことが必要である．本節では対象とするリソース群などをモデル化し，目的関数を求める．

(1) 提供リソース

単一クラウド，インタークラウドを含め，合計で n 個のリソース群（クラウド事業者）が現在（一定期間内に）提供できるリソース（サーバ，ストレージ，ネットワークインフラなど）を $S=\{s_1,s_2,s_3 \ldots s_n\}$と表す．このとき，ユーザのサービス品質を維持できる場合のみが対象となるリソース s_i となるものとする．

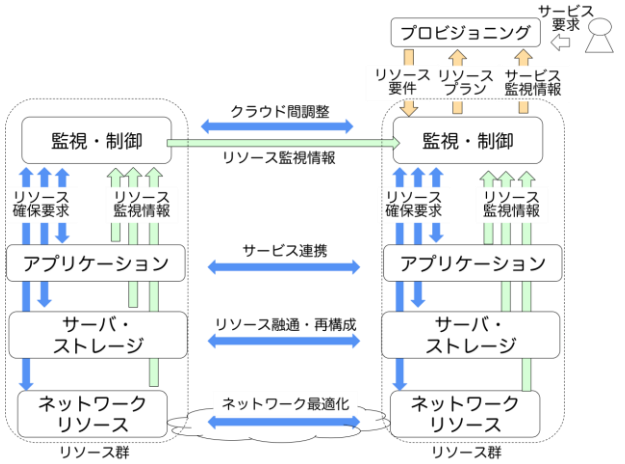

図 1　レイヤ間リソース連携

Figure 1　Resource Collaboration over Layers

(2) リソース監視情報

取得できるリソース監視情報は様々で，例えば RTT やパケットロス率などのネットワークインフラに関する情報，CPU 使用率やメモリ使用率などのサーバリソースに関する情報，ストレージの使用量などがある．これらリソース監視情報をネットワーク，サーバ，ストレージの大きく 3 つに大別し，これを $I=\{I_N, I_{SA}, I_{ST}\}$と表す．

(3) リソース使用状況

これらの 3 つの監視情報から，リソース s_i のリソースの使用状況を 3 項組の $r_i=(I_{Ni}, I_{SAi}, I_{STi})$と表し，S に含まれるサーバそれぞれのリソースの使用状況全体は $R=\{r_1,r_2,r_3 \ldots r_n\}$ ($r_i=(I_{Ni}, I_{SAi}, I_{STi})$, $1 \leq i \leq n$))と表すこととする．R は I の定義するリソース監視情報を持つ．

ここで，I_x ($x=\{N, SA, ST\}$)の値は，リソースの使用が小さい(提供できるリソースが多い)場合に小さい値をとり，$0 \leq I_x \leq 1$となる．ただし，監視情報が取得できずリソースの使用状況を提供できない場合は $I_x=2$　とする．

(4) 目的関数

本研究ではリソース使用状況が最小となるリソース s_i の組み合わせ(集合)を求めることを目的とし，式(1)で表す．3 項組から全体最小化を導出する方法はここでは定義しない．

目的関数：

$min(\sum_{r_i \in R} r_i)$　($r_i=(I_{Ni}, I_{SAi}, I_{STi})$, $(1 \leq i \leq n)$ … (1)

制約条件：

$0 \leq I_x \leq 1$　を満たす Ix のみが使用される

4. リソース監視情報提供度による優先的リソース配置の提案

3 におけるモデル化はリソース監視情報をリソース配置に用いるためのモデル化であり，目的関数で求められるリソースの組み合わせ（リソースの配置）はリソース監視情報が不足した状態でもリソース全体として使用するリソースを最小化するものである．このとき，どのリソース群からサービス監視情報を提供されているかを考慮してリソース配置をするわけではないので，情報提供の有無に関わらず，リソースが配置される．

しかし，実際には情報提供にも監視や提供のためにコストがかかるのが一般的であり，情報提供に見合うようなリソース配置を要望される．そこで，本研究では情報提供の有無によってリソース配置を優先的に行うことを提案する．例えば従量課金制クラウド事業者は提供できるリソースの範囲でできるだけ多くリソースを提供することを求めるので，それが反映できるようにリソース配置をするアルゴリズムを以降の具体的なモデルで示す．ここで「リソース配置」とは，サービスに使うリソースの使用状況をリソース

監視情報という形で取得し，どのリソースでサービス提供のための VM を起動するかを決めることであり，指標により決定したリソース配置場所の組み合わせを「リソースの組み合わせ」と呼ぶ.

5. 情報提供度によるリソース配置プロセス

本章では定義したモデルのうち，リソース監視情報をしぼり，具体的なリソース配置のプロセスをシミュレーションモデルによって示す.

5.1 シミュレーションモデル

リソース配置のシミュレーションの例として，ある一定期間内にストリーム配信サービスをユーザに提供するとき，複数のクラウド事業者が提供するサーバ群をリソース群とし，その間でどのようにリソースを配置するかを示す．ここではストリーム配信のシステムは図 2 のとおり，ストリーム配信端末（src），受信端末（dst）とサーバ群から構成され，サーバ間の接続ネットワークのトポロジは全サーバが相互に接続されたメッシュ構成を仮定する．ただし，メッシュ構成はシミュレーションで全ての経路から適切な組み合わせを求めるためのもので，実際にリンクのないところやリソース監視情報を取得できない経路は存在しない経路として RTT を大きくすることで，実際のシステムのトポロジを表現するものとする．また，ネットワーク上のサーバ群から提供されるリソース監視情報から，どのサーバにリソースを配置すべきか判断する管理サーバが存在するものとする．図 2 において，$l_{t_i}(s_p, s_q)$は時刻 t_iのサーバ s_p とサーバ s_q間の RTT である.

このとき，ユーザのサービス品質を維持するという利用条件を，各サーバでサービス提供（ここではストリーム配信）に必要なサーバのストリーム処理能力（ここでは CPU 未使用率）で表すこととする．すなわち，CPU の未使用率の高いサーバを使うことを前提として，RTT を最小とする経路を探す．具体的にはシミュレーションでは，一定期間（時間 t_0~t_k）の各ステップ t_i（$0 \leq i \leq k$）ごとに図 2 の各サーバ，配信端末，受信端末間の RTT が入力値として乱数値で与えられ，そのときにストリームの処理能力の余裕があるサーバの集合から，配信端末から受信端末までの RTT が最小の組を抽出する．このとき抽出されたサーバはリソース消費しているので，サーバの残処理能力の値を更新し，その更新値を次のステップ t_{i+1} の入力項目として入れる．この処理をどのリソース群にもストリームの処理の能力がなくなるまで繰り返す．各サーバや端末間の RTT の初期値は乱数で与え，各サーバの残処理能力は未使用率 1（100%処理に使える）とする．シミュレーションの出力項目は期間 t_0~t_k の間にストリームを配信する際に使われたサーバの組である.

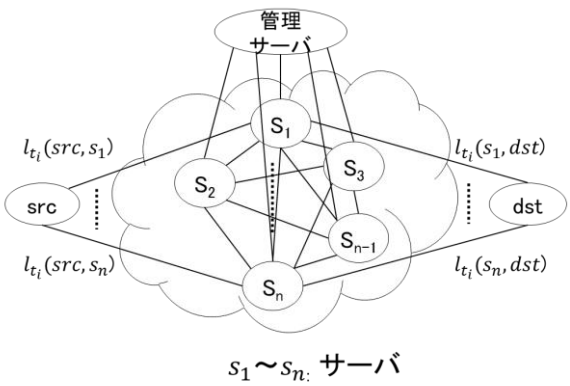

図 2　システム構成

Figure 2　System configuration

このような条件において 3.2 のモデルをシミュレーション用に以下のように定義する.

(1)提供リソース

シミュレーションモデルでは n 個のリソース群（クラウド事業者）を現在（一定期間内に）提供できるサーバとし，S=$\{s_1, s_2, s_3 \ldots s_n\}$と表す．これらのサーバは地理的，ネットワーク的な距離や処理性能の差はあるが，どのサーバでも要求の処理ができる VM を起動できるものとする．このとき，ユーザのサービス品質をできるだけ維持できるものがリソースの候補として挙げと仮定するので，このシミュレーションにおいては，配信予定のストリームを処理するために必要な CPU 性能に余裕があるサーバのみが候補となる.

(2)リソース監視情報

リソース監視情報 I=$\{I_N, I_{SA}, I_{ST}\}$は今回のシミュレーションにおいては， I_Nとして RTT を，I_{SA}として CPU 使用率を用い，I_{ST}を考慮しないこととする．よって，I_{ST} は常に 0 とする.

(3) リソース使用状況

各サーバのリソースの使用状況を表す 3 項組を $r_i = (I_{Ni}, I_{SAi}, 0)$と表し，S に含まれるサーバそれぞれのリソースの使用状況全体は R=$\{r_1, r_2, r_3 \ldots r_n\}$（ $r_i = (I_{Ni}, I_{SAi}, 0)$），$1 \leq i \leq n$)と表すこととする．各リソース使用状況は$0 \leq r_i \leq 1$で表され，$I_{Ni}$はストリームの配信端末から候補となるサーバ間を通り，受信端末までの経路の RTT の総和とし，最小 RTT を 0，最大 RTT を 1 となるよう正規化した値で表されるものとする．また，I_{SAi} はストリーム配信にどれくらいの処理能力に余裕があるか，すなわち CPU 未使用率を用いることとし，未使用率が 0%の場合を 0，未使用率 100%の場合を 1 と表す.

なお，サービス品質を維持する指標として用いるストリーム配信処理に必要な I_{SA}の値をここでは S_Tと定義する.

(4) 目的関数

シミュレーションでは RTT が最小となるサーバ s_i の組み合わせ（集合）S_M を求めることを目的とし，式(2)で表す.

目的関数：

$$min\left(\sum_{r_i \in R} r_i\right) \quad (\mathrm{r_i}=\begin{cases} I_{Ni} \ if \ I_{SAi} \geq S_T \\ 0 \ \ if \ I_{SAi} < S_T \end{cases}), (1 \leq i \leq n) \quad \cdots \ (2)$$

制約条件：

$0 \leq I_x \leq 1$　を満たす I_xのみが使用される

(5) シミュレーションの入力と出力

シミュレーションの入力と出力をまとめると以下のようになる．ここで$l_{t_i}(s_p, s_q)$は時刻 t_iにおける s_pとs_qの間のRTT である．

時刻 t_i の入力：

t_i の各サーバや端末間の RTT の集合 $I_{Ni}(t_i)$

$$I_{N_i}(t_i) = \left\{ l_{t_i}(s_p, s_q \in \{s_1 \sim s_n,\ src,\ dst\}) \right\}$$

t_i のサーバ S_i の CPU 未使用率 $I_{SAi}(t_i)$

$$I_{SA_i}(t_i) = \left\{ I_{SA_i}(t_{i-1}) - St,\ s_i \in \{S_M(t_{i-1})\} \right\}$$

$$I_{SA_i}(t_0) = 1$$

時刻 t_i の出力：

$$S_M(t_i) = \{ s_i (1 \leq i \leq n), s_i \text{は式(2)を満たす} \}$$

シミュレーション全体の入力：

$$\{ I_{N_i}(t_i), \quad t_i = t_0 \sim t_k \}$$

シミュレーション全体の出力：

$$S_M = \{ S_M(t_0), S_M(t_1), \ldots S_M(t_k) \}$$

また，図 3 にシミュレーションの流れを示す．

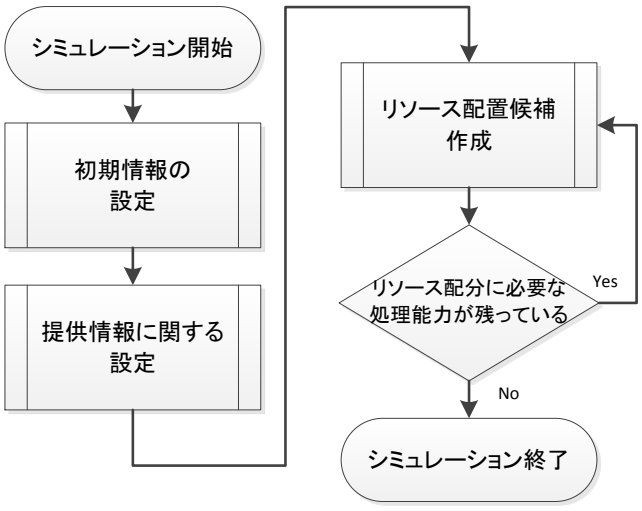

図 3　リソース配置フロー

Figure 3　Resource Allocation Flow

5.2 リソース配置候補の作成

シミュレーション全体のフローを図 3 において，リソース配置抽出候補作成にリソースの優先配置を考慮しない場合のフローを図 4 に示す．

このフローは各サーバのリソース状況を入力し，CPU リソースの余剰が十分でないサーバを間引き，ダイクストラ法により src と dst 間の RTT が最小となるサーバの組み合わせ（リスト）を求める．

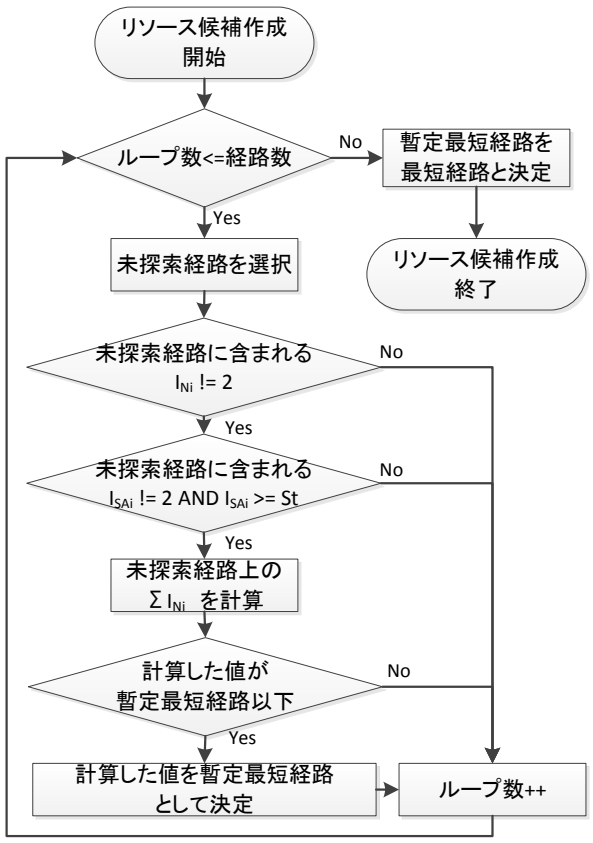

図 4　リソース配置候補作成アルゴリズム

Figure 4　Resource candidate extraction algorithm

このとき，サービス品質維持（ストリーム配信処理に必要な CPU 性能）を満たすサーバ以外はリストから外れる．

5.3 リソース監視情報の提供度による優先的リソース配置

4 章で提案したリソース監視情報提供の有無によってリソース配置の優先度を変える手順を本節で示す．プロバイダのリソース提供条件の 1 つとして考慮されるべき事項「提供できるリソースの範囲で十分にリソースを使用する」ことを考慮して，提供するリソース監視情報数の多い(情報提供度の高い)ところに優先的にリソース配分する．ここで，情報提供度とは(ネットワーク内で提供されるリソース監視情報数/対象とする全リソース監視情報数)で定義されるものとする．配置候補抽出するフローが図 5 で，図 3 の「リソース配置候補作成」に相当する．このフローにはサーバ毎のリソース監視情報提供度を考慮するための処理として，サーバごとに RTT と CPU 使用率の提供の割合を設定し，その割合をサーバの候補の選択に考慮している．具体的な提供度は 2 種類の情報をどちらか一方を提供する場合 50%，どちらも提供する場合を 100%としている．フロー内の処理としては” I_{SA_i} == 2 AND { rand() mod 100 } < 50 ”の部分に当たる．

5.4 シミュレーション実験

5.2 と 5.3 のフローを C 言語でプログラムを作成し，こ

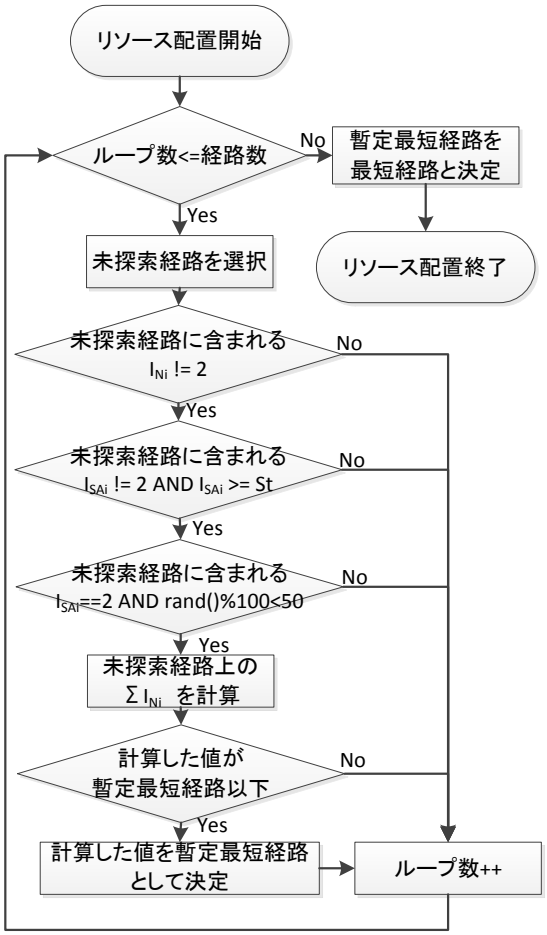

図 5　優先的リソース配置アルゴリズム
Figure 5　Preferential resource allocation algorithm

れを実行する際のパラメータを表 1 のように設定した．これとは別にストリームの配信端末とストリーム受信端末が存在するものとする．このシミュレーションでは配信端末から受信端末間の総 RTT が最小となるようなサーバの集合を選択することとする．

シミュレータ起動時にはリソース監視情報数とサーバ台数を初期値として入力する．この時「リソース監視情報数」とはネットワーク上の取得できるリソースの総和の意味であり，RTT が取得できるサーバ数が 2，CPU 使用率が取得できるサーバ数が 1 の場合は 3 が指定される．この値はリソースの種類×サーバ台数を超えることがないよう設定される．

各サーバは RTT と CPU 使用率の提供を求められている

表 1　シミュレーションパラメータ
Table 1　Simulation parameters.

パラメータ	値
N (サーバ台数)	20
I_{SAi} (各サーバの持つ CPU 性能	1
St (1 ストリーム配信に必要な処理性能)	0.05
I_{Ni} (サーバ間 RTT)	0< I_{Ni} ≤1

ものとし，シミュレータで指定されたリソース監視情報数からランダムに提供できるリソース監視情報が定められる．また，受信端末から各サーバへの RTT や，各サーバから配信端末までの RTT は試行 1 回ごとに乱数値を振り直すこととし，毎回別々の配信端末，受信端末がこのネットワークを利用してストリームを配信するものとする．

5.5 結果と考察

(1) リソース配置の候補抽出

まず，5.2 のシミュレーションによりリソース監視情報数とリソース（サーバ）群全体の関係について考察する．シミュレーション結果として，リソース監視情報数と全サーバの平均 CPU 使用率の関係を図 6 に示す．この結果はリソース監視情報数の総和が 1 から 40 において，それぞれ 20 回の試行をしたときの，全サーバの平均 CPU 使用率である．この時リソース監視情報数は，シミュレーション開始時のパラメータとして与えられる．また，平均 CPU 使用率はシミュレーション終了後の各サーバ CPU 使用率を取得し平均を取ったものである．

図 6 には同時に配信できたストリーム数の傾向も示す．これはシミュレーション終了時に配信されていたストリームの総和であり，リソース監視情報数の総和が少ないほど，送信できたストリーム数が減り，サーバの平均 CPU 使用率が減少している．これは提供されるリソース監視情報数の総和が少ないと，サーバが使用されない，すなわちリソ

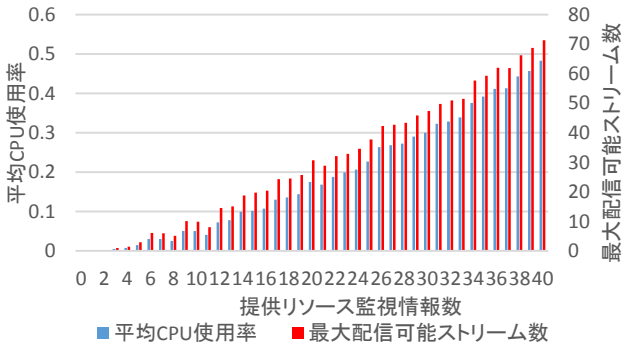

図 6　リソース監視情報数と CPU 使用率の関係
Figure 6 The number of resource monitoring information and CPU utilization rates

ース全体で非効率な状況になることがわかる．言い換えると各サーバが十分に使用されるほど，多くのストリーム処理ができることとなる．

(2) リソース監視情報の提供度優先的リソース配置

次に 5.3（図 5）のシミュレーションにより，優先処理リソース監視情報数の提供度合いがサーバの優先的使用にどのように影響するかについて考察する．図 7 に優先的リソース配置のフローを適用した場合としなかった場合における，各サーバの使用率の違いを示す．ここではリソース監視情報提供度を対象とするリソース監視情報のうちいくつ

の情報を提供しているかを表す値としている．今回のシミュレーションではリソース監視情報は RTT と CPU 使用率なので，その両方を提供している場合は 100%の提供度，どちらか一方を提供している場合は 50%，どちらも提供し

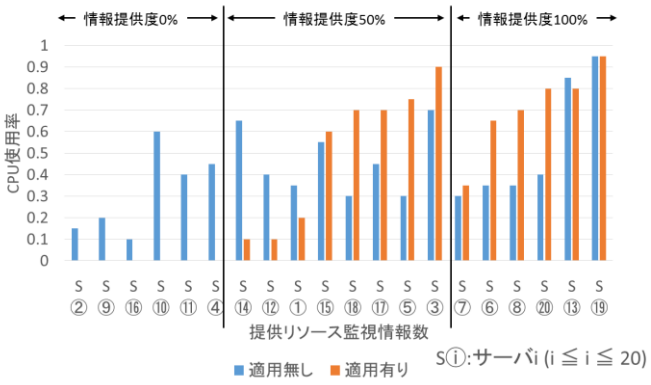

図 7　リソース監視情報提供度と CPU 使用率の関係
Figure 7　The degree to provide resource monitoring information and CPU utilization rates

ていない場合は 0%となる．図 7 には提供度が 100%のサーバが 6 台，50%のサーバが 8 台，0%のサーバが 6 台の際の CPU 使用率に関する偏りを，優先配置無しと，優先配置有りの場合の比較を示している．優先配置が適用されない場合は，リソース監視情報提供度に関わらず一様にサーバが使用されている．これはリソースが使用率に関する従量制の場合，すなわち，リソース監視情報を提供しているリソースを使用可能な範囲で積極的に提供したいという要望に応えにくい配置となってしまう．一方，優先配置ありの場合，サーバ S①から S⑳の合計 20 台のサーバのうち，ばらつきはあるもののリソース監視情報数が多いサーバほど CPU が使用されている割合が多いことが見て取れる．具体的には提案手法の場合はリソース全体のサーバの使用率が 14/20 でリソース監視情報提供度の高いサーバが優先的に利用されているが，優先配置無しの場合は使用率が 20/20 で全てのサーバが一様に利用されていることがわかる．また，リソース監視情報提供度が 100%の CPU 使用率の平均は 70.8%，提供度 50%の CPU 使用率は 50.6%であった．これらの結果から，クラウドリソース監視情報提供者を優遇することができることが示された．

6. おわりに

6.1 まとめ

本稿ではまず，エンドユーザの品質要求を満たしつつ，リソース監視情報が欠損する場合でもできるだけ効率的なリソース配置を行えるモデルを作成した．更に，このモデルから，現実のネットワークを想定し，数あるリソース監視情報の中からネットワークリソースとサーバリソースを取りあげ，それぞれ RTT と CPU 使用率を仮定したシミュレーションモデルとそのシミュレータを作成した．

シミュレーションの結果から，提供されるリソース監視情報が少なければ少ないほど全体で使用される CPU 使用率が少なくなる，非効率な状況に置かれることを示した．また，定義したアルゴリズムを適用することによってリソース監視情報を積極的に提供するクラウドリソースを優遇できることを示した．

6.2 今後の課題

今回は RTT と CPU 使用率の 2 つにリソース監視情報を限定してのシミュレーションを行ったが，GICTF では数多くの細分化されたリソース監視情報が定義されている．これに対応できるよう，様々な情報を加味するアルゴリズムを作成していく．

また，本研究では各サーバが入力，ルーティング，出力の 3 つの機能しか持たなかったが，より詳細な役割を持たせ，それぞれの処理負荷が異なる場合などを想定する．

著者らは多様な要求に柔軟に対応するために，ストリーム加工処理（モジュール）単位を 1 つの VM で行う配信プラットフォームを提案しており[9]，VM の連結，複製などが多々行われる．本稿で提案したリソース配置のアルゴリズムはこのような頻繁に行われる VM のデプロイに対して，限られたリソース監視情報で，リソース配置を決めることができる．また，2.2 で記述したように，リソースの動的配置の頻度が多いクラウドサービスは今後も増えていくと予想される．ストリームのコード変換などをするストリーム配信はクラウドサービスとして提供されているが，ストリームの加工はコード変換のみならず，合成，レート変換，配信プロトコル変換など多様化し，しかもユーザの好みや受信環境に応じたものが要求されるケースが増えていくと想定される．よって，リソース監視情報のみでなく，ユーザの要求も加味されるようなアルゴリズムを提案していく．

謝辞　本研究の一部は日本学術振興会科学研究費助成金 15K00130 の支援を受けて実施しました．

参考文献

1) 後藤厚宏 “クラウド事業者の相互連携でサービス継続性を高めるインタークラウド技術の動向,” 電子情報通信学会誌, Vol.97, No.2, pp.127-132, Feb. 2014.
2) 情報通信技術委員会クラウドコンピューティングアドバイザリグループ， http://www.ttc.or.jp/j/std/ag/cloud/ （2014/08/31 参照).
3) 江丸裕教，高井昌彰, “動的配置法によるハイブリッドクラウドの運用管理コスト最小化,” 情報処理学会論文誌, Vol.54, No.4, pp. 1581-1591, Apr. 2013.
4) 波戸邦夫，上水流由香，岡本隆史，横山大作, “複数の異種クラウド間におけるスケールアウトおよびディザスタリカバリ機構の実装とその評価,” 情報処理学会デジタルプラクティス, Vol.4, No.4, Oct.2013.
5) 高野了成，中田秀基，竹房あつ子，柳田誠也，工藤知宏, “インタークラウドにおける仮想インフラ構築システムの提案,” 情報処理学会研究報告, Vol.2013-OS-124, No.5, pp.1-8, Feb. 2013.
6) 神屋郁子，川津祐基，下川俊彦，吉田紀彦, “複数のクラウドを利用したサーバ広域分散配置システムの構築,” 電子情報通信学会

論文誌 B, Vol. J96-B, No.10, pp. 1164-1175, Oct. 2013.
7) Shriram Rajagopalan, Dan Williams, Hani Jamjoom, Andrew Warfield, “Split/Merge: System Support for Elastic Execution in Virtual Machine,” 10th USENIX Symosium on Networked Systems Design and Implementation, pp.227-240, Apr. 2014.

8) 中島倫明, “OpenStack の概要と CTC の取り組み,” 電子情報通信学会研究報告, ICM2014-40, pp.43-48, Jan. 2015.
9) 田中克哉, 近堂徹, 前田香織, “柔軟なサービス構成を可能にするパーソナライズドストリーム配信フレームワークの設計,” 電子情報通信学会研究報告, IA2014-62, pp.7-12, Nov. 2014.
10) Takeshi Usui, Yoshinori Kitatsuji, Kiyohide Nakauchi, Yozo Shoji, Hidetoshi Yokota, Nozomu Nishinaga, “Design and Implementation of Service Mobility Middleware with Session State Migration,” 電子情報通信学会研究報告, IA2011-91, pp.119-124, Mar. 2012.
11) “インタークラウドのユースケースと機能要件,“ GICTF White Paper, Aug. 2010.
http://www.ttc.or.jp/files/8114/1214/5208/GICTF_Whitepaper_20100902.pdf （2015/08/31 参照）