フォグコンピューティングのソフトウェア基盤における計算リソースの指定手法

2014SE009 江坂亮　2014SE026 稲垣皓太
指導教員：宮澤元

1　はじめに

コンピュータやサーバ等の情報機器のみならず，世の中の様々なモノに通信機能を持たせてインターネットに接続する Internet of Things（IoT）の技術が急速に広まってきている．IoT では，インターネットに接続されたモノ（IoT デバイス）にセンサを持たせて情報を取得し，この情報をインターネットを介してクラウドコンピューティング（クラウド）上のサービスに送信したり，送信した情報をクラウド上のサービスが分析した結果に応じて，IoT デバイスが持つアクチュエータを制御したりすることが可能となる．IoT デバイスとクラウドを連携させることにより，様々な IoT アプリケーションを提供することができる．

しかし，利用される IoT デバイスが増大すると，IoT アプリケーションを効率的に提供することは難しい．2020 年には約 300 億個の IoT デバイスがインターネットに接続されると予想され [1], これらのデバイスからの情報すべてをクラウドが処理することになれば，ネットワークのトラフィックは異常に膨れ上がってしまう．また，クラウドと IoT デバイス間の距離が遠いほど通信レイテンシも大きくなるので，リアルタイム性が求められるようなサービスをクラウドを用いて提供することは困難である．

これらの問題を解決するために，クラウドを拡張したフォグコンピューティング（フォグ）が提案されている [2]．フォグでは，クラウドのデータセンタと IoT デバイスの間にエッジと呼ばれる計算リソースを IoT デバイスからネットワーク的に近い位置に用意する． IoT デバイスからの情報をエッジで処理することにより，データセンタへの一極集中と通信レイテンシの問題を解決することができる．

データセンタの計算リソースとエッジにある計算リソースの具体的な利用方法について，様々なものが考えられる．例えば，データセンタとエッジの計算リソースを一つのソフトウェア基盤の上で統一的に管理するような提案もなされている [3]．IoT デバイスは，この統一的なソフトウェア基盤に対して従来のクラウドと同様に処理要求を出す．実際の処理は統一的なソフトウェア基盤によって適切な計算リソースを用いて行われる．

フォグコンピューティングにおいて，このような統一的なソフトウェア基盤を実現するためには，計算リソースの多様性が問題となる．性能の異なる様々な計算リソースが存在する上に，エッジの管理状況も異なる．エッジによっては，セキュリティが十分確保されていなかったり，悪意を持ったエッジが配置されてしまう可能性がある．このようなエッジの管理状況を自動的に判断することは難しいので，適切な計算リソースを選択するための基準を与える仕組みが必要となる．

本研究の目的は，フォグのサービスプロバイダが安心してサービスを実行できるようなフォグコンピューティング基盤を提供することである．プロバイダがサービスを提供する際に VM が起動するホストを指定できるようにすることで，プロバイダが信用でき，納得できるエッジを用いて VM を起動し，安心してサービスを実行することができる．

2　研究の背景

ここでは，クラウドと IoT の関連とフォグコンピューティング，クラウドの実行基盤をフォグまで含めて拡張する OpenVolcano [3] について述べる．

2.1　クラウドコンピューティングと IoT

クラウドとは，インターネットを介して様々なサービスを提供するコンピュータの利用形態である．計算基盤をサービスとして提供する IaaS などがある．

IaaS クラウドコンピューティングのソフトウェア基盤の例として，OpenStack が挙げられる．OpenStack は，2010 年に Rackspace Hosting と NASA によって始められた IaaS クラウドコンピューティングプロジェクトで，オープンソースのクラウドオペレーティングシステムである [3]．複数のコンポーネントから構成され，データセンタ全体のコンピュートリソース, ストレージリソース，ネットワークリソースの大規模なプールを制御する．OpenStack クラウドは，Controller ノードと Compute ノードから構成される．Controller ノードは Compute ノードの管理を行い，Compute ノードはプロセッシング，メモリ，ネットワーク，ストレージの各リソースを提供して VM インスタンスを実行する．

現在，人の行動や状態, 自然環境の情報など，これまでネットワークとは無縁であった様々なモノがインターネットに接続され情報を送受信する IoT が進展し，これまで活用できていなかった大量の情報を把握し，活用することができるようになった．クラウドコンピューティングと連携して，IoT デバイスから取得したデータを蓄積・分析することでより利

便性の高いサービスを実現できる．

しかし，IoT デバイスの急速な増大により，データ量の爆発的な増加が予想される．このデータをクラウドがすべて処理するとなれば，ネットワークのトラフィックは膨れ上がり，処理時間も大幅に遅くなってしまう．

2.2 フォグコンピューティング

データセンタへの一極集中とレイテンシの問題を解決するために，利用者や IoT デバイスにネットワーク的に近い位置に配置されたエッジと呼ばれる計算リソースを利用してサービスを実現する利用形態であるフォグコンピューティング（フォグ）が提案されている．データセンタへ送られる大量の情報をエッジで事前に処理し，データ量を減らしてからクラウドに送ることでクラウドの負担を軽減することができる．フォグの利用形態として，エッジの計算能力を別のシステムがコントロールし，そのシステムがデータセンタとやり取りするようなものや，エッジ上で動作するアプリケーションがローカルに情報をまとめ，それらをさらにクラウド上のアプリケーションで処理するようなものが考えられる。

2.3 OpenVolcano

OpenVolcano は，データセンタにある計算リソースとエッジにある計算リソースを統一的に管理する，フォグコンピューティングのためのオープンソースソフトウェアプラットフォームである [3]．これまでのクラウドではデータセンタとユーザデバイスで行っていたコンピューティングとストレージの機能を，OpenVolcano ではエッジを含めて再配置し，これらの間のネットワークをソフトウェアによって直接制御することによって，ネットワークとユーザおよびデータセンタとのスケーラビリティを向上させることができる．

OpenVolcano のアーキテクチャは，主に広域のネットワーク制御を司る要素と，データセンタやエッジ内のデータ処理を司る要素から構成される．これらの要素がネットワーク内の複数のマシンに分散配置されることによって，クラウドサービスを実現するために必要なアプリケーションや機能が最適に割り当てられることが保証される．例えば，サービスを実行する VM が，データセンタでもエッジでも動作し，利用者の移動に合わせて VM を移動させたりすることができる．OpenVolcano における利用者の移動に応じた VM の移動の例を図 1 に示す．まず，データセンタから, 利用者に近いエッジ A に VM が移動する．利用者がエッジ B，エッジ C に移動した場合，VM も同様にエッジ B，エッジ C へと移動する．

OpenVolcano ではデータセンタとエッジを含めた全体の計算リソースを統一的に管理するような提案はなされているが，通信会社やクラウドプロバイダがエッジを提供することを考えており，様々な主体がエッジを提供するような状況は考慮していない．

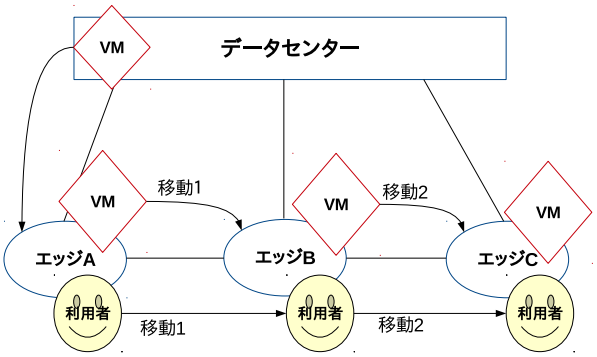

図 1 OpenVolcano における VM の移動

3 フォグサービス実行基盤に対する要求

サービス提供者は利用するサービスの品質について性能面や安定性など様々な要求を持っている．フォグ環境では，サービス提供者はサービス実行基盤の信用性についても要求がある場合がある．利用する実行基盤によっては，性能的なサービス品質を満たしているとしても，実行基盤の提供者が悪意をもっていないとは限らずサービス提供者は不安に感じる可能性があり，信用できる実行基盤でサービスを実行できるようにしたい．この問題を解決するため，サービスの実行基盤を利用者ごとに指定できるようにすることで，指定した条件を満たす実行基盤でサービスを実行する VM を起動できるようにする．提供者が信用できる実行基盤でサービスを実行することで，信用できない実行基盤での実行を防ぎ，安全にサービスを提供することができる．

3.1 実行基盤の指定方法

フォグコンピューティング環境では，サービスを安全に実行できる環境を自動的に判断するのは難しい．実行基盤の中でもエッジは様々な主体が分散して配置するので統一的に管理されていない．そのため，セキュリティが十分に確保されていなかったり，悪意を持つエッジが配置されている可能性がある．提供者が信頼できるノードを指定して VM を起動できれば，この問題が解決できる．

ノードの指定方法として，ホスト名を明示的に決定する方法と，VM を起動する範囲を指定する方法が考えられる．クラウドやフォグの場合，ホスト名を明示的に決定するのは難しい．なぜなら，フォグコンピューティングにおいてデータセンタやエッジはサーバの所在地を意識せずに利用できる点がメリットとして存在し，どこでサービスが提供されている

かわかりづらくなっている．また，問題として，VM を起動するコマンドから情報が漏れた場合，ホスト名が特定され，攻撃の対象となる可能性がある．そのため，機密性を保持するためにも明示的に決定するのは避けた方がよい．

本システムでは，データセンタ・エッジ・全範囲のように，あらかじめ設定した範囲の中から利用者が処理を実行したい実行基盤の範囲を指定し，その中から VM を起動するノードを決定するという方法をとる．起動範囲を判別する情報としてノード位置属性を定義する．データセンタであれば「d」，エッジであれば「e」のように実行基盤ごとに所属を表すノード位置属性を設定することで，VM の起動範囲を指定できるようにする．

3.2 VM を起動するノードの決定

VM の起動範囲と処理能力の有無の 2 つの条件から VM を起動するノードを決定する．

VM の起動範囲 提供者自身がどこまでの範囲なら信用でき，サービスを実行してもよいかを決定する． 本研究では，クラウド，エッジ，全範囲の 3 段階を指定できるようにする．

処理能力の有無 ノードが VM を起動できる程度の処理能力を有しているかを判断する．（OpenStack のフィルタリングによって処理能力の有無を判断できる．）

VM を起動するノードは，提供者が指定した起動範囲の中で決定される．提供者は VM 起動時にどの範囲で VM を起動するかを指定する．例えば，提供者がデータセンタでの VM の起動を指定した場合，Controller ノードが把握している Compute ノードのうち，データセンタに属するノードの中で VM を起動する．図 2 にデータセンタでの VM の起動を指定した場合の起動例を示す．Controller ノードは複数の Compute ノードを統括し，Compute ノードは VM の起動を担う．利用者が Controller ノードに VM の起動命令を送る．起動命令の中にはノード位置属性の指定も入っている．これを受け取った Controller ノードは，ノード位置属性の指定に基づいて Scheduler でスケジューリングを行う．指定された範囲のうち，VM を起動できるノードに VM の起動命令を送り，VM を起動する．

4 実装

3 章で述べたサービス実行基盤の指定手法を OpenStack Newton に実装した．本実装では，ユーザの指定に応じたサービス実行基盤を使用するために，OpenStack の FilterScheduler に変更を加えた．

4.1 スケジューラの変更

OpenStack では，クライアントからの要求に応じてノードを選択し，選択されたノードに VM を起動させる．我々は OpenStack 標準のスケジューラを変更することによって，本サービス実行基盤指定手法を実現した．

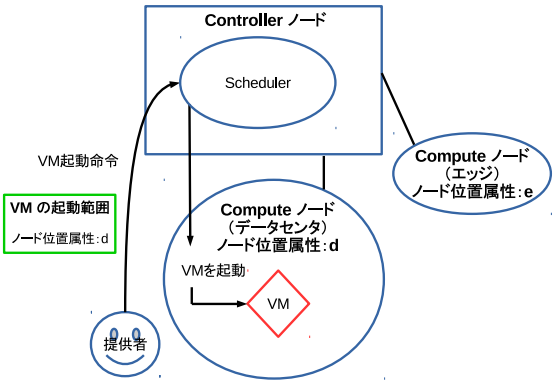

図 2 VM の起動例

4.1.1 標準の OpenStack スケジューラ

OpenStack はデフォルトの設定では FilterScheduler というスケジューラを使用する．FilterScheduler では，Controller ノードが把握している Compute ノードから，VM インスタンスを作成することができる Compute ノードを取り出し，その中からランダムな Compute ノードに VM インスタンスを作成する．

4.1.2 FilterScheduler への変更

実装には FilterScheduler に変更を加え，データセンタに VM を作成するスケジューラとエッジに VM を作成するスケジューラの 2 つを作成する．フィルタリング後のリストからランダムに選ぶのではなく，指定したノードに VM を作成するように変更する．ノード位置属性で検索をかけ，検索にヒットしなかったノードをリストから削除することで，指定したノードが VM の配置先に選ばれるように変更する．

4.2 ユーザインタフェースの変更

ユーザの要求を Controller ノードに伝えるために，実行基盤を指定できるようにユーザインタフェースを拡張する必要がある．例えば，VM を起動するためのコマンドにノード位置属性によって実行基盤を指定できるようなオプションを追加したりすることが考えられる．今回の実装では，単純化のためにユーザインタフェースへの変更は行わなかった．

5 実験

4 章で実装したスケジューラをフォグコンピューティング環境を想定した環境で実行し，指定したノードへ VM インスタンスを作成できるかどうかを確認する実験を行った．

5.1 実験内容

VM を起動する際にノード位置属性の指定をし，VM を作成する．
実験では，ユーザインタフェースから動的にノード位置属性を指定するのではなく，スケジューラのソースコードを変更して静的にノード位置属性を指定して実験している．ホスト名の末尾をノード位置属性として設定し，Compute01d ノードをデータセンタ，Compute02e ノードをエッジとして実験する．

5.2 実験環境

表 1 に Controller ノード，データセンタ側となる Compute01d ノード，エッジ側となる Compute02e ノードの実行環境を示す．

表 1 実験環境

	Controller	Compute01d	Compute02e
CPU	Intel(R) Core(TM)) i7-7700K		
コア数	4		
クロック周波数	4.20GHz		
メモリ	32GB		
ネットワーク	10GBASE-T ギガビットイーサネット		
OS	Ubuntu 16.04 LTS		

5.3 実験結果

ノード位置属性を「d」「e」にそれぞれ設定し，VM を作成した．全ホストの取得後（all_hosts），処理能力でのフィルタリング後（filtered_hosts1），ノード位置属性でのフィルタリング後（filtered_hosts2），最終的な VM の配置先（selected_hosts）のそれぞれのリストを出力し，結果を以下に示す．
○データセンタを指定した場合の実験結果
・all_hosts:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
・filtered_hosts1:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
・filtered_hosts2:
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
・selected_hosts:
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1

○エッジを指定した場合の実験結果
・all_hosts:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 2
・filtered_hosts1:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
(compute01d, compute01d) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 2
・filtered_hosts2:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1
・selected_hosts:
(compute02e, compute02e) ram: 31536MB disk: 1746944MB io_ops: 0 instances: 1

ノード位置属性でフィルタリングをかけた filtered_hosts2 には指定した範囲，データセンタであれば「d」，エッジであれば「e」がノード名の末尾についているノードのみリストに残っていることが分かる．最終的な VM 配置先も指定したノード位置属性のノードが選ばれていることが確認できた．

6 おわりに

本研究では，フォグコンピューティングのソフトウェア基盤における計算リソースを指定する手法を提案した．フォグコンピューティング環境を想定し，VM の配置先を指定できるようにするスケジューラを作成し，それを用いた実験を行った．その結果，指定した実行基盤のノードに VM を配置できることが確認できた．今後は，ユーザインタフェースから動的に起動範囲を指定できるように実装を拡張する．また，エッジの信用性以外の要求に応じて適切なリソース割当てを行う枠組について検討を進める．

参考文献

[1] 総務省: 平成 29 年版 情報通信白書, http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/html/nc133100.html.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli: Fog Computing and Its Role in the Internet of Things, MCC '12, pages 13-16, 2012.

[3] R. Bruschi, P.Lago, G. Lamanna, C. Lombardo, S. Mangialardi: OpenVolcano: An Open-Source Software Platform for Fog Computing, 28th International Teletraffic Congress,2016.

[4] OpenStack: https://www.openstack.org/.