フォグコンピューティングにおけるストレージシステムの試作

2014SE008 榎本国雄　2014SE046 川崎滉司　2014SE097 高橋佑輔
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) が普及し，様々なアプリケーションがクラウドを利用するようになっている．例えば Internet of Things(IoT) は世の中に存在するモノ (IoT デバイス) に通信機能を持たせてインターネットに接続させ，IoT デバイスが取得したデータをクラウドに分析させ, その結果を人や IoT デバイスにフィードバックするという技術である．

ところが現在では、ユーザの大幅な増加やアプリケーションの性質の多様化によりクラウドへの負荷が増加しており，従来のクラウドの処理形態ではこのような多様なアプリケーションの要求に十分に応えられない．レスポンスの遅れがある程度許容される処理，例えば Web 検索，動画の取得再生程度ならばクラウドで処理することも可能だが，車の自動運転など，よりリアルタイム性の高いアプリケーションの要求はクラウドでは処理し切れない場合がある．また IoT のように多数のクライアントが接続される場合，ネットワークのトラフィックがクラウドに集中することも問題となる．このようなクラウドの問題を解決するためにフォグコンピューティングが提案されている．フォグコンピューティングではエッジとよばれる分散処理環境をデバイスからネットワーク的に近い場所に設置する．デバイスからの要求をエッジで処理することによってレイテンシの問題を解決することができる．また，クラウドのデータセンタに送るデータをエッジで前処理することによって，送信データ量を削減することができる．

フォグコンピューティングを用いて効率的にサービスを提供するには，データセンタとエッジの計算リソースを適切に使い分ける必要がある．しかし，サービス提供者がサービスの種類や利用者の多様性を考慮して計算リソースの適切な使い分けを行うのは困難である．フォグコンピューティングにおいても，クラウドと同様に，サービス提供者がデータセンタの計算リソースを仮想化技術を用いて抽象化された形で利用できるような仕組みが必要だと考えられるが，データセンタとエッジの計算リソースを統一的に扱うような提案はほとんどされていない．本研究の目的は，フォグコンピューティング環境においてデータセンタとエッジのストレージリソースを適切に使い分けるようなストレージシステムを実現することである．ストレージシステムのクライアントからネットワーク的に近いストレージリソースにを用いて必要なデータを格納することによって，クライアントは低レイテンシでストレージシステムにアクセスできる．具体的には，クライアントからのネットワーク距離やユーザによる指定などをヒントとして用いて，格納されるデータの複製を適切なストレージリソースに配置する．本ストレージシステムのプロトタイプを Linux 上で動作するエミュレータとして実装した．擬似的に実現したフォグコンピューティング環境において，エミュレータを用いて実現したフォグコンピューティングにおいて，エミュレータを用いてファイルにアクセスする実験を行い、システムが想定通りに動作することを確認する．

2　クラウドストレージ

クラウド上でストレージを提供するサービスをクラウドストレージと呼ぶ. クラウド上ではさまざまなサービスが動作している. これらのサービスはファイルやブロックなどさまざまな形でストレージを利用する. クラウドストレージはこれらのサービスに対してストレージ機能を提供する．

2.1　クラウドストレージの概要

クラウドストレージとはクラウド内のサービスがクラウド内で利用するストレージを提供するサービスであり，主にオブジェクトストレージとしてはたらくものが多い．オブジェクトストレージとはデータを「オブジェクト」という単位で扱う記憶装置でありディレクトリ構造で管理するファイルストレージとは異なりデータサイズやデータ数の保存制限がないので, 大容量のデータを保存するのに適している. データを分割保存したり，データの複製を配置することで信頼性を向上することもできる．

2.2　Ceph

Ceph とはクラウドストレージの一種でありオブジェクトの配置場所をメタデータとして持たないという特徴がある [1]．これによりオブジェクトへのアクセスを行う時にはメタデータサーバへのアクセスが必要なく動作が軽くなる．

Ceph におけるデータは RADOS(Reliable Autonomic Distributed Object Store) とよばれる分散オブジェクトストレージに保存される.RADOS はおもにモニタと OSD(object strage device) で構成されている. モニタは OSD の構成, クラスタ管理、状態管理を行い,OSD はハードディスクや RADOS と 1 対 1 で対応し、オブジェクトの配置の管理, 実ストレージへのデータの読み書き, データの冗長化などを行う. クライアントはモニタからクラスタの情報を取得しその情報をもとに CRUSH アルゴリズムという計算方式を用いてデータの配置先を計算し該当する OSD にアクセスする [2].

Ceph ではストレージを階層的に管理する．どの OSD にデータを保存するかを決める際にデータのノードのリストを必要とする．そのノードのリストをクラスタマッ

プと呼ぶ．クラスタマップとはクラウドストレージの構造を木構造で記述したものである．ノード部分をバケット (Bucket)，その出口部分をアイテム (item)，葉の部分をデバイス (device) と呼ぶ．アイテムにはそれぞれ別のバケットまたはデバイスが含まれており階層構造となっている．バケットは 4 種類存在し，それぞれでアイテムの選び方が異なるが Ceph では主にストローバケットが用いられる．ストローバケットではバケット内のアイテムを選択する際，CRUSH アルゴリズムを用いる．アイテムの重みやアイテムの ID を引数にしたハッシュ関数を使って各アイテムについてハッシュ値を求め，バケット内のアイテムの内，ハッシュ値が最大のものを選択する

クラスタマップの例を図 1 に示す．クラウドストレージの構成要素をラック，ホスト，OSD の三つのタイプに分類することで構造的に表すことができる．OSD は CRUSH

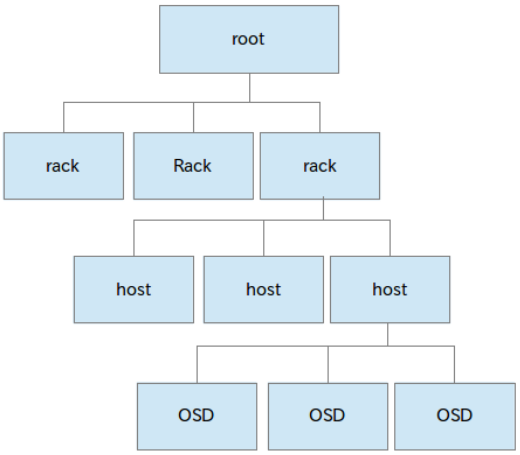

図 1　クラスタマップの例

アルゴリズムを利用して選択される．クラスタマップとストローバケットを用いて選択された OSD の組を複数用意してその中から保存するデータの名前を引数に一つ選択する．そうすることで同じデータ名に対しては同じ OSD のリストを返すので OSD の選択をローカルで行うことができ，クラウドとの通信量を減らすことができる．

2.2.1　クラウドストレージの問題点

フォグ環境においてはクラウドストレージは効率的に動作することができない. フォグ環境ではオブジェクトの複製を配置する際にどのホストに配置するかが重要となるからである.CRUSH ではホストへのネットワーク遅延を考慮しないのでストレージ利用者から離れたエッジに複製を作成した場合、アクセス時のレイテンシが大きくなる恐れがある．

3　本ストレージシステムの概要

本ストレージシステムでは，フォグコンピューティング環境において複製を作成するエッジを選択する際に，ネットワーク遅延を考慮に入れた選択アルゴリズムを用いる．これによって，クライアント毎に適切なエッジを利用できるように制御する．具体的には Ceph における CRUSH アルゴリズムを拡張し，エッジの選択アルゴリズムにクライアントからのネットワーク遅延を反映させる．ストローバケットからアイテムを選択する際に，拡張したアルゴリズムでは，アイテムごとに定めた重みを動的に変更し，クライアントにとって適切と考えられるエッジにより大きな重み情報を与える．大きな重みをもつアイテムのハッシュ値がより大きくなり選ばれやすくなる CRUSH アルゴリズムにおけるハッシュ値は値の大小関係を使ってエッジを選択するもので，配列インデックスには使われない．したがって，拡張アルゴリズムのようにアイテムの重みを変化させても，システムの動作が意味的に変わることはない．クライアントにとって適切なエッジを表すアイテムの重みを大きくするために，クライアントから各エッジへのアクセス頻度を利用する．クライアントにとって適切なエッジのアクセス頻度は高いと考え，そのようなエッジの重みを大きく設定する．これにより図 2 のようにアクセス数が多いエッジファイルの複製場所として選ばれやすくなる．エッジへのアクセス頻度が変化した場合は，エッジの重みも変化するので格納している複製を再配置する必要がある．

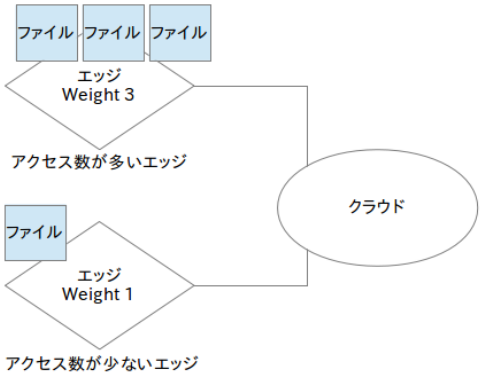

図 2　エッジにファイルが保存された例

4　システムの実装

3 章で述べた設計に基づき，ストローバケットの OSD 選択のシミュレータと，複製作成のクライアント・サーバ・MDS のエミュレータを実装した．

4.1　複製作成のエミュレータ

提案するストレージシステムはクライアント・サーバ・MDS で構成される．サーバは，クライアントからのデータを保存した後，そのデータの複製を作成する．その複製作成の際，その作成先のサーバをストローバケットを用いて決定し，そこへ複製の作成及び MDS への書き込みを

行う．

本研究では実装の簡略化のために，エミュレータには OSD の選択アルゴリズムは実装せず，あらかじめ決められたサーバへ複製を作成するものとした．OSD 選択アルゴリズムの動作確認のためには別途シミュレーションを行った (5.1 節)．

4.2　エミュレータの概要

エミュレータはクライアント・サーバ・MDS の 3 つのプログラムから構成されている．

クライアント

サーバに対し，データの保存・読み出しの命令を行う．

サーバ

クライアントからの要求に応じてデータの保存・読み出しや，MDS へメタデータの書き込み，他のサーバへ複製の作成，及び複製の保存を行う．

MDS

サーバに保存されたデータのメタデータを保存する．

図 3 ではクライアントからデータが送られてきた際の流れを示している．サーバはデータを固定サイズで分割して保存し，MDS へメタデータの書き込みを行う．保存と書き込みが終わったらクライアントと MDS との通信を切り，あらかじめ決められたサーバと接続し，複製の作成を行う．

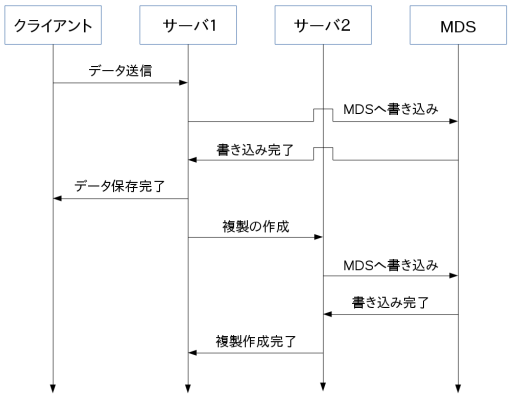

図 3　エミュレータの処理の流れ

5　実験

ファイルを保存するエッジの選択が，エッジへのアクセス数によって変化する様子を確認するためにシミュレーションを行った．また，tc コマンドを使ってネットワークレイテンシを調整した擬似的なフォグコンピューティング環境を作成してエッジを利用したファイル読み書きを行う際にネットワーク遅延によって性能にどれほどの差がでるのかを確かめた．

5.1　ストローバケットのシミュレータ

ストローバケットは各 OSD に重みを設定することにより，複製作成の際に特定の OSD を選ばれやすくすることができる．Ceph のストローバケットでは 3 引数の Jenkins ハッシュ関数を用いてハッシュ値を求め，そのハッシュ値と重みで OSD を選択するが本研究では実装の簡略化のために，rand 関数で生成した擬似乱数をハッシュ値とし，そこへ各 OSD に重みを設定することによってストローバケットのシミュレータの実装とする．このシミュレータにより，ストローバケットを用いることで複製の作成先として特定の ODS が選択されやすくなることを確認する．

図 4 にシミュレータの概念図を示す．OSD に設定された重みが大きいほど複製の作成先として選択されやすいようにしている．

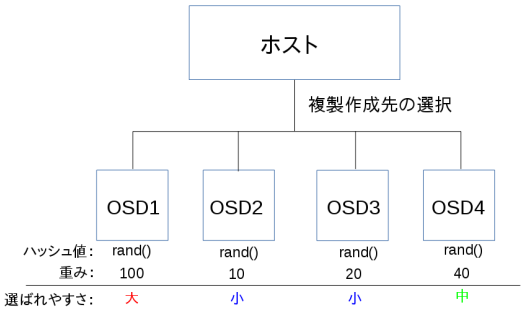

図 4　ストローバケットシミュレータの概念図

5.2　ストローバケットの実験

30 個の OSD に見立てた配列を用意しオブジェクトの選択を 1000 回繰り返した．その時のどのエッジが選ばれるかの平均値を取り、実際に重みを設定した OSD が選ばれやすくなっていることを確認した．今回 OSD10 のアクセス数が多いと見なし重みを増やしてどれだけ選択されるかを調べた．まず最初にすべての OSD に重みを設定せずシミュレートし、そこから OSD10 の重みを増やしていくことでどれだけ選択されやすくなっていくかをシミュレートした．その結果を図 5 で示す．この結果から、OSD に重みを設定することで、特定の OSD が複製作成の選択先として選ばれやすくなっていることが確認できた．

5.3　エミュレータの実験

データの分割書き込みとその読み出しのエミュレータを動作させそれぞれに遅延が生じた場合にどれだけ性能が下がるのかを確かめる．

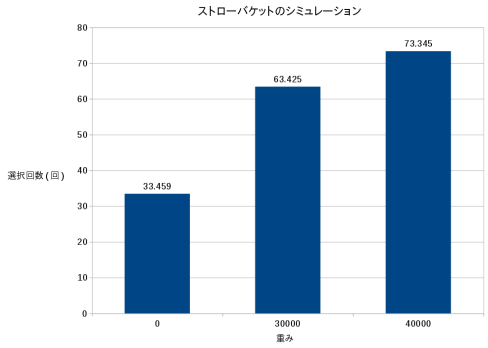

図 5　目的のバケットアイテムの選択回数

5.3.1　データの書き込み

tc コマンドを用いてクライアント-サーバ間の通信に擬似的なネットワーク遅延を発生させた状況でクライアントがサーバからデータを書き込む実験を行った．書き出すデータのサイズは 524MB で，これを 64MB ずつ 9 つに分割して保存する処理を計測した．ネットワーク遅延は 0ms と 50ms の 2 通りとし，それぞれ 3 回実験を行った，平均実行時間を比較した結果を図 6 に示す．50ms 遅延させた方がおよそ 8 秒多く時間がかかっていることがわかる．

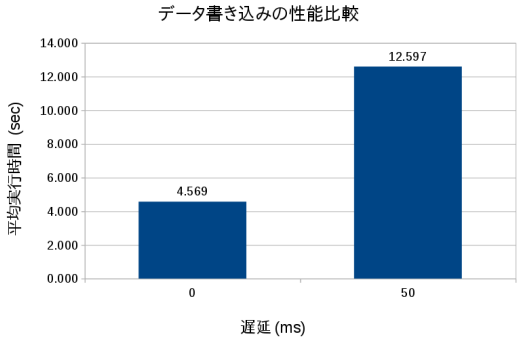

図 6　データの書き込みにかかった時間

5.3.2　データの読み出し

tc コマンドを用いてクライアント-サーバ間の通信に擬似的なネットワーク遅延を発生させた状況でクライアントがサーバからデータを読み出す実験を行った．読み出しデータのサイズは 524MB で，これを 64MB ずつ 9 つに分割して保存されているものをクライアントがサーバからまとめて読み出すのに要する時間を計測する．ネットワーク遅延は 0ms と 50ms の 2 通りとし、それぞれで 3 回実験を行った．平均実行時間を比較した結果を図 7 に示す．ネットワーク遅延が発生したとき，データの読み出し性能が大きく下がっていることがわかる．

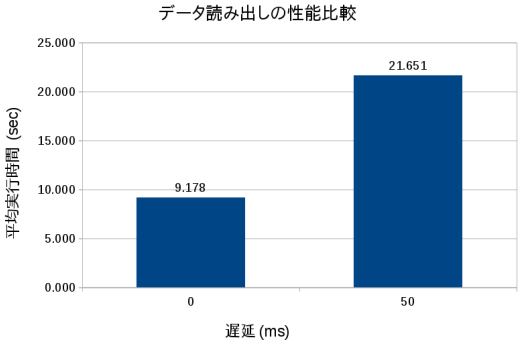

図 7　データの読み込みにかかった時間

6　おわりに

フォグコンピューティング環境において効率的に動作するストレージシステムを提案した．本ストレージシステムでは，クライアントのエッジに対するアクセス頻度を元に設定した重み情報を OSD の選択に利用する．これによって，クライアントごとに適切な OSD にファイルの複製が配置されやすくなり通信レイテンシを低減できる．提案システムのファイル転送のエミュレータを作成し，ネットワーク遅延があることによってファイルアクセス頻度が下がることを確認した．また，シミュレーションによって提案システムで OSD が適切に選択されることも確かめた．今後は，細部についてさらに見当し，本ストレージシステムを実装する．

参考文献

[1] Sage A.Weil,ScottA. Brandt,Ethan LMiller Darrell D.E. Long and Carios Maizahn,“Ceph A Scalable,High-performance Distributed File System,“in Proceedings of the 7th Symposium on Operating Systems Design and implementation pp307-320.2006.

[2] Sage A.well,Scott A.Brandt,Ethan L.Miller,and Carlos Maltzahn,” CRUSH: controlled, scalable, decentralized placement of replicated data”,Proceedings of SC’06 proseeding of the 2006 ACM/IEEE conference on Supercomputing Article,No.122,2006