クラウド環境におけるネットワーク制御時のマイグレーションの性能考察

西出彩花[1]　小口正人[1]

概要： 近年,増大するシステムやデータの管理のためにクラウド基盤が用いられることが多い.クラウド基盤においては,負荷分散やリソースの最適配置を目的として,インスタンスマイグレーション等のリソース転送が行われる.このような環境は平常時には安定して動作するよう設計されているが,例えば震災などの何らかの大きなイベントが起こった際に,大量のデータをバースト的に転送する事になり,クラウド内・クラウド間のネットワークに大きな負荷が掛かり,帯域が不足する経路ができてしまう危険性を大いに含んでいる.そのため,その時々のネットワークの状況に基づいて,混雑する経路を迂回して別の経路に振り分けるなどの対策が必要である.これは統計情報に基づいたデータ転送処理により実現に近づけることができる.そのような処理には OpenFlow を用いたネットワーク制御が有用である.また,緊急時には,通常時とは異なるシステム構成が望ましいことが多い.この実現のために,クラウド内のリソースを転送したり,接続ネットワークの構成を変更したりする必要がある.このようなシステム構成の変更を極めて短い時間に実行するためには,これらを制御し,最適化を行う仕組みが必要である.本研究では OpenStack を用いて実験用のクラウド基盤を構築し,OpenFlow コントローラを用いてデータのマイグレーションをコントロールすることによって,通信性能の向上とクラウドシステム制御の安定化を図ることを目標とする.

Performance consideration of migration at the time of the network control in cloud environment

SAYAKA NISHIDE[1]　MASATO OGUCHI[1]

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/スケールダウンす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要因のひとつである.

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少ない.

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利用することができる.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て,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でも,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るなど,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.この理由の一つとして,大規模で複雑化しているクラウドの構成は,手動での制御の限界に近づいていることが挙げられる.例えば大規模災害時に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急なアクセスの増加や部分的なネットワークの故障に対応できる可能性がある.しかし,人間が手作業で行うには,規模的にも速度的にも限界がある.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.

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.緊急時のクラウド環境は,通常時と異なるシステム構成になると考えられる.従って,物理的に異なるノードへインスタンスを移動させたり,クラウド内・クラウド間のネットワーク構成を動的に変更したりすることが必要となる.本研究はクラウド環境におけるこのようなリソース制御の実現を目指す.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] [2] を用いてクラウドを構築し,そのシステム上でリソースの転送を制御する仕組みを検討する.その際,クラウド内外の

1　お茶の水女子大学

ネットワークは OpenFlow コントローラ [3] [4]を用いて制御する.先述した例の場合には,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.なお大規模災害時等にどのような形で外部から情報を取り込むかについては,本研究では検討の対象としない.どのようなシステム構成が適切であるかを判断できる情報が,何らかの形でシグナルとして飛び込んでくるものと仮定している.例えば [5] では,緊急地震速報をインターネット経由で受信し,これをトリガとしてネットワークの経路制御を切り替える手法を検討している.また [6] では,緊急時にソーシャルメディア情報を利用してネットワークやシステムの障害を検知し,シグナルとして発信する手法を実装している.本研究ではこのような手法で発信されたシグナルを受け取り,状況に応じてシステム構成を最適化することを検討する.

2. クラウドにおけるリソース配置の最適化

クラウドは極めて多くのリソースから構成され,そのリソースをどう配置するのが良いかは状況により変化する.前節で述べたように,通常時のシステム構成と大規模災害などが起きた時の緊急時の構成では大きく異なる可能性がある.そこで本研究では特に仮想マシンのマイグレーションに焦点を当て,以下のようにリソース配置の最適化を検討する.

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,特定のノード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ノードから比較的空いているノード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

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,障害発生時に,仮想マシンのマイグレーションによる構成の切り替えによって,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ことを目標とする.例えば,緊急地震速報でこれから大規模な地震が来る可能性が出てきた場合に,マイグレーションによって離れたところにリソースを避難させるということが考えられる.

3. 本研究の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.クラウドを 2 組作成し,これらの間をネットワークで接続することでハイブリッドクラウド環境が実現される.実験システムにおいては,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 台からなるクラウド環境が構築される.

実験システムの各ノードのスペックを表 1 に表す.

表 1 Public Cloud and Private Cloud's Node Servers

OS	Linux3.13.0-43-generic
CPU	Intel®Xeon®CPU E3-1270@3.50GHz 4C/8T
Memory	16GB
Disk	500GB

OpenStack のメリットの 1 つに,コンポーネントが細かく分けられていてモジュール化されているので,自由度が高いということが挙げられる.今回用いている IceHouse には,ハイパーバイザと仮想マシンの管理をする Nova,オブジェクトストレージ機能を率いる Swift,スナップショットの管理などをする Glance,認証を司る Keystone,ダッシュボード管理の Horizon,仮想ネットワーク管理の Neutron,仮想ボリューム管理の Cinder,OpenStack 全体のリソース量を計量する Ceilometer,オーケストレーション機能を提供する Heat などがある.本実験での実験システムの各ノードは,表 2 に示すような構成とした.

表 2 実験システムの各ノードのモジュール

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
Keystone	Neutron	Nova
Glance		Neutron
Nova		Ceilometer
Neutron		Swift
Cinder		
Ceilometer		
フロントエンド		
セキュリティグループ		
Swift		
Heat		

構築した OpenStack のハイブリッドクラウド環境を図 1 に示す.

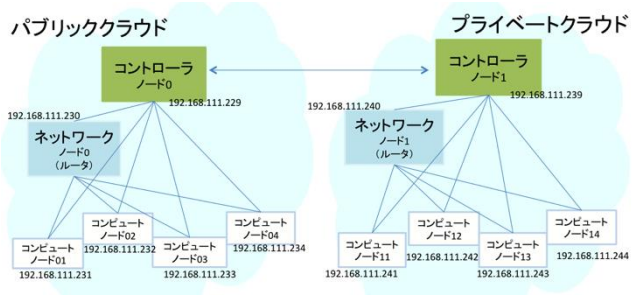

図１　構築したハイブリッドクラウド環境

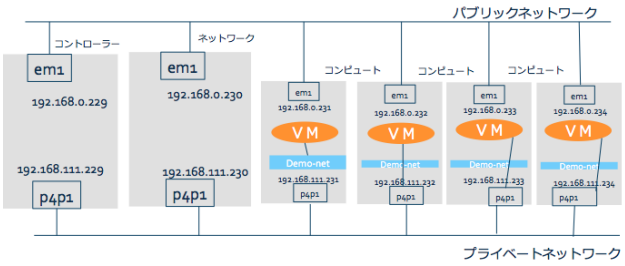

図２　ネットワーク構成図

本研究ではこの実験システムを用いて評価を行う．また,構築したネットワーク構成を図 2 に示す.各ノードはパブリックネットワークを通して外のインターネットの世界や,通信先のクラウドと接続されている.クラウド内のノード間の通信は,基本的にプライベートネットワークを通して行われている.図 1,2 のようなクラウドではこの物理的な環境の中に,仮想マシンが構築され,仮想的なネットワークに接続される.これらが実際のシステムとして機能する.これらの制御はコントローラやネットワークノードから行われるが,実際に仮想マシンや仮想ルータ,仮想ネットワークが動作しているのは複数のコンピュートノード内である.また,マイグレーションの実験を行うために,コンピュートノード間には共有ストレージを設けた.具体的には,コンピュートノードのうち 1 台(compute11)のストレージに,残りのコンピュートノードがリモートから NFS マウントしてディレクトリを共有している.前述のように,OpenStack には共有ディスク環境を必要としないブロックマイグレーションも導入されているが,一部の環境のみにしか対応されていないため,本研究では,共有ディスクを用意してコールドマイグレーションを行う.

4. OpenStack 上でのマイグレーション基礎実験

本実験では OpenStack 環境のコンピュートノード下に仮想マシンを作成し,そのマイグレーションについて考察を行う.本節ではまず基礎実験により OpenStack 環境におけるマイグレーションの基本性能を評価する.

ライブマイグレーション時にはマイグレーション先のノードの指定ができるが,コールドマイグレーション時には 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,どのホスト下に配置されるかはスケジューラにより自動選択される.初期状態においてコールドマイグレーションを実行した様子を図３に示す.

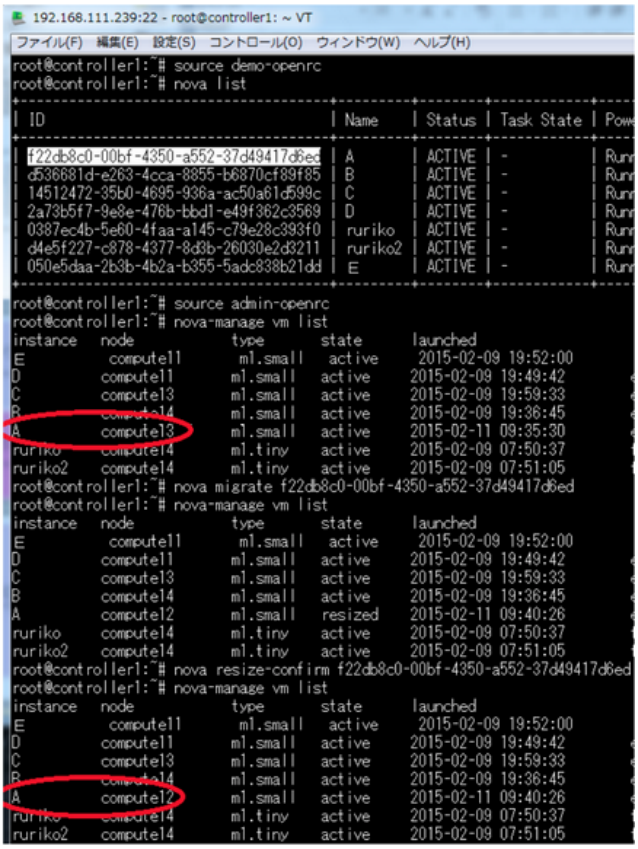

図３　初期状態でのコールドマイグレーション結果

4.1 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一方において,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の実験を行う.この時にかかった時間と,どの仮想マシンがどのノードからどのノードへ移動したかを,図 3,図 4 に示す.ただし,ここで測定したマイグレーションの実行時間は,マイグレーションを開始してから,その完了が Horizon に表示されるまでの時間である.

	11	12	13	14
11		32	35.4	35.1
12	10.5		27.6	33.2
13	10.6	28.7		32.9
14	10.6	31.3	28.9	

図3　単数仮想マシンのコールドマイグレーション結果

	11	12	13	14
11		10.5	10.3	10.4
12	10.3		10.3	10.2
13	10.6	10.5		10.5
14	10.6	10.3	10.4	

図4　単数仮想マシンのライブマイグレーション結果

図 3 のコールドマイグレーションにおいて,マイグレーション元が 11 の際の最短所要時間は約 10 秒と,他の結果に比べてかなり短いことがわかる.これは前節で述べたように,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の仮想マシンの数によって転送性能が変わることが考えられたが,ここで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,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なう. なお実験の操作上,完全に同時にマイグレートを開始してはおらず,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を行う.この結果により,複数のマイグレーションの合計の所要時間は,それぞれを単独でマイグレーションさせた際の合計よりも短い場合が多いことがわかった. また,それぞれの仮想マシンがマイグレートされるのにかかる時間は,単独でのマイグレート時とは少し異なった.これにより,複数のマイグレーションが並行して動作していることが確認できた.

また,共有ディスクを保有している compute11 からのマイグレート時間は,ほとんどの場合,単独でのマイグレート時よりも明らかに所要時間が短い.一方で,その他のノードからのマイグレート時間は単独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わかった. この時,実際のデータ転送そのものの実行時間より,マイグレートの準備と終了に多くの時間がかかっていると考えられる.

5. OpenStack によるマイグレーションの制御

5. 1 緊急時のクラウドの制御機能

本節ではクラウドに対し,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くる際に,それをトリガとしてシステム構成を切り替える方式について検討する.これは以下のような動作を行う.

・ このトリガを機に,どのリソースをどのように配置すべきかを判断する.

・ マイグレーションを最適に行うためにネットワークを制御する.

まず,トリガがかかった際に,どの仮想マシンをどこへマイグレートすべきかを判断する.これを OpenStack のクラウド環境において実装し,OpenFlow を用いてネットワーク制御を行う.具体的な制御としては,緊急時には他に元から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ことが挙げられる.

5. 2 機能実現時の性能評価

上記のような機能を実現すると,緊急時により短い時間で最適なシステム構成へとリソースの配置を変更することができる可能性がある.ここで,マイグレーションを実行する際に,バックグラウンドに負荷がある場合と,これを抑えてマイグレーションを優先させた場合の性能比較を行う.バックグラウンドの負荷としてノード間の通信を想定し,通信性能測定ツールである Iperf [8]を用いてパケット通信を行う.この結果を表にしたものを図 5 に示す.

iperfあり

コールド

	11	12	13	14
11		52.3	45.4	57.4
12	11		49.8	56.8
13	11.4	48.9		51.1
14	10.8	52.3	42.2	

ライブ

	11	12	13	14
11		18.3	18.2	18.2
12	10.9		10.9	10.5
13	10.9	10.5		10.4
14	10.7	10.9	10.6	

iperfなし

コールド

	11	12	13	14
11		32	35.4	35.1
12	10.5		27.6	33.2
13	10.6	28.7		32.9
14	10.6	31.3	28.9	

ライブ

	11	12	13	14
11		10.5	10.3	10.4
12	10.3		10.3	10.2
13	10.6	10.5		10.5
14	10.6	10.3	10.4	

図5 バックグラウンドの負荷の有無とマイグレーションの所要時間の関係

図 5 においては,いずれの場合も compute11 がマイグレーション元の場合に所要時間が短いことがわかる. これは前述の compute11 の共有ディスクの影響だと考えられる. また,その他の場合,バックグラウンドに Iperf による負荷がある場合に比べて,これを抑えた場合にマイグレーションの所要時間が短くなっており,バックグラウンドのトラフィックを制御する効果が確認できる.

次に,ホスト内の仮想マシンから iperf コマンドを用いてパケットを流すことにより,マイグレーションの性能考察を行う.今回はホスト内にある仮想マシンから他のホスト内にある仮想マシンへパケットを流しながらコールドマイグレーションを行う. この結果を表にしたものを図 6 に示す. 列がマイグレーション元ホスト,行がマイグレーション先ホストを表す. ホストから直接パケットを流す場合に比べてかなり所要時間がかかることから,同じネットワーク上にネットワーク稼働中の仮想マシンがあることはマイグレーションを行う上でかなり大きなボトルネックとなることがわかる. また,ライブマイグレーションの結果は負荷をかけない場合とほぼ等しい

host12→host11

	11	12	13	14
13	17.6	65.9		47.7
14	16.9	57.5	60.6	

host12→host13

	11	12	13	14
11			55.8	
13	22.1	58.3		106

host11→host12

	11	12	13	14
11		55.7	57.1	56.4
12				
13	17.8	53		
14			57.2	52.7

図6 仮想マシンからパケットを流す際のコールドマイグレーションの所要時間

図 6 の結果においても,compute11 がマイグレーション元の場合は所要時間が短くなっており,共有ディスクが compute11 にあることの影響が考えられる. また,これ以外の場合には,測定したすべてのケースで所要時間が大幅に増加している. これは,バックグラウンドの Iperf の送信元および送信先,マイグレーション元およびマイグレーション先のいずれかが重なった場合に,のノードの負荷が重く

なるためだと考えられる．さらに,これら 4 つのノードが重ならなかった場合にも compute11 が共有ディスクを持っていることにより,マイグレーションの際にこのノードの負荷が影響を及ぼすため所要時間が長くなることも考えられる．いずれの場合も,バックグラウンドの Iperf のトラフィックを抑えることにより,マイグレーション時間の短縮が期待できる．

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した.また,外部情報に基づいてこの切り替えの実行を制御する仕組みの最適化を検討した.今後は,このシステムの実装を進め,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このシステムを利用することが可能になる.地理的に遠隔にあるクラウド間でこのようなシステムを実現することにより,大きなダメージが予想される地点がある際に,遠隔の安全なクラウドへリソースを転送するようなことが考えられる.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抽出したものへと発展させる.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．

参考文献

[1] OpenStack : http://www.openstack.org/

[2] 中井 悦司,中島 倫明:「オープンソース・クラウド基盤 OpenStack 入門」2014 年 7 月 29 日第一版第三刷

[3] OpenFlow : https://www.opennetworking.org/sdn-resources/openflow/

[4] Ryu : http://osrg.github.io/ryu/

[5] 原 瑠理子,長谷川 友香,小口 正人:「緊急地震速報に基づく OpenFlow を用いたトラフィックエンジニアリング」マルチメディア,分散,協調とモバイル (DICOMO2014) シンポジウム,2G-5, pp.494-497, 2014 年 7 月.

[6] 丸 千尋,榎 美紀,中尾 彰宏,山本 周,山口 実靖,小口 正人:「大規模災害時における Twitter を用いたネットワークシステム制御に有用な情報の抽出」 第 7 回データ工学と情報マネジメントに関するフォーラム(DEIM2015C7-3)2015 年 3 月

[7] Open Networking Foundation : https://www.opennetworking.org/ja/

[8] Iperf - The TCP/UDP Bandwidth Measurement Tool :https://iperf.fr/