クラウド環境における OpenFlow を用いたリソース転送の制御

西出 彩花　(指導教員：小口 正人)

1　はじめに

近年，クラウドコンピューティングモデルの出現に伴い，パブリッククラウドやプライベートクラウドが普及しつつある．さらに，それらのクラウドをシームレスに結合する形態のハイブリッドクラウドの検討も行われてきている．パブリッククラウドでは，スケールアウト/スケールダウンできることにより無駄なくリソースを使用でき，コスト削減が期待できる．またデータ管理を専門の業者に預けることにより，技術面のリスク削減にも繋がる．しかし，社外のサービスを利用することによるセキュリティへの不安の声も多くあり，これはクラウド導入が積極的に行われていない重要な要因でもある．他方でプライベートクラウドは，社内のシステムとして構築されており，パブリッククラウドに比べるとセキュリティの不安は小さい．

この 2 つのクラウドを併用するハイブリッドクラウドでは，それぞれの持つ拡張性や安全性などのメリットを両立することができる．例えば，自社でデータセンタを保有し，通常はデータ管理に自社システムのプライベートクラウドを使用し，時期により短期間に大容量データ処理が必要になった場合やアクセスが急増したときの対応手段として，拡張性の高いパブリッククラウドを利用することが挙げられる．また，通常パブリッククラウドを使用している場合にも，セキュリティ面を考慮し，個人情報や社外秘の情報はプライベートクラウドで管理するといった利用方法も検討されて，現在，ハイブリッドクラウドの有用性は強調されている．

しかし，実社会においては，ハイブリッドクラウドの導入はあまり進んでいない．これは，大規模で複雑化しているクラウドの構成が手動での制御の限界に近づいているからである．外部から緊急の情報が入った場合，これに応じて通常時の構成から緊急時の構成へとクラウドを切り替えることで対応する必要がある．しかし，人間の手作業には，規模にも速度にも限界がある．そのため，これを自動化して制御する仕組みの実現が望まれる．

そこで，本研究では，クラウド環境において，外部から入ってくるシグナルをトリガーとし，ホストやネットワークの構成を切り替える仕組みを検討する．具体的には，オープンソースクラウド構築ソフトウェアの OpenStack[1] を用いて，クラウド環境においてリソースの転送を制御する．その際，クラウド内外のネットワークは OpenFlow コントローラの R y u [2] を用いて制御する．緊急の情報を受けた際に，クラウド環境内でインスタンスを急遽マイグレートすることで，緊急時の構成に切り替えるといった想定ができる．また，ネットワークについても同じように，同時に切り替えてリソースの転送に適した環境になるように制御を行う．

このようなシステムを，OpenStack クラウド環境において，OpenFlow の機能を利用して実現することを目標とする．

2　クラウドにおけるリソース配置の最適化

2.1　負荷分散のためのマイグレーション

ある地域でネットワークの利用が集中したり，データ量の多い通信が行われたりすると，特定のホストに負荷がかかるため，効率の良い通信ができなくなる．

そこで，本研究では大きな負荷がかかっているホストから比較的空いているホストに仮想マシンをマイグレーションすることで負荷を分散させ，より効率の良いハイブリッドクラウド環境の利用を実現する．

2.2　障害対応のためのマイグレーション

ハイブリッドクラウド環境上では，多くのインスタンスが相互に関連しあって動いている．そのため部分的な障害が重大な障害へと発展してしまう可能性がある．

そこで，本研究では，障害発生時も，仮想マシンのマイグレーションによって障害を避けることができるようなルートを設定し，障害の影響を最小限に抑える．

また，ホストとなるサーバが不調な際に，そのサーバを避けてインスタンス配置をすることも可能にする．

3　想定するクラウド環境と実験システムの構築

本研究で想定するクラウド環境を，IaaS のクラウド環境構築ソフトウェア OpenStack を用いて構築した．この構築した 2 つのクラウド間でリソースの転送を行うことで，ハイブリッドクラウド環境が実現される．この環境において，それぞれのクラウド内でコントローラノード，ネットワークノード，コンピュートノード 4 台の計 6 台からなるネットワークを構築した．ここで，コントローラには NTT 研究所が開発を行っている Ryu のコントローラ機能を用いることとする．構築したハイブリッドクラウド環境を図 1 に，各ノードのスペックを表 1 に示す．

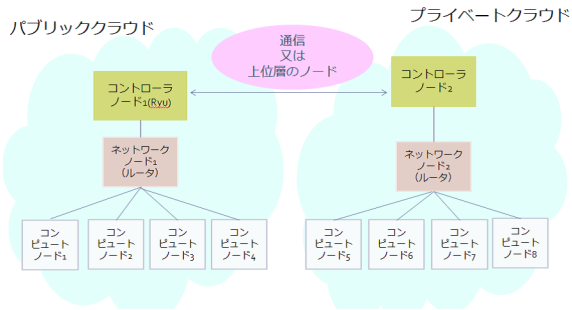

図 1: 構築したハイブリッドクラウド環境

表 1　Public Cloud and Private Cloud’ s Node Servers

OS	Linux 3.13.0-43-generic
CPU	Intel(R) Xeon(R) CPU E3-1270 V2 @ 3.50GHz 4C/8T
Memory	16GB
Disk	500GB

4 OpenStack 上でのマイグレーションの基礎実験

本実験ではコマンドラインを用いて OpenStack 環境のコンピュートノード下に仮想マシンを作り，そのマイグレーションについて考察を行った．

ライブマイグレーションの時にはマイグレーション先の指定ができるが，そうでないときには OpenStack の nova のスケジューラ機能がマイグレーション先のノードを自動で指定する．また，仮想マシン作成時にも，どのコンピュートノード下に置かれるのかはスケジューラにより自動選択される．

そこで仮想マシン作成時からユーザの意思によって作成先やマイグレート先のノードが選択できるように設定をする．

4.1 単数の仮想マシンのマイグレーションの所要時間の計測

構築したクラウド環境の片方で，OpenStack のダッシュボードである Horizon を用いて仮想マシンのマイグレーションを行い，かかった時間と移動したホストを図??に示す．

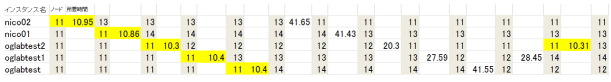

図 2: 仮想マシンを 1 つづつマイグレーションした所要時間（同上）

この結果より，compute11 から他のホストへのマイグレーションの際には最短所要時間は約 10 秒と，そのほかの所要時間が約 30 秒から約 40 秒であるのに対してかなり短いことがわかる．これは初期設定の際に，compute11 のディスクを共有ディスクとして定めたことが影響していると考えられる．また，各マイグレート先ホストにいくつ仮想マシンが存在するのかについては所要時間との関連は見られない．

また，マイグレート先ホストについては中にある仮想マシンがすべて同じ大きさの場合は，その個数が均等になるようにランダムに配置されていると推測できる．

4.2 複数の仮想マシンのマイグレーションの所要時間の計測

同様に仮想マシンのマイグレーションを複数個同時に行い，かかった時間と移動したホストを図 3 に示す．なお実験の操作上，完全に同時にマイグレートをスタートさせたわけではなく，それぞれ上の行の仮想マシンから順に連続でマイグレーションさせた．

インスタンス名	ノード	所要時間											
nico02	11	7.6	13	36	12	37.39	14		14		14		14
nico01	11	11.33	14	38.57	13	39.07	14		14		14	33.33	11
oglabtest2	11		11		11		11	7.57	13		13	27.7	11
oglabtest1	11		11		11		11	8.63	12	27.23	13		13
oglabtest	11		11		11		11		11	13.1	12		12
合計所要時間		18.93		43.67		44.93		13.46		27.23		33.33	

図 3: 仮想マシンを 2 つ同時にマイグレーションした所要時間

この結果により，合計の所要時間は別でマイグレーションさせた時よりも短いことが多いことがわかる．

一方でそれぞれの仮想マシンがマイグレート時間は，1 つだけでのマイグレート時とは少し異なった．compute11 からのマイグレート時間は，ほとんどの場合 1 つだけでのマイグレート時よりも明らかに所要時間が短かった．一方で，その他のホストからのマイグレート時間は 1 つだけでのマイグレート時よりも長くなっていることが多いことが分かった．所感として，この場合は実際の実行時間よりもマイグレートの準備と終了に多くの時間がかかっていたように感じた．

5 OpenStack によるマイグレーションの制御

緊急地震速報のような外部からの負荷変動を予告するシグナルが入って来る際に，それをトリガーとしてシステム構成を切り替えるシステムを考える．これは以下のような動作を行う．

- このトリガーを機に，どのリソースをどうすべきかを判断する．今回の場合，どの仮想マシンをどこにマイグレートすべきかを判断する．
- それを最適に行うためにネットワークを制御する．これを OpenStack のクラウド環境で、OpenFlow を用いてネットワーク制御を行う．例えば、緊急時には他のもともと行っている処理による通信を抑えて，マイグレーションを優先的に実行できるようにする，などが考えられる．

5.1 機能実現時の性能評価

上記のような機能が実現した場合に期待される性能を，手動で実現しその性能を評価した．

バックグラウンドに負荷がある場合とない場合のマイグレーションにかかる時間を測る．バックグラウンドの負荷として今回は iperf コマンドを用いてパケットを送信させた．マイグレーションごとに差はあるが，パケットを流しながらのマイグレーションでは流さない場合と比べて，平均して 8.724 秒の遅延が確認できた．これにより，先述のネットワーク制御の性能を評価できた．

6 まとめと今後の課題

本研究では，実環境に沿って構築したクラウド環境上で，どのクラウドにどのリソースを配置すればよいかを判断し構成を切り替えるシステムを提案する．最終的に，外部情報に基づいてこの切り替えを可能にする．

今回は本システムを手動で実装した．今後は，このシステムを自動化し，さらに現在はクラウド内で行っているマイグレーションをクラウドをまたいで行う．これにより，ハイブリッドクラウド上で本システムを利用することが可能になる．この際には，OpenFlow を用いることが期待される．本研究では Ryu のコントローラを導入してネットワークの制御をすることを考える．

その後，マイグレーションの対象を仮想マシンから，データベースなど必要最小限のデータをを抽出したものへと変える．これによりコストの削減や，データ喪失の危険性の低下を図る．最後にこれらを自動化しミドルウェアの形で実装する．

謝辞

本研究を進めるにあたり，NTT　三島健様に数多くの助言を賜りました．深く感謝いたします．

参考文献

[1] OpenStack：http://www.openstack.org/
[2] Ryu：http://osrg.github.io/ryu/
[3] 西出 彩花，小口 正人，三島 健：「OpenFlow を用いた OpenStack クラウド環境におけるリソース転送制御に関する検討」，DEIM2015，2015 年 3 月発表予定．