Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

エッジコンピューティング環境における広域分散アプリケーションの多目的最適資源割当

藤田駿一[†1]　棟朝雅晴[†2]

概要：広域分散アプリケーションの様々なサービス要求要件に応じて効率的なコンピューティングプラットフォームを提供するため，エッジコンピューティングが用いられる。本研究では利用コストの最小化、SLA の最大化、アプリケーションパフォーマンスの最大化を図る計算資源の最適配備について検討し、その数理モデルを提案する。様々な要求要件を持ったユーザに対して柔軟に最適なエッジコンピューティングを提供するプラットフォームの資源割当を多目的最適化として解決する。

キーワード：エッジコンピューティング、最適資源割り当て、広域分散アプリケーション

1. はじめに

近年、様々な要求要件に応じて効率的なコンピューティングプラットフォームを提供するためにクラウドコンピューティングやエッジコンピューティングの需要が高まっている。本研究ではエッジコンピューティングについて利用コストの最小化、SLA の最大化、実行するアプリケーションのパフォーマンスの最大化の３項目を最適化したコンピューティングプラットフォームを提供するための数理モデルを提案し、その最適化し手法について検討する。

クラウドコンピューティングとは、ネットワークを介してユーザがコンピューター資源を利用するコンピューティング手法である。この手法の利点はスマートフォンのような制限のある計算資源を持つ端末でも、十分な計算資源を持つ外部のコンピューターを利用することで、ユーザの端末の性能によらない、複雑かつ高速なアプリケーションの処理が実現可能であることである。

しかし、命令の実行やその結果は、ユーザの端末と外部のコンピューターの間でネットワークを介して行われるため、伝送遅延や伝搬遅延による通信ディレイが生じる。クラウドコンピューティングにおいて想定される外部のコンピューターは集約的に配備されているため、ユーザとの物理的距離が大きくなる。そのため発生する伝搬遅延の増大が、クラウドコンピューティングにおいて問題になっている。この問題を解決するのがエッジコンピューティングである。エッジコンピューティングとは、エンドユーザ付近に分散配備されているエッジサーバを利用するコンピューティング手法の一つである。このコンピューティング手法の利点は、サーバを分散配備することによってクラウドコンピューティングにおける伝搬遅延の問題を回避しながら、外部の計算資源を利用することにある。

しかし、エッジコンピューティングで用いる分散配備されたエッジサーバの計算資源はクラウドと比べて制限されており、かつコストも高いため、クラウドとエッジサーバ、それぞれの利点を生かしたアプリケーションのコンポーネントの分散配備がエッジコンピューティングを効率的に利用するための重要な課題となる。

例えば、計算処理だけを考えればデバイスで実行するよりも外部インフラにコンポーネントをマイグレーションした方がパフォーマンス、処理時間は向上するが、外部インフラを利用するにはタスクをそこにマイグレーションしなければならない。そのため、外部コンピュータとユーザ端末との通信時間やマイグレーション時間を考慮して、モバイル単体のパフォーマンスをこえられなければ有効にエッジコンピューティングを利用したとは言えない。

本研究では本研究では冬道情報サービス用アプリケーション[5]と、プライバシー保護のための隠消現実感アプリケーション[6] をエッジコンピューティング環境を用いた関連研究として参照しながら、クラウドおよびエッジサーバにアプリケーションの各コンポーネントを最適配備するための数理モデル化を通して、エッジコンピューティングの効率的な活用を目指す。

2. エッジコンピューティングの概要

エッジコンピューティングとは分散配備された、ユーザとのネットワーク距離が近いエッジサーバのリソースを用いてサービスを提供、利用することである。田中ら[1]、はエッジコンピューティングの特徴として以下の４点を挙げている。

リアルタイム・アプリケーションの実現

ユーザ・端末の近傍にあるエッジサーバ上でアプリケーションを実行し、リアルタイム性・応答性を改善

ユーザ体験の向上

端末が担ってきた負荷の高い処理をエッジサーバに移すことにより(オフロード)、端末性能に依存しないユーザ体験を実現

地域性のある通信・計算処理の局所化

地域性の高い IoT、M2M の処理をエッジで一次処理する

†1 北海道大学　情報科学研究科
Hokkaido University　Information Science
†2 北海道大学　情報基盤センター
Hokkaido University　Information Infrastructure Center

Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

ことにより、計算処理の分散と通信トラフィックの低減を実現

付加価値サービスへの展開
アプリケーション処理に割り込み、履歴などの取得・分析や様々な高機能化処理を提供

しかし、エッジサーバはクラウドに比べリソース制限やSLAの脆弱さ、コストの面において劣る点もあるため、クラウドの利点も活かすことがエッジコンピューティングの活用において重要な点になる。

3. エッジコンピューティング環境を想定したアプリケーション例

このセクションでは本研究で用いるエッジコンピューティング環境で利用されることを想定したアプリケーション例を示す。

3.1 冬道情報サービスアプリケーション

北海道の冬道では、凹凸や急なホワイトアウトが原因によるスリップ事故が大きな問題となっている。そこで、車の自動運転などを見越した際の車両間での雪道情報などの共有などが求められる。

「冬道情報サービス構築のためのエッジサーバを用いた分散処理に関する研究」[5]で紹介されているアプリケーションは冬道の凹凸を検出し、その情報を多数の車両で共有、活用することを目的としたアプリケーションである。車両に取り付けた加速度センサーを用いて車両情報を検知、共有することで周囲の車両に自分の車両がスリップしたことなどや、道路の凹凸情報を共有することを目的としている。このアプリケーションをエッジコンピューティングにより実装する場合、図1に示す通り、車載コンピュータ、エッジサーバ、クラウド間で情報をやり取りする必要がある。スリップ情報などの共有ではリアルタイム性を求めることから、エッジサーバを活用する必要があり、リアルタイム性を必要としない処理については、クラウドで実行する。

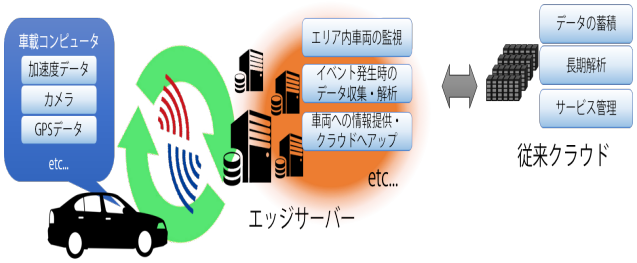

図1　冬道凹凸検出アプリケーションのエッジコンピューティングによる実現例[5]

GPSデータを認識し、その情報から車や路面の状態を検出する。さらにその情報を各車両に送信し、周りの車の状態や路面状況を共有するものである。車のセンサーの処理はエッジサーバで行い、溜まってゆくデータはクラウドに保存されるため、ユーザとエッジサーバ間での通信は盛んに行われ、比べてクラウドとユーザ、エッジサーバ間での通信は少ない。やり取りするデータは、加速度データやGPSデータなどのデータ量の少ないものが主であるので、通信トラフィックは大きく低減される。仮にこのアプリケーションを自動運転などで使う際には、周りの車の位置情報などのデータのレスポンス遅延の最小化は高いレベルが求められ、また安定にこのサービスが動き続けることが求められる。そのため、このアプリケーションの最適配備を検討する際には、高いSLAと高レベルのアプリケーションのパフォーマンス向上が求められる。

3.2 印象現実感アプリケーション

隠消現実感とはカメラやヘッドマウントディスプレイなどのフィルタをユーザが通してみることで、視覚的な物体除去や不可視領域の可視化を実現することである。「隠消現実感アプリケーションを用いたエッジコンピューティングの性能評価に関する研究」[6]ではプライバシー保護のために人を感知し、除去をおこなうというもである。図2にアプリケーション概要を説明する。

まず、カメラなどから画像を取得し、人を感知する。その後人を消した画像を前のフレームの画像、元のフレームの画像を参考に補正したものを出力する。このようなアプリケーションをエッジコンピューティングにより実装する場合、通信するデータがマイグレーションするタイミングによって画像、もしくは特徴量であるため、それぞれのコンポーネント間のデータのやり取りで起こる電送遅延時間に大きな差が生じる。また、このアプリケーションはリアルタイム処理を想定している。そのため、通信環境などを考慮した分散配備によりレスポンス遅延を小さくすることが求められる。

図2にボックスとして示される各処理が本研究におけるアプリケーションのコンポーネントとなる。また、図2の矢印によって示されるデータフローの依存関係を考慮しながら、通信状況や通信時間からクリティカルパスを見つける。このアプリケーションに関してはデータフローは分岐しないため、開始から終了までのレスポンス時間を計算する。また、このアプリケーションでは、コンポーネント間でやり取りするデータが大きいため、コンポーネントの配備を集約的に行うことで効率的にレスポンス遅延を小さくすることが可能である。

Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

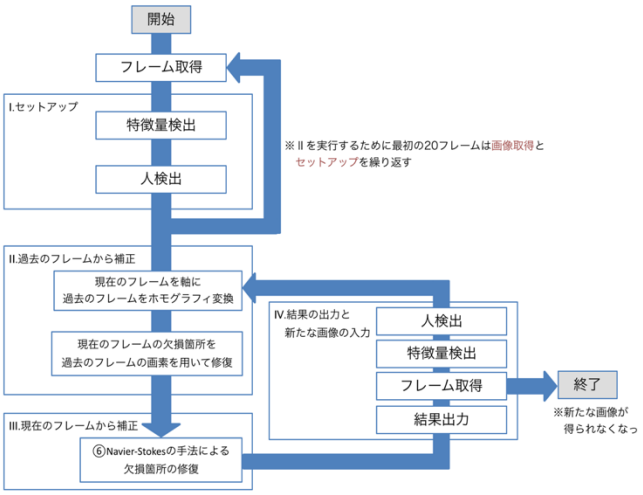

図２　隠消現実アプリケーションの概要[5]

4.　最適資源配備の数理モデル

エッジコンピューティング環境の活用において、対象となるアプリケーションの要求要件と、エッジサーバ、クラウド、ネットワークに関わるシステム基盤の条件を考慮して、最適な仮想マシンを最適配備する必要がある。本研究ではエッジコンピューティング環境の最適化として利用コストの最小化、SLA の最大化、利用するアプリケーションのパフォーマンスの最大化の３項目を挙げたが、これらを目的関数とした多目的最適化問題を解くことで最適化を行う。

4.1　エッジコンピューティング環境の数理モデル化

はじめに、本研究で対象とするエッジコンピューティング環境の数理モデルを示していく。

アプリケーション

エッジコンピューティングに配備される、対象となるアプリケーションを構成する各コンポーネントとそのコンポーネントを実行するために必要となる仮想マシンのインスタンスタイプ（サイズ）を以下のベクトルで記述する。

$$\vec{a} = (a_1, a_2, \cdots . a_n)$$
$$\vec{\imath} = (i_1, i_2, \cdots i_k)$$

ベクトル$\vec{a}$はアプリケーションを構成するコンポーネント番号を表しており、ベクトル$\vec{\imath}$は各コンポーネントを実行するために必要となるインスタンスタイプの番号を示している。また、各コンポーネントを各インスタンスタイプによる仮想マシンで実行した際の処理時間を以下で示す。

$$T = \begin{pmatrix} t_{1,1} t_{1,2} & t_{1,n} \\ t_{2,1} t_{2,2} \ldots & t_{2,n} \\ \vdots \quad \vdots & \vdots \\ t_{k,1} t_{k,2} & t_{k,n} \end{pmatrix}$$

行列 T の要素$t_{i,j}$は i 番目のコンポーネントを j 番目のインスタンスタイプで実行した時の処理時間を示している。

コンポーネントの依存関係

実行するアプリケーションのコンポーネント間には図３に示すようなデータフローによる依存関係が存在する。以下の行列 F によって依存関係を定義する。

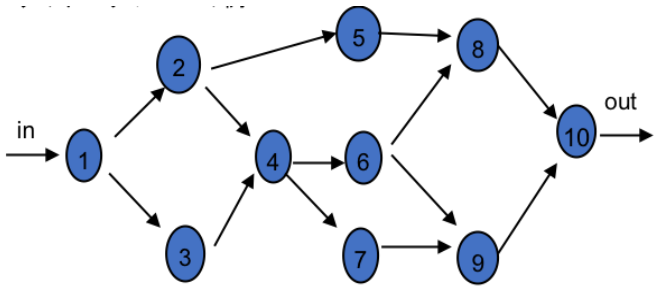

図３　コンポーネントの依存関係の例

$$F = \begin{pmatrix} f_{1,1} f_{1,2} & f_{1,n} \\ f_{2,1} f_{2,2} \ldots & f_{2,n} \\ \vdots \quad \vdots & \vdots \\ f_{n,1} f_{n,2} & f_{n,n} \end{pmatrix}$$

行列 F はアプリケーションのコンポーネントの依存関係を示しており$f_{i,j}$は i 番目のコンポーネントを実行したあと、j 番目のコンポーネントを実行する場合に 1、そうでなければ 0 を示す。図３を例にとると$f_{1,2}$は 1 を示すが、$f_{2,3}$は 0 を示す。

リソースアドレスとリソースキャパシティ

アプリケーションを実行するクラウドやエッジサーバのアドレスの割り当てと、それぞれのリソースが有するキャパシティを以下のベクトルで定義する。

$$\vec{\mathrm{r}} = (r_1, r_2, \cdots . r_m)$$
$$\vec{c} = (c_1, c_2, \cdots c_m)$$

ベクトル r は想定するエッジコンピューティング環境における、アプリケーションを分散配備するためのクラウド、またはエッジサーバのアドレスを要素とするベクトルである。ベクトル c はアドレス付けされているそれぞれのクラウド、エッジサーバのリソースのキャパシティを表している。

リソースコスト

エッジコンピューティング環境をユーザが利用する際、どのリソースにどのインタンスタイプの仮想マシンをデプロイするかによってにかかるコストは変化していく。よって以下に利用するリソースとそこにデプロイする仮想マシンのインスタンスタイプによって発生する利用コストについて以下の行列で定義する。

Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & & w_{1,m} \\ w_{2,1} & w_{2,2} & , \cdots . & w_{2,m} \\ \vdots & \vdots & & \vdots \\ w_{k,1} & w_{k,2} & & w_{k,m} \end{pmatrix}$$

式中の$w_{i,j}$について、j は利用するリソースのアドレスを、i はデプロイするインスタンスタイプを示している。そして$w_{i,j}$はその時の利用コストの大きさを示している。

各リソースの安定性

エッジコンピューティング環境を利用する際、ユーザは様々な外部リソースを利用する。そのため各リソースが安定して運用、サービスの提供が可能かどうかが重要な評価値となる。一般的にエッジサーバは安定性が低く、クラウドは安定性が高いとされている。

$$\vec{p} = \{p_1, p_2, \cdots . p_m\}$$

p_iは i 番目のアドレス番号のリソースについて単位時間当たりにそのリソースが安定して運用することができる確率を示している。

リソースの通信時間

本研究では通信時間に関して、2 つの要素を定義する。その 2 つは伝搬遅延時間と伝送遅延時間である。

伝搬遅延時間

電気信号が伝わる速さはとても速い。例えば空気中を電気信号が伝わる速さは 30 万 km/sec である。よって、エッジコンピューティング環境のリソース間の通信時間においてある程度の物理的距離であれば遅延として考慮する必要のない大きさである。しかし、クラウドなどの集中的に配備されている、ユーザとの物理的距離が大きく離れたリソースと通信する際には、この遅延は無視できないものとなる。この物理的距離によって通信時にかかる遅延を伝搬遅延と定義し、いかに行列として示す。

$$d = \begin{pmatrix} d_{1,1} & d_{1,2} & & d_{1,m} \\ d_{2,1} & d_{2,2} & \ldots & d_{2,m} \\ \vdots & \vdots & & \vdots \\ d_{m,1} & d_{m,2} & & d_{m,m} \end{pmatrix}$$

行列 D の要素$b_{i,j}$はリソースアドレスが i と j のリソース間で通信した場合の伝搬遅延の大きさを示している。

伝送遅延時間

データの送信は、実際には「0」と「1」の数字の羅列を送ることであり、それらは 1 つずつ送信される必要がある。その羅列が通信ポートから送信されてから、最後のビットが送信されるまでの時間を伝送遅延時間とする。この遅延は、送信するデータの大きさと年とワーク通信帯域によって変化する。よって伝送遅延時間を数理モデルで表すために、まずアプリの依存関係があった場合に送る必要のあるデータのサイズを表す行列を示す。

$$B = \{b_1, b_2, \cdots b_m\}$$

行列 B の要素b_iはエッジコンピューティング環境でアプリケーションの i 番目のコンポーネントを処理した後に、i 番目のコンポーネントと依存関係のあるコンポーネントを実行するために参照しなければならないデータサイズを示す。これとユーザの置かれている、エッジコンピューティング環境の通信帯域の大きさを C と置くことで、電装遅延時間は以下のように示される。

$$Transmissiondelay_{i,j} = b_i / C$$

アプリケーションのコンポーネントのリソースアロケーション

アプリケーションを構成するコンポーネントを処理する仮想マシンをどのリソースにデプロイするか、どのインスタンスタイプで処理するかを示すベクトルを以下に示す。

$$\vec{x} = (x_1, x_2, \cdots . x_n)$$
$$\vec{y} = (y_1, y_2, \cdots . y_n)$$

ベクトル$\vec{x}$の要素x_iは、i 番目のコンポーネントをどのリソースに割り当てるかについてリソースアドレスである。y_iは、i 番目のコンポーネントをどのインスタンスタイプで実行するかを示すインスタンスタイプの番号である。このベクトル$\vec{x}$と$\vec{y}$が目的関数の入力となる。

4.2 目的関数

数理モデル化したエッジコンピューティング環境を元に、本研究で最適化するユーザの利用コスト、SLA、アプリケーションパフォーマンスについての目的関数とリソース制限の制約条件について定義する。

ユーザの利用コスト

$$Cost(x, y) = \sum_{i=1}^{n} w_{x_i, y_i}$$

ユーザの利用コストは利用する物理リソースと、そこにデプロイする仮想マシンのインスタンスタイプによって変化する。アプリケーションを構成するコンポーネントを処理するインスタンス毎に利用コストはかかっているため、リソースコストの行列 W を用いることで以上のようにユ

Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

ーザ利用コストを算出する。

SLA

$$SLA(x) = \prod_{i=1}^{n} (p_{x_i})$$

エッジコンピューティング環境がユーザに対して安定してサービスを提供できるかどうかを調べ SLA を評価する。今回評価する SLA は「安定してユーザにサービスを提供できているかどうか」と定義する。よって全てのリソースが安定して運用できているかを評価するため、各リソースの安定性を掛け合わせることで SLA を評価する値を算出する。

アプリケーションパフォーマンス

アプリケーションのパフォーマンス評価であるが、本研究ではアプリケーションを実行する際にユーザ側の入力が結果としてユーザの端末に戻ってくるまでにかかる時間とした。アプリケーションをエッジコンピューティング環境で実行する際に発生する処理時間は大きく分けて 2 種類存在する。

1 つはアプリケーションのコンポーネントを実行、処理する際にかかる時間である。この時間は行列 T によって定義された、各コンポーネント各仮想マシンのインスタンスタイプで実行する際にかかる処理時間によって算出される。

もう 1 つは依存関係のあるコンポーネント同士を、別々のリソースで実行する際に発生する通信時間である。これは行列 D と B によて定義されている、電装遅延時間と伝搬遅延時間によって算出される。

評価はユーザの入力が結果として帰ってくるまでの時間であるため、エッジコンピューテイング環境で実行するアプリケーションの 1 番目のコポーネントを処理し始めてから、最後のコンポーネントを処理しユーザの端末にその結果が返ってくるまでの時間を考える。よって以下の図 4 のようなデータフローのあるアプリケーションの場合、全てのコンポーネントを処理する時間やコンポーネント間での通信時間は考慮せず、赤い矢印で示されているような、実行時間の合計が最もかかるクリティカルパスの時間のみを算出し、パフォーマンスとして評価する。

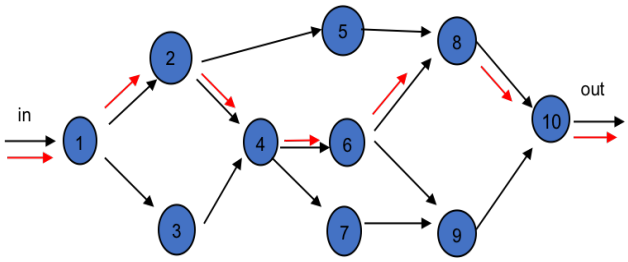

図 4　アプリケーションのクリティカルパスの例

よって、パフォーマンスは実行時間とし、以下の式のように定義する。さらに、クリティカルパス算出のアルゴリズムを、エッジコンピューティング環境を数理モデル化した、各行列とベクトルを用いて示す。

$$Make\ span(x, y) = \sum_{i \in N} \max CriticalPath$$

Algorithm 1 Performance calculation

```
for i = 0 to Number of components do
  L[i] ← 0;
end for

for j = 0 to Number of components do
  for k = 0 to Number of components do
    if F[k][j] = 1 then
      if D[x[k][x[j]] ≠ 0 then
        Box = L[k] + D[x[k]][x[j]] + (B[k]/C) + T[y[j]][j];
        if Box > L[j] then
          L[j] = Box;
        end if
      else
        Box = L[k] + T[y[j]][j];
        if Box > L[j] then
          L[j] = Box;
        end if
      end if
    end if
  end for
end for
Return L[Number of components];
```

5. シミュレーション結果

以上の定義した数理モデルを用いることで、最適化されたエッジコンピューティング環境を評価する。数理モデル化されたエッジコンピューティング環境の各数値は、「A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing」[7]を参考に設定している。

よって、ベクトル$\vec{x}$と$\vec{y}$を入力値とし、$Cost(x, y)$、$SLA(x)$、$Make\ span(x, y)$の 3 つのを多目的問題として最適化する。本研究の最適化手法には「Multi-objective evolutionary

Vol.2019-DPS-178 No.13
Vol.2019-CSEC-84 No.13
2019/3/4

algorithm by Decompose-based」(MOEA/D)[8]を用いる。
以下に[7]の環境を例にエッジコンピューティング環境を数値化した数理モデルを用いて最適化したグラフを示す。

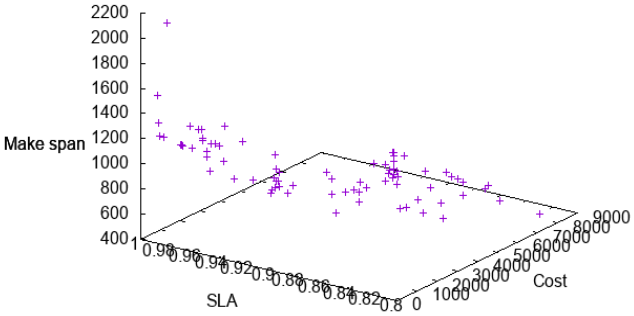

図５　エッジコンピューティング環境での
シミュレーション結果

図５の結果はMOEA/D[8]によって出力されたランク１の個体集団である。Cost の最小化、SLA の最大化、Makes pan の最小化であるため、グラフから最適化されたエッジコンピューティング環境を表現できている。また、以下にエッジコンピューティング環境でのシミュレーション結果とクラウドとモバイルのみを用いたクラウドコンピューティング環境下でのシミュレーション結果の比較グラフを示す。

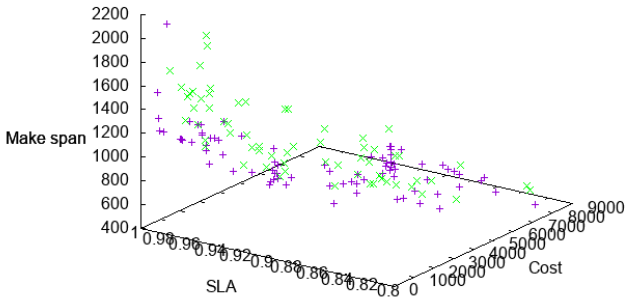

図６　エッジコンピューティング環境と
クラウドコンピューティング環境の比較

紫でプロットした点が図５で示したエッジコンピューティング環境でのシミュレーション結果であり、緑色がクラウドコンピューティング環境でのシミュレーション結果である。

6.　まとめと今後の課題

まとめ

エッジコンピューティング環境でのシミュレーション結果から、エッジコンピューティング環境を最適化することができた。よって、アプリケーションのパフォーマンス向上、利用コスト最小化、SLA の向上のトレードオフ関係のある３項目を最適化したエッジコンピューティング環境をユーザに対して提案することが可能であることを示せた。また、エッジコンピューティング環境と、クラウドコンピューティング環境の比較から、最適化されたエッジコンピューティング環境は従来のクラウドコンピューティング環境と比較して優れていることが証明できた。

今後の課題

今回は、[7]に示されているようなエッジコンピューティング環境とアプリケーションのみを想定して、各シミュレーション数値を設定したが、様々なユーザの状況に応じて柔軟なコンピューティングプラットフォームを提案することが、エッジコンピューティングの目標であるため、様々な数値設定、また実環境下でのシミュレーションが今後の課題となる。

謝辞

本研究は富士通研究所（株）委託研究「広域分散インタークラウドアーキテクチャーにおけるアプリケーション開発・評価および分析」の支援による。

参考文献

1)　“Kostas Katsalis, Thanasis G. Papaioannou, Navid Nikaein, Leandros Tassiulas, “SLA-driven VM Scheduling in Mobile Edge Computing”, 2016 IEEE 9th International Conference on Cloud Computing, 750-757 (2016).
2) Lei Yang, Jiannong Cao, Shaojie Tang, Tao Li, Alvin T. S. Chan, “A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing”, 2012 IEEE Fifth International Conference on Cloud Computing, 794-802 (2012).
3) Krittin Intharawijitr, Katsuyoshi Iida,and Hiroyuki Koga, “Simulation Study of Low Latency Network Architecture using Mobile Edge Computing”, IEICE Transactions on Information and Systems, Vol.E100–D, No.5 (2017).
4) Yuan Zhang, Hao Liu, Lei Jiao, Xiaoming Fu, “To offload or not to offload: an efficient code Partition algorithm for mobile cloud computing”, IEEE 1st International Conference on Cloud Networking (CLOUDNET), 80-86 (2012).
5) 市井 遼平, 棟朝 雅晴, 杉木 章義, “冬道情報サービス構築のためのエッジサーバを用いた分散処理に関する研究”, IEICE-ITS2015-95 , IEICE-115, No.504 (2017).
6) 畑 徹 , 棟朝 雅晴, “隠消現実感アプリケーションに基づくエッジコンピューティングの性能推定”, 情報処理学会研究報告, 2018-MPS-117, No.23 (2018).
7) Lay Yang,Jiannong Cao,Yin Yuan,Tao LI,Andy Han,and Alivin Chan.”A framework for Patitoning and Execution of Data Stream Applications in Mobile Cloud Computing”2013
8) Q. Zhang and H. Li: MOEA/D: A multi-objective evolutionary algorithm based on decomposition, IEEE Trans. on Evolutionary Computation, Vol. 11, No. 6, pp. 712-731, 2007