A-016

クラウドとエッジサーバを用いた広域分散アプリケーションの最適資源割り当てに関する検討
A Study on Optimum Resource Allocation for Widely Distributed Application Using Cloud and Edge Server

藤田 駿一† 棟朝 雅晴‡
Shunichi Fujita Masaharu Munetomo

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ユーザ視点から利用コスト、SLA 違反、レスポンス遅延を最小に抑え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少ない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、複雑な処理を素早く実行できることにある。

しかし、命令の実行やその回答は、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や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になる。

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超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]を例にとり、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。

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

2. 資源最適配備の数理モデル

エッジコンピューティングにおいては、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、ユーザへのサービスの安定性とパフォーマンスの最大化（レスポンス遅延の最小化）、エッジコンピューティングのユーザの利用コストの最小化を目的関数とした多目的最適化問題を、クラウドおよびエッジサーバの資源制約を満たしつつ解決することを目標とした数理モデル化を行う。以下にその概要について説明する。

2.1 構成要素

はじめに、本研究で対象とする数理モデルを構成する各構成要素について説明する。

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、それ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。

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

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。

アプリケーションのデータフロー

実行するアプリケーションのコンポーネント間のデータフローによる依存関係を以下の行列によって定義する。

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

行列 F はアプリケーションのコンポーネントのデータフロー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。

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。

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

ベクトル r に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をユーザが利用する時、どのリソースにどのインスタンスタイプをデプロイするかによってユーザにかかるコストは変化する。ここでは利用するインスタンスによってかかる利用コストについて以下の行列で定義する。

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

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。

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、どのインスタンスタイプでデプロイするかを２つのベクトルで示す。

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

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。

各リソースの安定性

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

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停止してしまう確率を示している。

リソースの通信時間

本研究では通信時間に関係する要素として２つの要素を定義する。その二つは伝搬遅延時間と伝送遅延時間である。伝搬遅延時間とは、電波がある距離を通過するために必要な時間である。例えばこの時間はユーザから集約的に配備されているクラウドに対して通信を行う場合、物理的距離が大きいため大きくなり、分散的に配備されているエッジサーバと通信を行う場合、物理的距離が小さいので小さくなる。電送遅延時間とは送りたいデータを通信経路に送るまでにかかる時間のことを示している。この時間は送りたいデータの大きさと通信経路のスループットに依存する。この２つをベクトルを用いて以下に示す。

$$B = \begin{pmatrix} b_{1,1} \ b_{1,2} & b_{1,m} \\ b_{2,1} \ b_{2,2} \ \ldots & b_{2,m} \\ \vdots \ \ \vdots & \vdots \\ b_{m,1} b_{m,2} & b_{m,m} \end{pmatrix}$$

$$D = \begin{pmatrix} d_{1,1} d_{1,2} & d_{1,n} \\ d_{2,1} d_{2,2} \ldots & d_{2,n} \\ \vdots \ \ \vdots & \vdots \\ d_{n,1} d_{n,2} & d_{n,n} \end{pmatrix}$$

行列 B は伝搬遅延を示している。$b_{i,j}$はリソースアドレスが i と j のサーバ間で通信した場合の伝搬遅延の大きさを示している。行列 D は伝送遅延を示している。$d_{i,j}$はリソースアドレスが i と j のリソース間で通信した時の伝送遅延時間の大きさを示している。

2.2 目的関数

全体構成要素を元に、本研究で検討するユーザの利用コスト、レスポンス遅延、SLA 違反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

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

アプリケーションを構成する各コンポーネントをアロケーションしているリソースアドレスとそこにデプロイするインスタンスタイプから、リソースコストの行列Wを用いてリソースコストを算出し、その合計を求め、ユーザの利用コストとする。

SLA 違反

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

エッジコンピューティング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 違反を評価する。今回はどれか 1 つのコンポーネントが停止した場合にサービスが停止すると考えるため、利用しているリソースが 1 つでも動作しなくなった場合をサービス停止とした。目的関数は各リソースのフォールトトレランスを掛け合わせることで、利用している全てのリソースが正常に動く確率を算出している。

レスポンス遅延（性能）

アプリケーションをエッジサーバで実行するときに発生するレスポンス遅延は大きく分けて 2 種類に分かれる。

1 つはベクトル$\vec{a}$で定義されるアプリケーションを構成する各コンポーネントの実行時間である。これはど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で決まるため、ベクトル$\vec{y}$を入力として行列 T で表される処理時間から算出される。

もう 1 つはコンポーネントを外部の計算リソースにマイグレーションするための通信時間である。これは行列 B と D によって定義される電送遅延と伝搬遅延の合計によって算出される。1 つのコンポーネントをあるリソースで実行された後、データフローに従った次のコンポーネントが別のリソースで実行される場合、リソース間での通信が必要であるため、適宜通信遅延時間を加算していく。

レスポンス遅延は 1 つのアプリケーションに入力データが入り、それが結果としてユーザにレスポンスされるまでの時間として定義する。よって、通信遅延時間とコンポーネントの処理時間を用いて、アプリケーションを実行するクリティカルパスが目的関数となる。

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

リソース制約条件

エッジサーバにはリソース制限があるため、制限なくインスタンスをデプロイできるわけではない。以下にそのリソースの制約条件について示す。

$$\vec{l}=(l_1,l_2,\cdots l_k)$$

$$F_{(x,y)}=\begin{cases}0 & y-x=0\\ 1 & y-x\neq 0\end{cases}$$

$$\left\{C_i \geq \sum_{j=1}^{n}\left(l_{y_j}\times F_{(i,x_j)}\right)\right.$$

ベクトル$\vec{l}$は各インスタンスタイプをクラウドやエッジサーバにデプロイした際にどれほどのリソースを使用するかを示している。これを用いて、各クラウド、エッジサーバにデプロイされているインスタンスがリソース制限を超えていないか調べ、制約条件としている。

これらの評価値を多目的最適化問題として解くことで、様々な要求要件を持つユーザに対して、柔軟なコンピュティングプラットフォームをエッジコンピューティングで実現することを目指す。

3. 対象とするアプリケーション

今回の研究では車載コンピュータの有するデータを共有する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]を例にとり、広域分散アプリケーションのクラウドとエッジコンピューティングを用いた最適配備について、提案する数理モデルにより検討、検証する。

3.1 冬道情報サービスアプリケーション

この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を知らせることを目的としている。本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。リアルタイムで応答する必要性から、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、クラウドで実行できる。

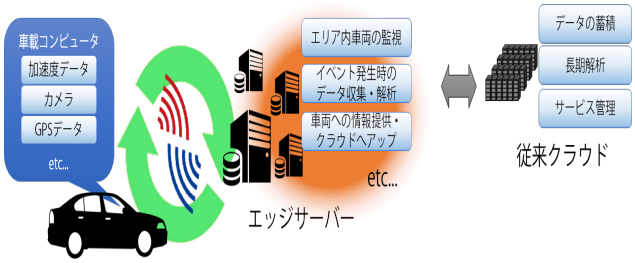

図 1　冬道凹凸検出アプリケーションの
エッジコンピューティングによる実現例[5]

アプリケーションの概要は、車のセンサーから加速度、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、行列Dで示される伝送遅延は小さくなる。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、目的関数の$SLA(x)$と$Response(x, y)$の 2 つが重要視されることになる。

3.2 隠消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」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のだった。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。

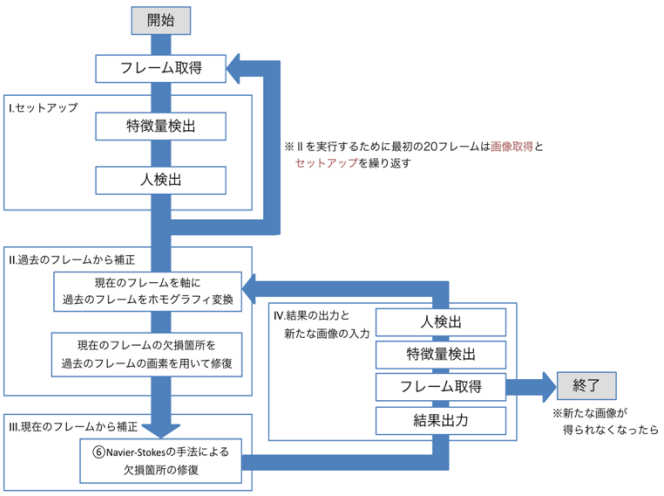

図 2　隠消現実アプリケーションの概要[5]

図 2 にボックスとして示される各処理がアプリケーションのコンポーネントであり、ベクトル$\vec{a} = (a_1, a_2, \cdots . a_n)$によって示される。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行列 F で定義し、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。

4. 今後の展望

対象とする 2 つのアプリケーションそれぞれに目的関数を適用し、特徴の異なるアプリケーションごとにレスポンス遅延、ユーザ利用コスト、SLA の維持において最適な仮想マシンの配備が可能か検討する。最適化については GA を用いて行い、ユーザにとって最適なエッジコンピューティング環境を提案することを目指す。

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).