CM-004

分散フォグノードを活用した移動端末向けオフローディング手法
Offloading Method to Distributed Fog Nodes for Mobile Devices

根市　豊† 戴　瑩‡
Yutaka Neichi　Ying Dai

1. はじめに

クラウドはデータセンタなどで構築され、計算資源を柔軟に提供することができ、実社会においても多くの企業が利用している。機械学習や Internet of Thigs (IoT)の発展により大量のデータを扱うこととなり、それに伴ってデータ通信もまた効率化される必要があると考えられる。しかしながら、データセンタとクラウド利用者の端末の距離が地理的に遠く離れているために通信遅延が発生し、リアルタイム性が求められるサービスにとってはこれが問題となってしまう。

IoT アプリケーションなどのリアルタイム処理を実現する技術としてフォグコンピューティングが注目されている。フォグコンピューティングとは、計算資源が不均質で、地理的に分散され、ネットワークエッジに存在し、クラウドと協調動作をするようなコンピューティングのことである[1,2,10]。例えば、センサネットワークに属する多様なデバイスやモノから生成される膨大なデータをフォグノードがフィルタリングし、有益なデータだけをクラウドへ蓄積・分析させることでスマートなサービス運用を行える。フォグノードとは、従来のネットワーク機能に加えて計算機能を有する機器のことを指しており、基地局や Wi-Fi アクセスポイントといったネットワークエッジがしばしば想定される[1]。

一方 IoT とは別に、クライアント・サーバ型のモバイルアプリケーションについて、サーバプログラムをフォグノード上に移して実行することで通信遅延を削減したという研究[1]がある。これはクラウド上にサーバプログラムを設けておき、その一部のタスクをフォグノードへ手動でオフロードすることによる手法であるが、限られた環境下でしか利用することができない。モバイルアプリケーションのオフロードという観点から、フォグの恩恵を十分に得るためにはスマートフォンやタブレットなどといった移動端末が移動先にかかわらずどこでもフォグを利用できることが望ましい。本稿では利用者の移動を考慮に入れ、地理的に分散されたフォグノードを活用してタスクのオフロードを行う手法を提案する。

オフロードとは計算能力の低い計算機が計算能力の高い計算機へタスクの実行を任せることで計算負荷を軽減することである。オフロードの手続きはコード解析、分割の最適化、分割コード生成、コード転送、実行となっている[6]が、本稿ではあらかじめすべてのノードにコード転送をしておき、選択されたノード上のタスクを実行することによってオフロードを行う。

†‡岩手県立大学大学院ソフトウェア情報学研究科
Graduate School of Software and Information Science, Iwate Prefectural University

2. 関連研究

モバイルアプリケーションに関するオフロードの研究が行われている。

[1]の研究ではクライアント・サーバ型のモバイルアプリケーションについて、クラウド上のサーバと通信する場合とフォグノード上のサーバと通信する場合とで応答時間を比較している。フォグノードへのオフロードはあらかじめ手動で行っており、かつユーザはそのフォグノードの接続範囲内でなければ利用することができず、利用環境が限定されてしまうという問題がある。

[3]の研究では、モバイルデバイスでローカルに実行するタスクをサーバ側の仮想マシンへオフロードする研究を行っている。サーバとしてクラウドとフォグを比較し、応答時間がフォグの方が優れていることを示している。この研究においてもやはり利用環境は限定されている。

[6]の研究では、任意のモバイルアプリケーションについて、クラウドへタスクをオフロードするシステムを提案している。どこからでもクラウドへアクセスできるため利用環境は限定されないが、通信遅延が発生する。

[11]の研究では、移動端末のコンテクストに基づいて最適なクラウドサーバへタスクをオフロードする手法を提案している。これも[6]の研究と同様にクラウドを用いているため、遅延が起こると思われる。

3. 提案手法

利用者の移動を考慮に入れ、地理的に分散されたフォグノードを活用してタスクのオフロードを行う手法として、利用者の位置と利用するタスクの状況により、オフロード対象になるフォグノードを選択するアルゴリズムを提案する。まずはオフロードサービスの利用モデルについて説明する。次にこの利用モデルに基づいたシステムのアーキテクチャについて、各モジュールの機能と動作を説明する。Resource Manager モジュールが計算資源のクラス分けを地域の側面と性能の側面で行うが、そのときに用いられる地域クラスと性能クラスについて述べたうえでフォグノード選択アルゴリズムを説明する。

3.1 サービス利用モデル

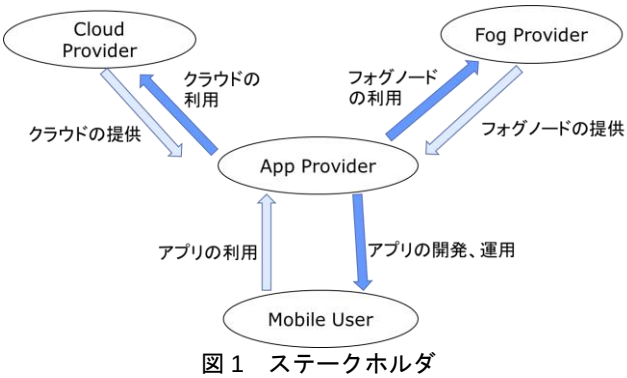

図 1　ステークホルダ

まず、ステークホルダとして Application Provider(AP)、Fog Provider(FP)、Cloud Provider(CP)、Mobile User(MU)の 4 つを定める。

・**AP**：モバイルアプリケーションの開発・運用を行う個人または企業を想定する。

・**FP**：フォグノードの管理・提供を行う営利組織を想定する。

・**CP**：従来のクラウドサービスプロバイダを想定する。

・**MU**：AP が提供するモバイルアプリケーションの利用者を想定する。

AP は、従来のパブリッククラウドの利用と同様に FP と契約を結ぶことによりフォグノードを利用する。提案手法では、AP が利用可能なすべてのフォグノードに静的にサーバタスクを設置しておき、MU からアクセスされたときにその MU に対して最適なフォグノードを選択し、使用させることによってオフロードを行う。通信遅延が許容されるようなタスクについては、CP との契約により利用可能なクラウドに設置しておき使用させるということも考えられる。初期アクセス時点で最適なフォグノードが決定し利用していても MU が移動するとノードとのネットワーク距離が伸びてしまいフォグの長所を活かせない。そこで、現在位置から有効に利用可能なフォグノード群を 1 つのグループとし、これを超えるとその時点で再びそのクラスにおいてフォグノードの選択計算を行うという形で移動を考慮する。

3.2 システムアーキテクチャ

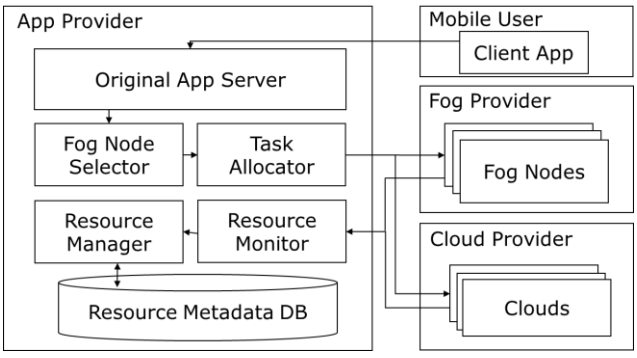

図 2　システムアーキテクチャ

上記のサービス利用モデルに従い、AP がオフロードを行うためのシステムを図 2 に示す。このシステムはオリジナルサーバ側で動作する。システム内の各モジュールの概要は以下の通りである。

・**Original App Server**：アクティブさせるタスクのフォグノード先を変更する必要があるとき、そのリクエストを Fog Node Selector に送る。ここで、アクティブにするとは対象ノード上のサーバプログラムを実行させてクライアントの要求を受け付けている状態にする操作を意味している。

・**Fog Node Selector**：利用可能なフォグノードから適切なフォグノードを選択するアルゴリズムにより、Task Allocator にアクティブさせるフォグノード情報を提供する。

・**Task Allocator**：FP との契約により利用可能となったフォグノードへタスクを割り当てる。または、選択されたフォグノード上にあるタスクをアクティブにする。

・**Resource Monitor**：現在利用可能なすべてのフォグノードの負荷状況を監視し、情報を Resource Manager に送る。

・**Resource Manager**：FP との契約時, または CP との契約時に利用可能となった計算資源に関する性能情報を管理する。

・**Resource Metadata DB**：Resource Manager により収集されたメタデータを蓄積する。

メタデータの要素としてフォグノードの IP アドレス、位置情報、ネットワーク帯域幅、メモリ容量、CPU 性能、ストレージ容量を用いる。これらにより利用可能なフォグノード群を生成する。また地域クラスと性能クラスを導入する。そして、クラスは MU のアクセス時点での位置情報をもとに選定され、同一クラス内のフォグノードをデータベースから検索するのに用いられる。 最適なノードが選択された後に、そのノードの IP アドレスを Task Allocator に知らせることで、アクティブにするオフロード先を変更する。

Resource Monitor により一定時間ごとに送られる各フォグノードの負荷状況を表すデータをここでは状態データと呼ぶこととする。状態データはネットワーク帯域の消費率、メモリ消費率、CPU 稼働率、ストレージ利用率、消費電力量の 5 項目で構成される。

3.3 地域クラス

ユーザの現在位置から地理的に近しいフォグノードを決定するために地域クラスを導入する。地域クラスは、フォグノードやユーザデバイスの地理的位置を平面座標で表し、一定範囲内に含まれるノード群をまとめたクラスである。ユーザの位置座標を取得したのち、その座標が含まれるクラスを算出することで候補ノードを決定する。地理的距離が短いほどネットワーク距離も短いとは限らないが、 1 ホップで到達可能なフォグノードならばネットワーク距離は最短となる。データベース中のフォグノードの検索を効率化するため、地域クラスを静的に生成する方法と動的に生成する方法に分ける。

3.3.1 静的生成法

要求パケット到着時ではなく、あらかじめ静的に地域クラスを生成しておく方法である。この段階では国や県とい

記号	意味
C_{Bi}	帯域幅の性能クラス
C_{Mi}	メモリの性能クラス
C_{Si}	ストレージの性能クラス
C_{Ci}	CPU の性能クラス
$C_{location}$	地域クラス
Location	端末の位置情報
IP Address	端末のアクセスポイントの IP アドレス
b	帯域幅の状態データ
m	メモリの状態データ
s	ストレージの状態データ
c	CPU の状態データ
p	消費電力の状態データ
T_b,T_m,T_s,T_c,T_p	各状態データの閾値
Req[Location,IP Address,C_{Bi},C_{Mi},C_{Si},C_{Ci}]	タスクの要求ベクトル
FN[Location,IP Address,C_{Bi},C_{Mi},C_{Si},C_{Ci}]	フォグノード
Candidate[]	候補ノードベクトル
Status[b,m,s,c,p]	状態ベクトル
T[T_b,T_m,T_s,T_c,T_p]	状態データの閾値ベクトル

表 1　表記一覧

うおおまかな地域でクラスを分割する。

3.3.2 動的生成法

タスクの要求パケットが到着した時点でのユーザの位置情報に基づいて地域クラスを動的に生成する方法である。静的生成法による地域クラスよりも細かいクラスである。

表 1 は各記号を説明する一覧表である。

移動端末の位置座標を$U(x,y)$、フォグノードの位置座標を$F_i(x_i,y_i)$と表すと、移動端末とフォグノードのユークリッド距離は式(1)で表される。

$$dis = \sqrt{(x-x_i)^2+(y-y_i)^2} \qquad (1)$$

ここで、閾値Lを与えて式(2)のような不等式を得る。

$$dis \le L \qquad (2)$$

式(2)の不等式は移動端末から各フォグノードまでの距離が一定の長さ L 以下であるかどうかを判定するのに用いられる。

3.4 性能クラス

オフロードを行うタスクが要求する資源量を要素ごとにレベル分けして表すために性能クラスを導入する。性能クラスC_{Xi}は計算資源 X がレベル i の水準であるということを表している。例えば、帯域幅を低・中・高の間隔で定義していたとすると、中程度の帯域幅資源は$C_{B,Mid}$で表される。

アプリケーションプロバイダが自身でクラスのレベル水準を設定し、クラス分けを行う。

3.5 フォグノード選択アルゴリズム

ユーザの現在位置とタスクの性質から、オフロードするのに最適なフォグノードを選択するためのアルゴリズムを示したのが図 3 である。

リクエスト待ち： ユーザからタスクのオフロード要求を待ち続ける。

静的に生成された地域クラスによるフォグノードの選定：おおまかな地域クラスを特定することで、候補ノードを絞り検索時間を短縮する。

性能クラスによるフォグノードの選定：利用可能なフォグノードから、要求されたタスクを処理するために必要な計算資源それぞれの水準をすべて満たすフォグノードを選定する。

手順 1. Selection by capability class

for all FN i,
  if FN_i [C_{Bi}] >= Req[C_{Bi}] and
  FN_i [C_{Mi}] >= Req[C_{Mi}] and
  FN_i [C_{Si}] >= Req[C_{Si}] and
  FN_i [C_{Ci}] >= Req[C_{Ci}]
    then candidate[] ← FN_i

1 ホップ到達可能か：性能クラスにより選定されたフォグノードについて、1 ホップで到達可能かどうかを判定する。手続き 1 を経た候補ノードベクトル内のすべてのフォグノードに対して、ユーザのアクセスポイントの IP アドレスを持つフォグノードが存在するかをデータベースで検索し、ヒットすればそのノードを候補ノードベクトルへ格納する。

手順 2. Decision one-hop reachability

for all FN j in candidate[],
  if FN_j[IP address] == Req[IP address]
    then candidate[] ← initialized, candidate[] ← FN_j,
  do Decision candidate usability

動的に生成された地域クラスによるフォグノードの選定：地域クラスを用いて、ユーザデバイスの位置座標からどの領域にいるかを明らかにし、その領域中のすべてのフォグノードを選定する。

手順 3. Selection by location class

for all FN k in candidate[]
  if FN_k[location] ∈ $C_{location}$
    then candidate[] ← FN_k
    else eliminate FN_k from candidate[]

ネットワーク距離の測定：地域クラスで選定されたすべてのフォグノードに対してパケットをマルチキャストし、Round Trip Time (RTT)を測定する。

候補ノードを最適順にソート：ネットワーク距離について短い順にソートする。

候補ノードが現在使えるかどうかの判定：状態データの各値が閾値より小さければそのフォグノードは過剰な負荷がかかっていないと判断され、このノードが選択される。

手順 4. Decision candidate usability

for all FN t in candidate[]
  Status[] ← get current state data of FN_t
if Status[b] <= T_b and
Status[m] <=T_m and
Status[s] <=T_s and
Status[c] <=T_c
  then return FN_t
  else t++

選択されたノードの情報は Task Allocator へ送られ、ユーザは以後そのノードと通信を行う。

4. 実験および評価

iFogSim[7]というシミュレーションツールを使用して評価実験を行う。オリジナルサーバはクラウドを用いて構築されているものとし、クラウドと直接通信を行う場合と選択されたフォグノードにタスクをオフロードする場合とで比較する。オリジナルサーバの実行に関わる資源消費を含む通信遅延時間やアプリ実行時間をはじめ、タスクあたりの各資源の消費率、資源の利用率などを測定し、それらにより得られた結果から総合的に評価を行う。

実験で使用するアプリケーションとして、観光情報アプリと顔認証アプリの２つを検討している。

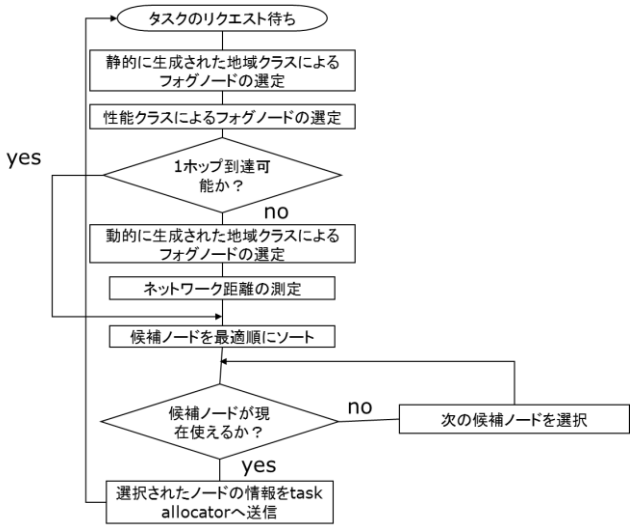

図３　フォグノード選択アルゴリズム

5. おわりに

本稿では端末の移動を考慮したフォグノードへのオフロード方法を提案した。

今後の予定として、シミュレーションを行うつもりである。

参考文献

[1] Blesson Varghese, Nan Wang, Dimitrios S. Nikolopoulos, and Rajkumar Buyya, ”Feasibility of Fog Computing”, arXiv: 1701.05451v1[cs.DC], (2017).
[2] Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr, “Coding for Distributed Fog Computing”, arXiv:1702.06082v1[cs.IT], (2017)
[3] Mohammed A. Hassan, Mengbai Xiao, Qi Wei and Songqing Chen, “Help Your Mobile Applications with Fog Computing”, 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops), (2015)
[4] Marcelo Yannuzzi, Frank van Lingen, Anuj Jain, Oriol Lluch Parellada, Manel Mendoza Flores, David Carrera, Juan Luis Pérez, Diego Montero, Pablo Chacin, Angelo Corsaro, and Albert Olive, “A New Era for Cities with Fog Computing”, IEEE Internet Computing, pp. 54 – 67, (2017)
[5] Anirudh Paranjothi, Mohammad S. Khan, and Mais Nijim, “Survey on Three Components of Mobile Cloud Computing: Offloading, Distribution and Privacy”, Journal of Computer and Communications, (2017)
[6] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti, “CloneCloud: Elastic Execution between Mobile Device and Cloud”, ACM EuroSys’11, (2011)
[7] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments”, arXiv:1606.02007v1 [cs.DC], (2016)
[8] Mahadev Satyanarayanan, ”The Emergence of Edge Computing”, Computer, pp. 30 – 39, (2017)
[9] Katherine Guo, Krishan Sabanani, Arun N. Netravali, “Future of Edge Cloud”, 電子情報通信学会誌 Vol.100 No.1, pp. 13 – 18, (2017)
[10] Zijiang Hao, Ed Novak, Shanhe Yi and Qun Li, “Challenges and Software Architecture for Fog Computing”, IEEE Computer Society, pp. 44 – 53, (2017)
[11] Bowen Zhou et al, “mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud”, IEEE transaction on services computing, vol.10, no.5, pp. 797 – 810, (2017)