フォグコンピューティングにおける IaaS 向けストレージシステムの試作

2016SE003 浅井由衣　2016SE009 深谷知香　2016SE032 加藤楓
指導教員：宮澤元

1　はじめに

クラウドコンピューティング (クラウド) における負荷の集中や，ユーザとクラウドサーバ間の通信遅延 (レイテンシ) の問題を軽減する技術としてフォグコンピューティング (フォグ) が注目されている．フォグでは，ユーザからネットワーク的に近い場所にフォグサーバと呼ばれる計算ノードを設置し，クラウドサーバで実行する処理の一部をフォグサーバ上で実行することにより，クラウドへの負荷を分散させたり，レイテンシを短縮することができる．

フォグを効果的に利用するためには，ユーザが要求するサービスを適切な計算ノードで実行させることが必要となる．クラウドで IaaS(Infrastructure as a Service) を提供する場合，データセンタ内に設置された均質な計算ノードだけがサービスを実行させる対象となるので，IaaS 基盤ソフトウェアは各計算ノードの負荷状況を考慮するだけでサービスを実行する計算ノードを決定できる．一方，フォグでは様々な計算ノードがネットワーク上に分散配置されるので，ユーザから計算ノードまでのネットワーク距離や，サービスの処理に必要な計算能力を考慮してサービスの処理を実行する計算ノードを決定する必要がある．このようにフォグを利用して IaaS サービスを提供するシステムについては多くの研究がなされている [1][2]．我々はこのようなシステムをフォグ IaaS と呼ぶことにする．

フォグ IaaS では，適切な計算ノードを選択するだけでなく，サービスのバイナリイメージなど，サービスを起動するために必要となるストレージもあわせて提供しなければならない．Chang らは Edge Cloud というフォグ IaaS と同様の概念を提案しており，エッジネットワークに配置したストレージノード上のデータをエッジネットワーク内で共有するようなユースケースについても考察している [1]．しかし，データセンタとエッジネットワーク間でのストレージの扱いについては触れていない．Lebre らはクラウド向けの IaaS 基盤ソフトウェアである OpenStack をフォグに対応するように拡張しているが，ストレージについては今後の拡張が必要であると指摘するにとどめている [2]．また，フォグに対応していない OpenStack をそのままフォグサーバのストレージとして利用した場合，クラウドサーバとフォグサーバ間のネットワーク距離を考慮せずにクラウドサーバのストレージとフォグサーバのストレージを統一的に扱い，レイテンシが増加する可能性がある．

本研究の目的は，ユーザと計算ノード間のレイテンシを考慮してクラウドとフォグを統一的に扱うフォグ IaaS 向けのストレージシステムを実現するためにフォグサーバに読み出し専用キャッシュを設けることの効果がクラウドサーバとフォグサーバ間のレイテンシに応じてどのように変化するかを明らかにすることである．キャッシュの効果を具体的に調べることで，今後 IaaS 向けストレージシステムを実現する際の検討材料とする．

2　研究背景

2.1　IaaS

IaaS とは，仮想化技術を利用して CPU やメモリ，ストレージなどのハードウェアリソースをクラウドサーバからインターネットを介して提供するサービスである．IaaS を用いると，利用者は自身でハードウェアを持たなくてもインターネットを介して，必要なときに必要な分だけハードウェアリソースを利用することができる．

2.1.1　クラウドストレージ

クラウドストレージとは，IaaS を実現させるために必要な，クラウドサーバで動作するストレージシステムのことである．本節ではクラウドストレージの一例として，OpenStack Swift について説明する．OpenStack とは，IaaS 環境を構築するためのクラウド基盤ソフトウェアの一種のことである．Swift はクラウドストレージシステムを提供する OpenStack のコンポーネントで，オブジェクトストレージである [4]．Swift を用いることで自動レプリケーションや分散化といった機能を持つクラウドストレージサービスを構築することができる．

2.2　フォグコンピューティング

フォグコンピューティング (フォグ) は，ユーザ側のネットワークの端にフォグサーバと呼ばれる計算ノードを設置しフォグサーバで情報の処理などを行う技術である．クラウドでは，ユーザはクラウドサーバと情報通信を行う．これに対し，フォグでは，ユーザからネットワーク距離が近いフォグサーバと通信を行うので，クラウドの負荷をフォグに分散させたり，レイテンシを短縮させることができる．

2.2.1　フォグ IaaS

クラウドで提供されていた IaaS をフォグまで範囲を広げて提供するサービスについて説明する．本研究ではこれをフォグ IaaS と呼ぶ．フォグ IaaS はクラウドサーバ，フォグサーバ，ユーザの 3 つの要素で構成される．クラウドサーバがユーザからの要求に対して，複数あるフォグサーバから適切なサーバを見つける．そして，クラウドサーバはそのフォグサーバに仮想マシンを起動させた後，ユーザと通信を行うように命令を送信し，ユーザとフォ

グサーバ間で通信を行わせる．フォグ IaaS については研究がなされている [1]．クラウドで IaaS を提供する場合，データセンタ内に設置された均質な計算ノードのみがサービスを実行させる対象であり，IaaS 基盤ソフトウェアは各計算ノードの負荷状況を考慮してサービスを実行する計算ノードを決定する．それに対し，フォグで IaaS を提供する場合，様々な計算ノードがネットワーク上に分散配置される．そのため，ユーザから計算ノードまでのネットワーク距離やサービスの処理に必要な計算能力を考慮してサービスの処理をする計算ノードを決定する必要がある．また，適切な計算ノードを選択するだけでなく，サービスのバイナリイメージなど，サービスを起動するために必要となるストレージもあわせて提供しなければならない．

2.2.2 フォグ IaaS での仮想マシンの利用

IaaS で提供されるサービスとし仮想マシンがある．フォグ IaaS において，ユーザが仮想マシンを利用する具体例を以下に示す．ユーザが仮想マシンを利用する際に，ユーザはクラウドサーバに対して仮想マシンの起動要求を送信する．クラウドサーバはユーザの位置からネットワーク距離が近いフォグサーバを判断する．そして，クラウドサーバがフォグサーバに仮想マシンを起動させるように要求し，フォグサーバが起動させる．

ユーザはクラウドサーバで仮想マシンを使用するのではなく，フォグサーバで仮想マシンを使用する．その結果，クラウドへの負荷の分散やユーザとサーバの間のネットワーク距離が短くなることにより通信レイテンシを短縮させることが可能になる．

しかし，フォグに対応していないクラウドストレージをそのままフォグサーバのストレージで利用した場合，クラウドストレージは，クラウドサーバとフォグサーバ間のネットワーク距離を考慮せずに，クラウドサーバのストレージとフォグサーバのストレージを統一的に管理する（図 1）．このとき，フォグサーバがデータを必要とした場合，ネットワーク距離を考慮していないクラウドストレージは，フォグサーバのストレージにあるのにもかかわらずレイテンシの大きいクラウドサーバのストレージから取り出してしまう可能性があり，レイテンシの増加を引き起こす恐れがある．

2.2.3 関連研究

クラウド向けの IaaS 基盤ソフトウェアである OpenStack をフォグに対応するように拡張している研究がある [2]．この研究によると．大規模な集中型クラウドコンピューティングからフォグに分散をした小規模への移行が提唱されている．そのために，フォグに対応するように OpenStack を改良し，IaaS プラットフォームを実現している．クラウドストレージの課題を指摘し，ストレージについては今後の拡張が必要であると述べられている．

また，エッジコンピューティングのための分散レプリカ配置アルゴリズムを提案する研究がある [5]．この研究によると，エッジサーバに計算能力を持たせユーザからネットワーク距離の近い場所で処理を行うことでレイテンシは減少するとされている．しかし，エッジサーバにストレージを備えず，エッジサーバがデータを必要とする度にクラウドサーバに要求すると，エッジの利点をなくしてしまう．各エッジサーバにキャッシュを保存すればこの問題を解決できるが，エッジサーバによりそのデータの需要は異なるため，リソースが無駄になってしまう可能性がある．このために，アルゴリズムによりレプリカを配置するエッジサーバを決定して，ネットワークにつながった周囲のエッジサーバと共有することが提案されている．

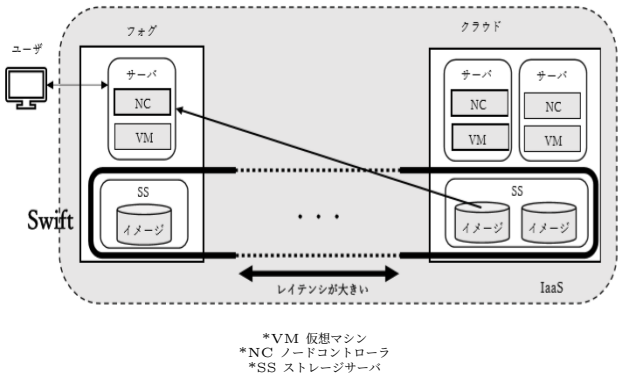

図 1　Swift をフォグ IaaS でそのまま利用する場合の問題点

2.2.4 FUSE

FUSE（Filesystem in Userspace）[6] とは，ユーザがカーネルコードを修正することなく独自のファイルシステムを作成できる機能を提供するものである．従来のファイルシステムは，ファイルシステム毎にカーネル内に処理するコードがあり，必要な時にドライバをロードする，カーネルを再構築するなどの必要があった．しかし，FUSE ではファイルシステムにアクセスするための処理を，ユーザ空間で行わせることができるため，カーネルはそのままで様々なファイルシステムを扱えるようになる．

3　フォグ IaaS 向けストレージシステムの概要

ストレージシステムは，クラウドサーバとフォグサーバで構成されている．ユーザからの要求は FUSE が提供するファイルシステムインタフェースを経由して，フォグサーバに送信される．フォグサーバは必要に応じてクラウドサーバと通信をしてユーザの要求に対する処理を行い，ユーザに返す．また，クラウドサーバとフォグサーバ間のネットワーク距離を考慮し，フォグサーバに読み出し専用キャッシュを作成する．フォグサーバ内にキャッシュを保有している場合は，クラウドサーバとは通信はしない．

4　フォグ IaaS 向けストレージシステムの実装

フォグ IaaS に対応するネットワーク距離を考慮したストレージシステムの検討のために必要な実験を行うファイ

ル転送プログラムを，仮想マシンなどの大容量データを想定し FUSE を用いて試作した．ファイル転送プログラムはフォグサーバ上の FUSE プログラムと，クラウドサーバ上のサーバプログラムの 2 つのプログラムから構成されている．フォグサーバ上の FUSE プログラムが，アプリケーションからのファイル要求をシステムコール経由で受け取り，クラウドサーバ上のサーバプログラムと通信してファイル要求を処理する．

4.1 作成したプログラムの概要

ファイルアクセス時の処理の流れを図 2 に示す．フォグサーバ上のアプリケーションがファイル要求を出すと，FUSE プログラムにカーネルを経由してシステムコールを送る．これを受けて FUSE プログラムはクラウドサーバと通信し，サーバプログラムに必要なデータを送るよう要求する．サーバプログラムは，要求されたクラウドサーバ上のデータを FUSE プログラムに送り，FUSE プログラムはクラウドサーバから送信されたデータを処理する．ここでフォグサーバが保有していないファイルが送信されたとき，キャッシュファイルとして保存し，次回以降はアプリケーションがファイル要求を出すと，キャッシュファイルを処理する．

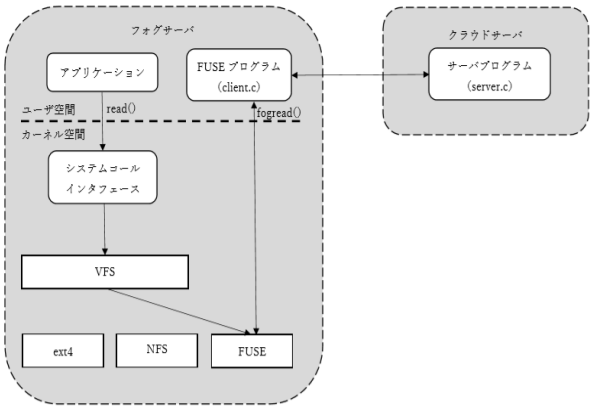

図 2 フォグサーバとクラウドサーバの関係

フォグサーバ・クラウドサーバ間でのファイル転送は，C 言語でソケット通信を用いたクライアントサーバ方式で行った. また，FUSE プログラムがシステムコールに対して処理を行っている間に新たなシステムコールを非同期に受け取ってしまうので，各関数に排他制御を施した.

4.2 ファイルアクセス時の処理の流れ

ソケットを用いて，フォグサーバとクラウドサーバを接続する．FUSE プログラムはシステムコールを受け取るとクラウドサーバと通信し，以下のように各システムコールに合った処理を行う．

- stat システムコールを受け取ったとき
  1. FUSE プログラム内の fogstat() がクラウドサーバにアプリケーションから要求されたファイルの stat 構造体を要求する．
  2. フォグサーバは，クラウドサーバが取得した stat 構造体を受け取る．
  3. FUSE プログラムは，クラウドサーバから得た stat 構造体をアプリケーションに返す．
- readdir システムコールを受け取ったとき
  1. FUSE プログラム内の readdir() がクラウドサーバにアプリケーションから要求されたディレクトリ内の情報を要求する．
  2. フォグサーバは，クラウドサーバが取得したディレクトリ内の情報を受け取る．
  3. FUSE プログラムは，クラウドサーバから得たディレクトリ内の情報をアプリケーションに返す．
- open システムコールを受け取ったとき
  1. FUSE プログラム内の fogopen() が，アプリケーションが要求したファイルに対してアクセス権を持っているかを確認する．
- read システムコールを受け取ったとき
  1. FUSE プログラム内の fogread() がフォグサーバ内にキャッシュファイルを保有しているかを確認する．
  2. 保有している場合は，3 に進む．保有していない場合は，4 に進む．
  3. キャッシュファイルを保有している場合
     (a) FUSE プログラムが，キャッシュファイルの内容を読む．
     (b) FUSE プログラムは，読んだファイルの内容をアプリケーションに返す．
  4. キャッシュファイルを保有していない場合
     (a) フォグサーバは，クラウドサーバにファイルの内容を要求する．
     (b) フォグサーバは，クラウドサーバが取得したファイルの内容を受け取る．
     (c) FUSE プログラムは，クラウドサーバから得たファイルの内容をアプリケーションに返す．

5 実験

クラウドサーバとフォグサーバ間のレイテンシに応じた，フォグサーバに読み出し専用キャッシュを設けることの効果の変化を調べるための実験を行った．実装したファイル転送プログラムを用いて，仮想マシンイメージなどを想定した比較的大容量データを転送する際の，レイテンシの変化やキャッシュの有無によるファイル転送時間を測定する．1 回目は，クラウドサーバからのファイルの読み込みとフォグサーバへのキャッシュファイルの保存を行う．2 回目は，フォグサーバから 1 回目で保存したキャッシュファイルを読み込む．クラウドサーバ・フォグサーバ間のレイテンシは 0ms，10ms，20ms，50ms，100ms，ファイ

ルサイズは 100Mbyte とする．

コンピュータを 2 台用意し，クラウドサーバ, フォグサーバとしデータ送信の実験を行う．実験で使用するコンピュータの仕様について表 1 に示す．

表 1　クラウドサーバ, フォグサーバの仕様

CPU	Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
OS	Ubuntu 16.04.6 LTS
HDD	250GB
メモリ	4.2GB
コア数	6
クロック周波数	4.7GHz

5.1　実験結果

ファイル転送実験をレイテンシ毎に測定したそれぞれの平均値を表 2 に示す．

表 2　実験結果

レイテンシ	0ms	10ms	20ms	50ms	100ms
1 回目	32.123s	1m36.279s	2m40.441s	5m52.919s	11m13.726s
2 回目	0.127s	0.255s	0.361s	0.719s	1.321s

1 回目と 2 回目の差が大きく，1 回目のファイル転送が実用的に利用できる時間では行われていないことが分かった．また，キャッシュがある場合の方がない場合よりもレイテンシの影響を受けていないことが分かった．

5.2　考察

実験結果から実用的なファイル転送が行われていないことが分かった．この原因について 2 つ述べる．1 つ目は，システムコールごとに複数回ソケットへの読み書きを行うことである．1 回の read システムコールに対し最大 131072byte のデータが転送されており，これにより，100Mbyte のファイルを転送するために，データ転送が約 800 回行われていること，1 回のデータ転送につき 8 回のレイテンシが掛かけられていることが分かる．今回実装したプログラムでは，システムコール毎の複数回のソケットへの読み書きによるレイテンシの影響が増加していた．そのため，フォグサーバとクラウドサーバ間のプロトコルを再考する必要がある．2 つ目は，今回想定したデータサイズは仮想マシンイメージの様な大容量のデータであるのに対し，フォグサーバとクラウドサーバ間の送受信回数が，データサイズが大容量になるに従い増加することを考慮できていなかったことである．この問題に対し事前に必要と思われるデータを送信しておくなどの工夫を施すことが必要である．これらの問題を解決することで，より実用的なファイル転送に近づくことが考えられる．また，キャッシュがある場合の方がない場合よりもレイテンシの影響を受けていないことが分かる．そのため，レイテンシの影響を減らすために，フォグサーバにストレージを備え関連研究のように選択した 1 台のフォグサーバのみがキャッシュを保有しネットワーク距離の近い場所にあるフォグサーバと共有を行ったり，レイテンシの大きさによってフォグサーバにキャッシュを保有するか否かを判断する必要があると考える．これらは，フォグサーバの限られているストレージ容量に対して，フォグサーバのストレージ内のデータ量を削減することにも効果があると予想される．

6　おわりに

我々は，フォグ IaaS を実現する際に従来のストレージシステムを使用するとネットワーク距離を考慮せずレイテンシを増加させる可能性に対し，フォグ IaaS 向けのストレージシステムが必要であると考えた．本稿では，フォグ IaaS 向けのストレージシステムを検討するために試作したファイル転送プログラムを用い，レイテンシの変化やキャッシュの有無によるファイル転送時間を実験により測定した．しかし，実験結果により実装が不十分であると分かったため，実用的なファイル転送が可能になるように改良する必要がある．また，キャッシュがある場合の方がない場合よりもレイテンシの影響を受けていないことが分かった．これより，フォグサーバに読み出し専用キャッシュを設けることでフォグ IaaS 向けストレージシステムの実現に近づくと考える．

参考文献

[1] Chang, Hyunseok and Hari, Adiseshu and Mukherjee, Sarit and Lakshman, T. V.: “2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),” http://ieeexplore.ieee.org/document/6849256/, 2014.

[2] Lebre, Adrien and Pastor, Jonathan and Simonet, Anthony and Desprez, Frederic.: “2017 IEEE International Conference on Cloud Engineering (IC2E),” *IEEE* http://ieeexplore.ieee.org/document/7923796/, 2017.

[3] Redowan Mahmud, Ramamohanarao Kotagiri and Rajkumar Buyya.: “Fog Computing: A Taxonomy, Survey and Future Directions,” *Internet of Everything*, 2017.

[4] Joe Arnold, SwiftStack team（菊池研自・船橋弘兼 監訳 （木下哲也 訳）.: “OpenStack Swift–Swift オブジェクトストレージの管理と開発,” *O’REILLY* オライリー・ジャパン, 2015.

[5] Atakan Aral and Tolga Ovatman.: “A Decentralized Replica Placement Algorithm for Edge Computing,” *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, VOL.15, NO.2 ,2018.6.

[6] SourceForge.net.：“FUSE: Filesystem in Userspace,” http://fuse.sourceforge.net, 2020.1.13.