WIDE クラウド WG 2011 年度活動報告

石田 渉* 岡本 慶大† 島 慶一‡ 関谷 勇司* 中村 遼§ 堀場 勝広§

2012 年 1 月 4 日

1 はじめに

WIDE クラウドワーキンググループは、今後のクラウド技術の研究開発を推進するために 2010 年 1 月に設立された。複数の WIDE 組織間に渡って運用される広域連邦型クラウドシステムである WIDE クラウドシステムの運用と、それを用いた研究開発を行っている。

2011 年度の主な活動を以下に列挙する。

- WIDE クラウドコントローラ (第 2 章)
- WIDE クラウドネットワーク (第 3 章)
- 甲子園中継 (第 4 章)
- 遠距離狭帯域環境でのマイグレーション (第 5 章)
- 分散ストレージ技術の検証 (第 6 章)
- IPv4-IPv6 トランスレータの評価 (第 7 章)
- LISP を用いたネットワークマイグレーション (第 8 章)
- IETF の活動 (第 9 章)

以降、それぞれの活動の詳細を各章にて報告する。

2 WIDE クラウドコントローラ

WIDE クラウドは各拠点のプライベートクラウドを連結して構成した、連邦型クラウドである。プライベートクラウドの場合には、統一の仕様にてクラウドを構築することができるため、そのマネージメントも単一のソフトウェアにて可能である。

しかし、連邦型クラウドの場合には、異なったハードウェアならびにソフトウェア、異なったネットワークと運用ポリシによって構築されたプライベートクラウドを連結し、マネージメントすることが必要とされる。そのため、WIDE クラウドでは、独自にクラウドマネージメントソフトウェアを設計し、実装した。これは WCC (WIDE Cloud Controller) と呼ばれており、オープンソースソフトウェアとして公開されている。異なったハードウェア、ソフトウェアを統合して管理するために、WCC は図 1 に示す構造にて作成された。リソース管理モジュール、SNMP Polling モジュール、ネットワーク制御モジュール、ストレージ制御モジュールは Ruby 言語にて作成され、ユーザインタフェースは Ruby on Rails を使用して作成された。リソース管理モジュールが全ての情報を集め、VM の作成場所やリソースの割り当てを判断した後に、libvirt ならびにネットワーク制御モジュール、ストレージ制御モジュールに対して指示を出すことで、VM の作成、VM へのネットワーク割り当てならびに VM イメージ用のストレージ割り当てを行う。

また、連邦型クラウドの場合には、組織毎にリソースを提供するポリシが異なる場合があるため、拠点単位で、他拠点に供出する CPU、メモリ等のリソースを定義することができ、かつ余剰リソースを組織間にて融通することができるよう設計、実装した。

2011 年に行われた WCC のバージョンアップと、バージョンアップに伴って加えられた主な新機能に関して以下にまとめる。

02/06 : Version 1.8 Release (Strict VM migration) VM マイグレーションの移動対象先を厳格に制限した

03/02 : Version 1.9 Release (NEMO Support) NEMO モバイルルータを利用したネットワーク移動透過性を導入した

*東京大学
†奈良先端科学技術大学院大学
‡株式会社 IIJ イノベーションインスティテュート
§慶應義塾大学

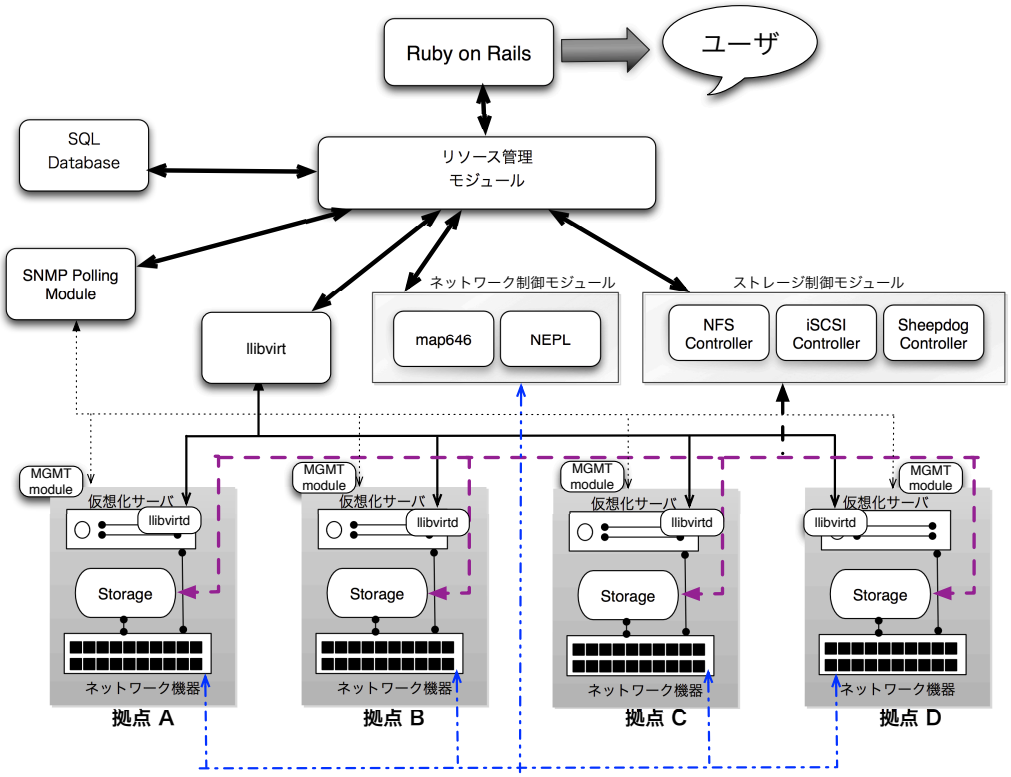

図 1: WCC アーキテクチャ

03/02 : Version 1.10 Release (Storage location) VM が利用するストレージの場所を指定できる機能を追加した

05/01 : Version 1.11 Release (Change VM owner & VM Hibernation) VM の所有者を変更する機能と VM ハイバネーションの機能を追加した

05/17 : Version 1.12 Release (VM address info) VM が DHCP にて取得している IPv4 アドレスの情報を取得できるようにした

10/21 : Version 1.13 Release (Storage migration) ストレージマイグレーション機能を導入した

12/17 : Version 1.14 Release (SUMIKAWA) メッセージ表示や VM 所有者によるフィルタリング機能を導入した

3 WIDE クラウドネットワーク

IaaS クラウドは、単一のネットワークセグメントを基本に構成される場合が多い。これは、コントローラ技術が VM を制御したり、ストレージ機器とサーバが通信を行うにあたっては、それぞれが同一のネットワークセグメントに存在する方が制御しやすいからである。また、VM がサービスに使うネットワークは、VLAN 等の技術を用いてサーバに提供され、サーバ内部で VLAN を分割してそれぞれの VM に提供される構成が多い。そのため、VM をサーバ間にてマイグレーションさせるにあたっては、移動元サーバと移動先サーバで同様のネットワーク構成が利用できなければ、VM マイグレーションや VM 耐故障技術を利用することができない。すなわち、サーバ間にてネットワーク移動透過性を確保する必要がある。

一般的な IaaS クラウドでは、サーバ間ならびに拠点間を物理的な回線で直接接続することで、サーバ間に同様の VLAN を設定できるようにしている。もしくは、IP トンネル技術等を用いて、拠点間で VPN

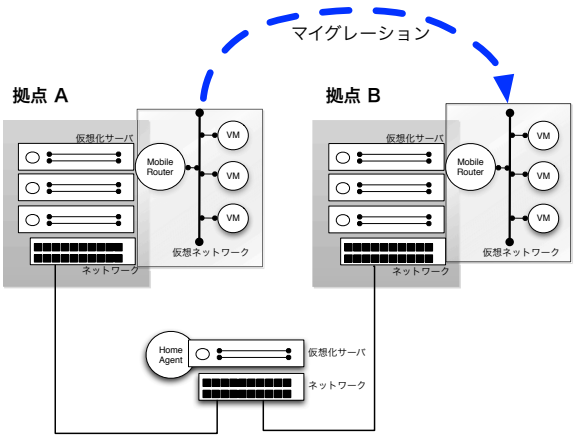

図 2: NEPL によるネットワークマイグレーション

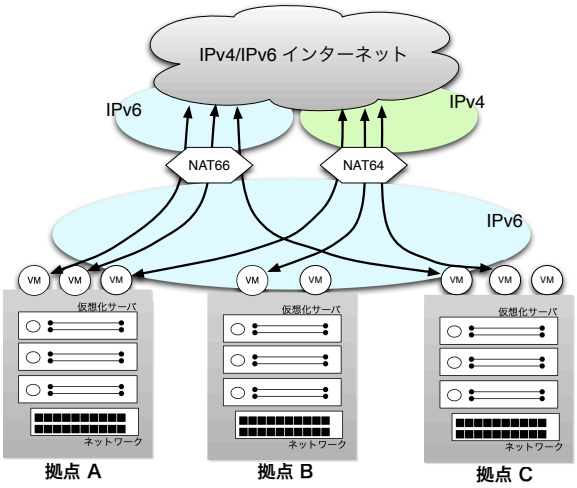

図 3: map646 によるアドレス変換

(Virtual Private Network) を構成することで、離れた拠点にあるサーバ同士で同じネットワークを利用することを可能にしている。しかし、拠点間をむすぶ専用の回線を利用するためには、拠点が離れているほど高額のコストが必要となり、また VPN を利用する場合でも、利用するネットワークが増減するたびに、VPN 設定を変更する管理コストが必要となる。

そこで WIDE クラウドでは、専用の回線や固定的な VPN 設定を用いることなく、拠点間にてネットワーク移動透過性を実現する手法を導入した。WIDE project にて開発されたソフトウェアである、UMIP (USAGI-patched Mobile IPv6 for Linux) ならびに NEPL (NEMO Platform for Linux) を用いて、拠点間において自由にネットワークを移動させることのできる仕組みを構築し、ユーザに対して提供した。ユーザは、自分専用のネットワークをどの拠点にも移動させることができるため、専用の回線で接続されていない拠点にも、VM を移動させることが可能となった。NEPL を用いたネットワーク移動透過性提供の仕組みを、図 2 に示す。

ネットワーク移動透過性を利用する VM は、Mobile Router (MR) が提供する仮想ネットワークに接続される。VM を拠点間でマイグレーションする際には、MR もあわせてマイグレーションすることにより、移動した先の拠点においても同じネットワークを利用し続けることを可能としている。

また、WIDE クラウドでは、VM には基本的に IPv6 アドレスのみが割り当てられる。IPv4 アドレスが必要な場合には、DHCP によってプライベート IPv4 アドレスを取得することが可能となっているが、あくまでも補助的な扱いとなっている。VM には、割り当てられる IPv6 アドレスの他に、外から VM にアクセスするためのサービス用 IPv6 アドレスならびに IPv4 アドレスを割り当てることができる。インターネットから VM に割り当てられたサービス用 IPv6 アドレスに対して通信があった場合には、NAT66 機能により IPv6→IPv6 アドレス変換が行われ、VM に接続される。同様に、サービス用 IPv4 アドレスに対して通信があった場合には、NAT64 機能により、IPv4→IPv6 プロトコル変換が行われ、VM に接続される。この概要を図 3 に示す。すなわち、VM は IPv6 に対してのみアプリケーションを設定しておくだけで、IPv6 と IPv4 どちらのクライアントに対してもサービスを提供することができる。さらに、VM を拠点間にてマイグレーションした場合にも、アドレス変換によってサービス用 IP アドレスを変更せずともサービスを継続することが可能となっている。

また、IPv4　IPv6 プロトコル変換にあたっては、IPv6 アドレスの下位 64 ビットの中に、IPv4 アドレスを埋め込むことによって、NAT のように状態を管理する必要のない、ステートレスなプロトコル変換を可能とした。IPv4　IPv4 変換を行う NAT の場合には、アドレスとポート番号の組み合わせを NAT 内部で管理しておく必要があり、クラウドと外部ネットワーク

の出入り口を冗長化する場合には、NAT 間でその情報を同期しておく必要がある。一方、map646 による IPv4 IPv6 プロトコル変換の場合には、マッピング情報が IPv6 アドレスに埋め込まれているため、通信途中でネットワーク出入り口が切り替わったとしても、問題なく通信を継続することが可能となっている。

現在の WIDE クラウドでは、東京と大阪の 2 地点に map646 サーバを設置し、WIDE クラウドとインターネットへの接続を冗長化している。それぞれの map646 サーバでは OSPF ならびに OSPFv3 が動作しており、クラウド内部で VM が利用しているアドレスを、WIDE インターネット内部に OSPF、OSPFv3 にて広告している。東京と大阪どちらかの map646 サーバやネットワークに障害が発生した場合にも、OSPF による経路制御によって WIDE クラウドと外部ネットワークとの通信が自動的に切り替わり、かつステートレスな IPv4 IPv6 変換であるために、TCP コネクションが切断されることなく、通信を継続することが可能となっている。

4 甲子園インターネット中継

全国高等学校野球選手権大会 (以下、甲子園) のインターネット中継は、朝日放送、奈良先端科学技術大学院大学、NTT スマートコネクトの共同プロジェクトとして 1998 年から行われており、試合経過や過去の試合のハイライトなどを Web でリアルタイムに提供している。 2010 年度同様、この Web コンテンツの一部を WIDE クラウドを利用して配信した。また今年度は map646 を利用したシステムを構築し、配信運用中の VM の広域ライブマイグレーションも可能とした。

4.1 甲子園システムの概要

甲子園システムは、3 種類のサーバから構成される。

koshien 甲子園の Web コンテンツ全体を配信

sentryc ライブ配信ページのパラパラアニメを配信

sentryd NTSC の映像入力から静止画の切出しと圧縮を行い sentryc へ提供

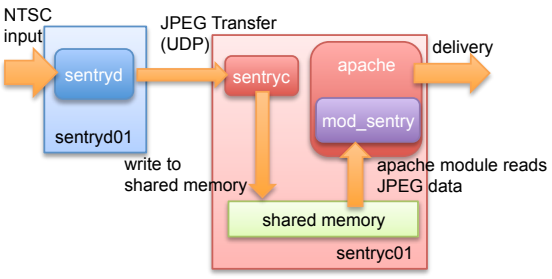

図 4: sentry サーバプログラムの動作

sentryd と sentryc の動作モデルを図 4 に示す。sentryd は入力された映像をパラパラアニメの要領で 1 秒毎に JPEG データとして切り出し、sentryc サーバに UDP 転送する。JPEG 画像データを受け取った sentryc は共有メモリへ JPEG データを書き出す。同一サーバ上で動作している apache には mod_sentry と呼ばれるモジュールを組み込んであり、特定の URL にアクセスするとこの共有メモリの内容が配信される。

koshien の配信する Web ページには画像表示用の JavaScript が配置してあり、この JavaScript が一定間隔で sentryc から画像を取得することで、パラパラアニメのように動作し、ストリーミングが閲覧出来ないような環境下でも、試合進行や球場の雰囲気を伝えることができる。

4.2 実験トポロジ

今年度の甲子園システムの概要図を図 5 に示す。昨年度は VM を配置する拠点のネットワークを利用し、グローバルアドレスを VM に直接設定しため、他の拠点への VM マイグレーションが不可能であった。今年度は VM を WIDE クラウドの IPv6 広域 L2 セグメント (VLAN6) に接続し、拠点間での VM マイグレーションを可能とした。map646 のサーバは、VLAN6 の内に RA を流しており、VLAN6 に接続している VM のデフォルトゲートウェイになっている。また、map646 を利用している VM のグローバル IPv4 アドレスは、複数の map646 サーバより等コストの OSPF によって経路広告され、広域での負荷分散と高可用性を実現している。

なお、sentryd から sentryc へのパラパラアニメの転送は、JGN-X 経由で sentryd を VLAN6 へ接続することで実現した。

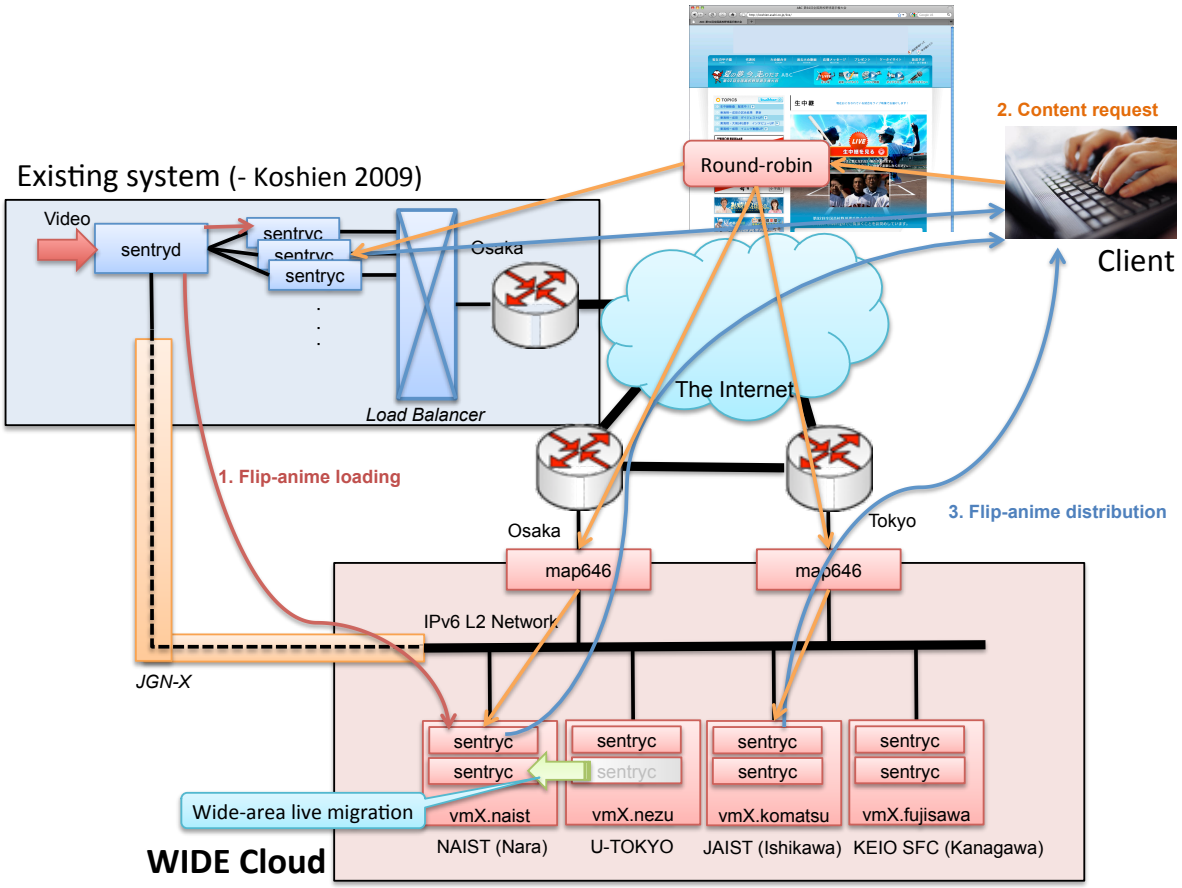

図 5: 2011 年度甲子園システムネットワーク概要図

4.3 運用報告

甲子園システムは開幕の 8 月 6 日より運用を開始した。WIDE クラウド上の VM を配信に投入したのは、大会 8 日目の 8 月 13 日からである。今回は、根津、奈良、北陸に各 1VM の計 3VM を初期配置し、東京と大阪の map646 サーバで経路広告および IPv4/IPv6 パケット変換を行った。

4.3.1 map646 による広域分散配信

map646 を利用した広域負荷分散は狙い通り動作し、図 6 に示すとおり、東京と大阪でほぼ同量のトラフィックの吐き出しとなった。また、運用中に大阪の map646 サーバがダウンするというトラブルがあったが、東京の map646 サーバ経由で問題無く配信された。最もトラフィックが多かった時間で 120Mbps、5000 コネクションを記録している。

しかし、選手権が進み配信が高負荷となってきた際に、map646 サーバが CPU リソースを使い果たし、画像が配信されないというトラブルが発生した。これらの map646 サーバは、WIDE クラウド上の他の VM でも利用されているため、Web ページが見れなくなるなどのトラブルも発生した。図 7 にトラブルが起きた際の東京と大阪の map646 サーバからのトラフィックを示す。

このトラブルは map646 サーバ上で動作していた stat 取得プロセスが CPU リソースの大半を使い果たしてしまったことにより、map646 のプロセスがパケットを変換できなくなったことにより発生したことが分かった。該当する stat 取得プロセスを停止することで正常に配信が行われることを確認した。

またこの時に大阪の map646 へトラフィックが集中している様子が図 7 からも読み取れるが、原因は不明である。

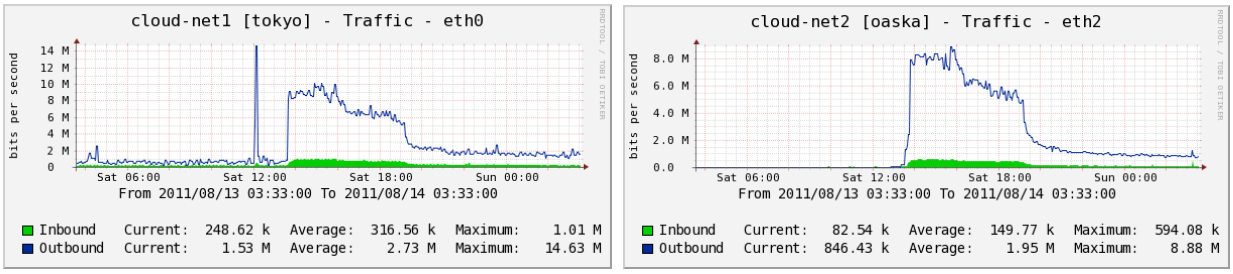

図 6: 東京と大阪の map646 サーバによる負荷分散

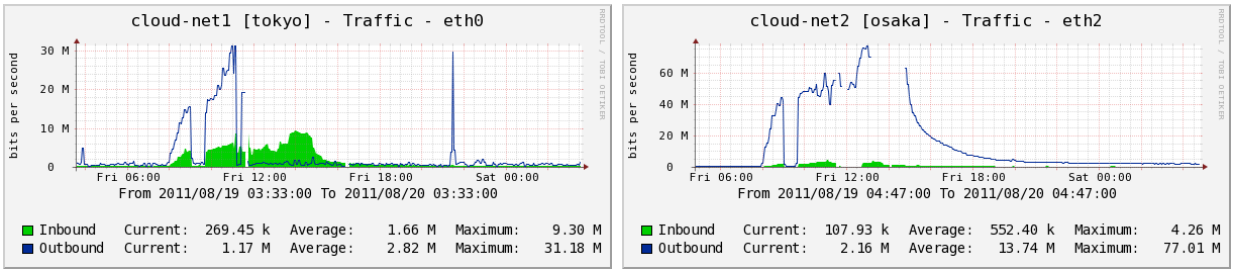

図 7: トラブルが発生した際の map646 サーバのトラフィック

4.3.2 運用中のライブマイグレーション

配信中の VM の広域マイグレーションに関しても問題無く動作し、東京から北陸、奈良などへ VM を移動しても画像は途切れることなく配信された。

しかし、決勝戦の最も負荷の高い時間に VM マイグレーションを行ったところ、マイグレーション開始から 22 分後に突然配信が停止した。調査したところ、Hypervisor から該当する KVM のプロセスが消失していた。KVM プロセスがクラッシュした直接の原因は不明であるが、高負荷時のライブマイグレーションは転送に必要なダーティページが多いためイテレーションの回数が多くなってしまう。また、広域環境であるためネットワークスループットが出ず、転送にかかる時間が想定以上に長くなってしまったことが、要因になっていると考えられる。

4.4 考察と課題

今年度の甲子園インターネット配信実験では、昨年度のクラウドを利用した配信に加え、ネットワーク負荷分散と高可用性を目的とした広域負荷分散配信と、VM の広域ライブマイグレーションの実験を行った。いずれも低負荷時には問題無く動作したが、高負荷時の安定性に難があることが分かった。

今後、map646 サーバのリソース増強や拠点増加、map646 プロセスの改良や、広域分散環境における VM マイグレーションの詳細な監視、マイグレーションスループット向上が必要と考えられる。また、サーバ-クライアント間のルーティング情報を利用した分散配信についても検討を行いたい。

5 遠距離狭帯域環境でのマイグレーション

仮想計算機を、あるハイパーバイザーから別のハイパーバイザーへ移動させる (マイグレーションさせる) 技術はすでに確立しており、ほとんどの仮想化環境で提供されている。ただし、マイグレーション可能なのは CPU の状態とメモリの移動に限られ、ネットワーク環境やストレージ環境は十分考慮されていない。マイグレーションの前提として、移動前と後のハイパーバイザーが、仮想計算機に対して同一のネットワーク環境を提供していることと、同一のストレージ環境にアクセスできることが必要条件となっている。ネットワークに関しては、2010 年度に報告した NEMO BS[1] 技術を用いたネットワーク資源の透過性技術に加え、今年度は LISP[2] などの別の IP モビリティ技術を用いた技術を研究中である。一方ストレージの透過性に

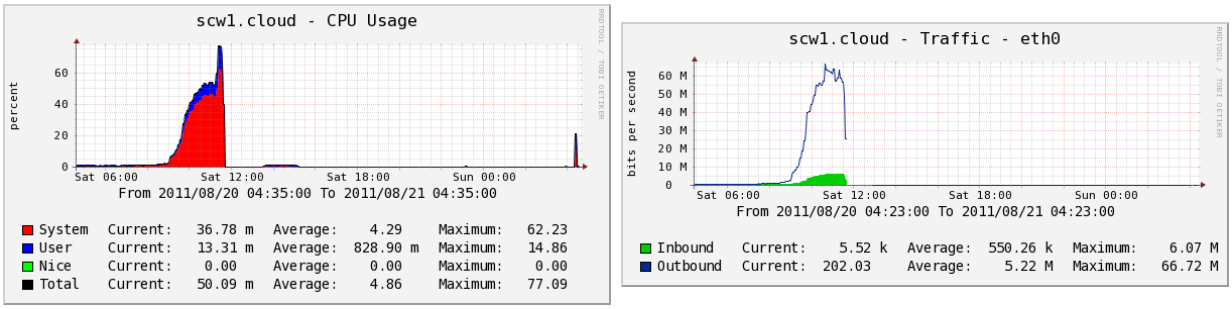

図 8: 高負荷時のライブマイグレーションによる VM クラッシュ

関しては、将来的な技術導入を目指した分散ストレージ技術の検証をすすめつつ (6 章)、現在利用できるブロックストレージマイグレーション技術を用いた実証実験を実施した。

5.1 ブロックストレージマイグレーション

ブロックストレージマイグレーションはオープンソースの仮想化環境 QEMU で提供されているストレージマイグレーション技術である。ブロックストレージマイグレーションは、メモリマイグレーションと同等の手順をストレージデータに対して実施するものであり、移動前のストレージのコンテンツを移動先のストレージにコピーする手法である。ただし、メモリと比べてストレージは容量が大きいため、単純なコピーに加えて、コピーオンライト形式 (QEMU の表記では QCOW もしくは QCOW2 形式) を用いた差分コピーの機能も提供されている。

5.2 実験環境

2011 年 3 月に三重で開催した合宿研究会において、合宿地のネットワークおよびサーバ環境の一部を WIDE クラウドを用いて構築する試みが実施された。合宿開催前に、あらかじめ合宿地で利用するサーバ群を設定しておき、合宿地へのネットワークが開通すると同時に、合宿地に設置したハイパーバイザーへ関連サーバ群を移動する。図 9 に、本マイグレーション実験に関わる合宿ネットワークのトポロジを示す。詳細なトポロジ情報は、合宿実験の報告 (「大規模な仮設ネットワークテストベッドの設計・構築とその運用」) を参照して欲しい。

IPv6 のネットワークの透過性に関しては、昨年度導入した NEMO BS 技術を用い、IPv4 のネットワーク透過性に関しては、本年度報告書の第 7 章で取り上げる map646 ソフトウェアを用いた IPv4-IPv6 ヘッダトランスレータ技術を利用する。3 つのサーバ (www、fu-ring、ns) は、IPv6 アドレス的にはモバイルルータの配下となり、WIDE プロジェクトの NEMO サービスセグメントに収容される形となる。モバイルルータ、および 3 台のサーバは、実際には WIDE クラウドの仮想計算機として構築されており、WIDE クラウドサービスネットワークに設置されたハイパーバイザー上で動作することになる。

仮想計算機のストレージはハイパーバイザーのローカルストレージから提供される。そのため、仮想計算機を、あるハイパーバイザーから別のハイパーバイザーに移動する際にはストレージマイグレーションを実行しなければならない。前述の通り、今回は QEMU が提供するブロックストレージマイグレーションを利用した。

合宿ネットワークは 3 本の対外接続線によってインターネットと接続された。今回、合宿地が三重県の伊勢志摩ということで、光回線などの広帯域回線を手配することができず、2 本の ADSL 回線と 1 本の衛星回線を利用せざるをえなかった。そのため、結果的に狭帯域でのストレージマイグレーションの実験の場となっている。ADSL 回線は 2 本合わせても 1Mbps 程度しか速度を得られず、帯域的には 1.5Mbps を活用できる衛星回線の方が太いということになり、仮想計算機、およびストレージマイグレーションは衛星回線を用いて実行された。

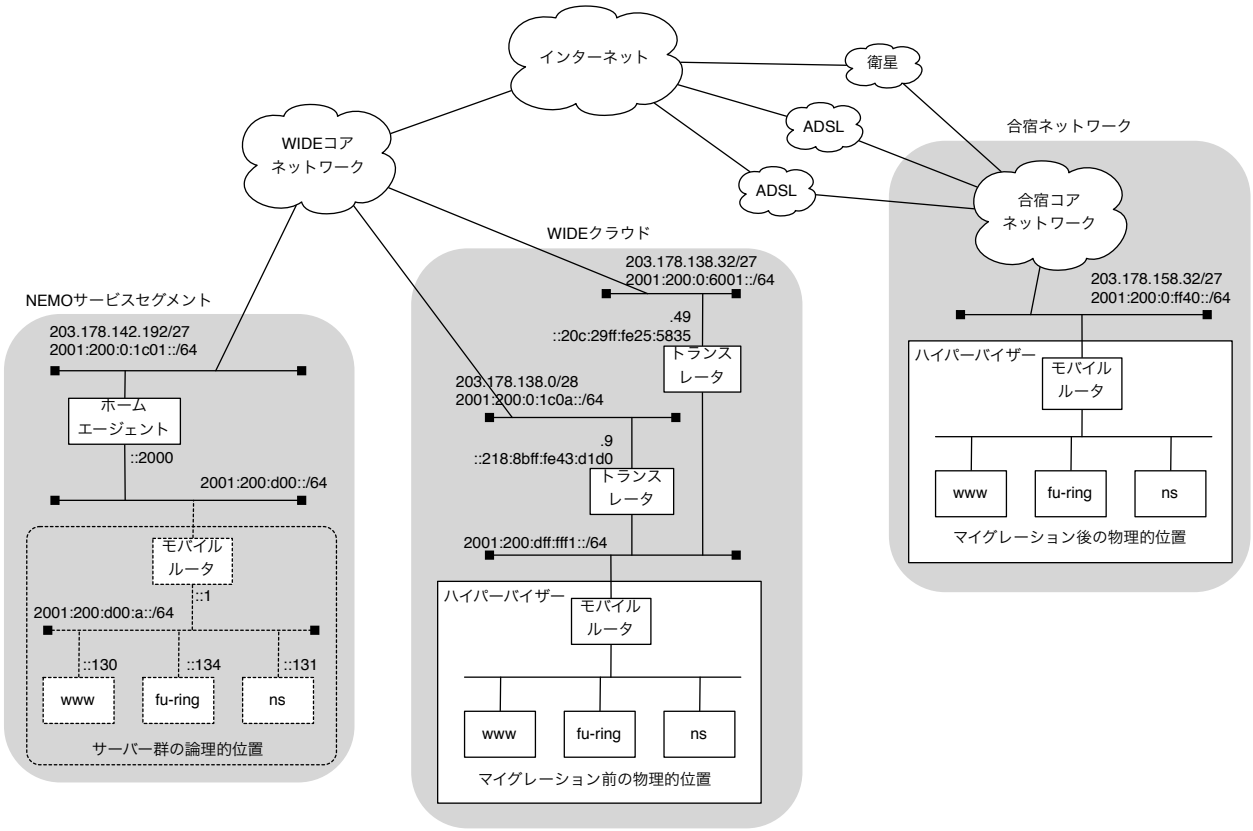

図 9: マイグレーション実験のトポロジ

表 1: 仮想計算機の情報

	メモリ容量	ディスク (差分)
モバイルルータ	512M	470M
fu-ring	1024M	370M
ns	1024M	205M
www	1000M	86M

表 2: マイグレーション時間

モバイルルータ	56 分
fu-ring	53 分
ns	28 分
www	20 分
合計	2 時間 37 分 (157 分)

5.3 結果

それぞれの仮想計算機の情報を表 1 に示す。前述の通り、ストレージは全体をマイグレートするのではなく、ベースイメージからの差分のみをマイグレートする差分ストレージマイグレーションを用いている。ベースイメージは、すべてのハイパーバイザーがあらかじめ持っていると仮定し、合宿ネットワーク内に設置したハイパーバイザーにも、元のハイパーバイザーで利用していたそれぞれの仮想計算機のベースイメージのコピーをインストールした。

表 2 にマイグレーションにかかった時間を示す。この時間には、ストレージのみでなく、メモリマイグレーションの時間も含まれる。表 1 を見ると、相対的にメモリ容量の方がディスク容量より大きく見えるが、使用されていないメモリ空間は転送されないため、実際に転送されるメモリデータは表に示したものよりも少ないと考えられる。それに対して、ディスクイメージ差分は、実際に使われている容量に等しいため、表に示す容量がすべて転送されている。

マイグレーションは、モバイルルータ、fu-ring、ns、

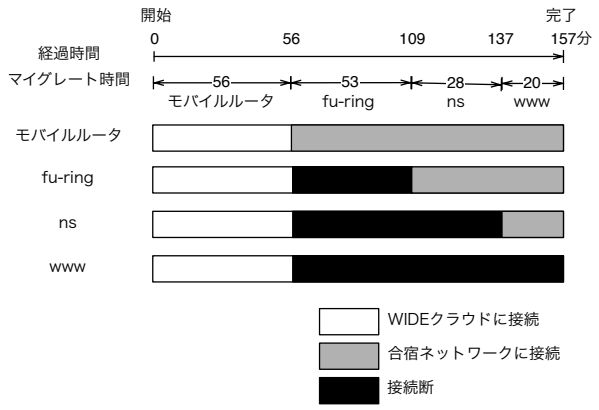

図 10: ネットワーク接続性

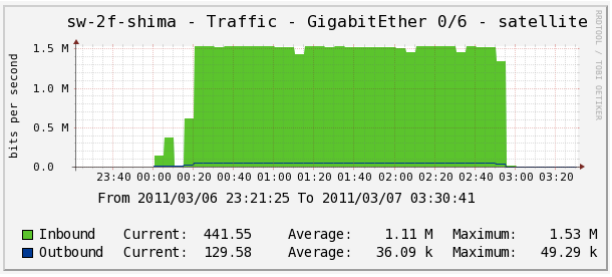

図 11: マイグレーション中の衛星回線トラフィック

www の順で実行された。すべての仮想計算機は QEMU のライブマイグレーション機能を使って、動作させたままマイグレートされているが、ネットワーク的にモバイルルータが存在しない期間が発生するため、インターネットから切り離された時間が発生している。図 10 に示したように、モバイルルータがマイグレーションを完了した瞬間から、fu-ring、ns、www はそれぞれ、53 分、81 分、101 分の間、インターネットへの接続性を失う。

図 11 はマイグレーションが実行されている間の衛星回線のトラフィック量を示すグラフである。開始直後から完了まで、1.5Mbps の衛星回線を完全に埋めている事がわかる。

5.4 考察

仮想計算機技術は、すでに商用製品、商用サービスが広く普及している成熟した分野であるが、広域ネットワークを前提とした運用技術はまだ確立されていない。WIDE クラウドワーキンググループでは、昨年度に引き続き、広域での仮想計算機マイグレーションの技術検討を継続しており、今年度はストレージマイグレーションの検証を実施した。現在提供されている QEMU のブロックストレージマイグレーション技術は、実際に広域ネットワークでも実行できることが確認できたものの、狭帯域のネットワークではそもそもマイグレーションに時間がかかる、また組み合わせるネットワークマイグレーション技術によっては、長くインターネットから切断される時間が発生することも確認できた。今後は、第 6 章でとりあげるような分散ストレージ技術、また第 9 章で言及するような、他の種類のネットワーク透過性技術等も考慮しつつ、広域運用可能な技術開発を継続していく。

6 分散ストレージ技術の検証

WIDE クラウドワーキンググループでは、広域クラウド環境で運用可能なストレージ技術の検証を行っている。今年度は、NTT サイバースペース研究所の森田氏が設計した QEMU 用の分散ストレージ技術 sheepdog[3] を検討し、その実用性を検討した。

6.1 動機

WIDE クラウドワーキンググループでは、複数の組織に広域に跨がった連邦型クラウドコンピューティング環境を目指しており、それを実現するために欠かせない技術が広域運用可能な分散ストレージである。WIDE クラウドワーキンググループが運用する IaaS 基盤 (WIDE クラウド) は、Linux のディストリビューションのひとつである ubuntu と、それに標準搭載されている KVM 仮想化環境を用いて構築されており、その環境と連携して動作するストレージシステムが必要とされている。

現在、WIDE クラウドのストレージとして広域 NFS および iSCSI 技術を用いている。iSCSI サーバは全国 4 カ所 (WIDE プロジェクト根津 NOC、WIDE プロジェクト大手町 NOC、北陸先端科学技術大学院大学、奈良先端科学技術大学院大学) に配置されている。各拠点に NFS サーバが設置されており、iSCSI ディスクをマウントした後に、各ハイパーバイザーに NFS ファイルシステムとしてストレージを提供する形で運用されている。

表 3: 実験ノードの仕様

CPU	Intel Xeon X5670 (6 コア) x 2
メモリ	48GB
ディスク	SATA 500GB x 2
NIC	Intel Gigabit double、Broadcom 5709 quad

現状の構成では、iSCSI あるいは NFS サーバをホストしている組織への到達性を失うと、該当ストレージに収容されている仮想計算機が利用できなくなる。そのため、仮想計算機イメージを分散収容できる広域ストレージシステムが必要となっている。

6.2 sheepdog

sheepdog は NTT サイバースペース研究所の森田氏によって設計された、完全対称型の分散ストレージシステムである。広く普及している仮想計化環境 QEMU のストレージとして利用されることを前提としており、メンテナンスの容易性、規模性を特徴として挙げている。WIDE クラウドワーキンググループでは、sheepdog が広域環境で利用できるかどうかを検討するため、大規模エミュレーション環境を実現できる StarBED[4] を用いて運用可能性を検討する作業に着手した。

6.3 StarBED での実験構成

今回、StarBED から Cisco UCS 200 M2 を 47 台借用し、大規模ノード数での運用現実性を検討した。UCS の仕様を表 3 に示す。

時間の都合上、今回の実験ではすべてのストレージノードが同一 LAN 上に配置され、その上での運用実験となった。ストレージノードとして 46 台を利用し、それぞれのストレージノードが内蔵ディスクのひとつ (500GB) を提供する形で sheepdog クラスタが構成されている。ディスクの合計数が全体のストレージ容量になるが、今回はデータのレプリケーション数を 3 に設定しているため、実際のストレージ容量はその 3 分の 1 となり、およそ 7TB となっている。ほとんどのストレージノードは、ストレージ機能しか提供しない構成となっているが、ひとつだけハイパーバイザーとして動作するノードを構成し、仮想計算機を動作させた。また、別のひとつのノードには iSCSI ターゲット機能を構成し、sheepdog クラスタ外のハイパーバイザーへのストレージ機能提供を試みた。図 12 にクラスタ構成図を示す。

6.4 考察

機材借用期間に限りがあったため、十分なデータの取得に至っていないものの、以下に述べる動作の検証を実施できた。

- 46 台におよぶ実ノードを用いたクラスタ運用の実証
- 既存のクラスタへの、新たなノードの追加
- クラスタ内での仮想ディスクの構成
- クラスタ内での仮想計算機の稼働

すべて、そもそも sheepdog が提供するとされる機能と明言されているものの、実環境に近い構成でその動作を確認できた意義は大きい。また、実験の過程で、ソフトウェアの不具合によると思われる現象を発見し、適宜開発者と連絡を取ることによってソフトウェア自体の品質の改善にも貢献できている。

逆に、検証できなかった、あるいは動作しない/動作不安定だった項目として以下のものが挙げられる。

- クラスタからのノードの切り離し
- クラスタを構成するメンバシップの一貫性が不安定
- iSCSI ターゲット機能

sheepdog では、クラスタの構成は各ノードが自立的に協調することによって実現されており、ストレージノードの参加、離脱は任意のタイミングで実行できることとなっている。しかしながら、検証の過程で、ストレージノードの切り離しにおいて動作が不安定になる場合がみられ、安定した運用ができなかった。現時点では、ノードを切り離した後、クラスタの状態が安定するまで待つことで、ある程度運用可能であることが分かっているが、クラスタが安定する状態が外部から把握しにくいことや、また、事故などによるストレージノード切り離しが発生する場合なども考慮すると、運用は難しいと考えられる。

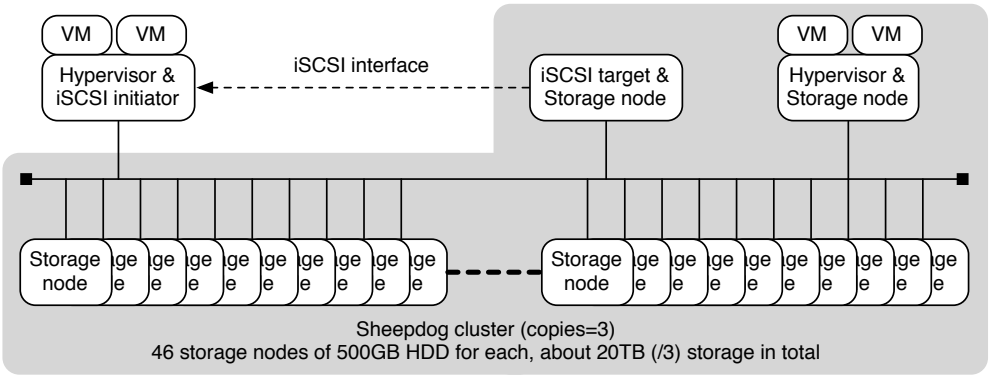

図 12: sheepdog クラスタの構成

sheepdog はストレージノードのグループ管理に corosync ライブラリ[1]を利用している。原因は追求できていないが、運用中、一部のストレージノードのグループメンバが、他のストレージノートと異なる状態が発生することがあった。この状態では、ストレージデータの格納先ノードに一貫性が取れなくなるため、クラスタが正常に動作しない。sheepdog は、今後 corosync 以外にも、zookeeper[2]や accord[3]といったグループ管理機構に対応する予定となっているため、継続して運用性を検証していく予定である。

最後の iSCSI ターゲット機能に関しては、iSCSI イニシエータとして設定したノードからディスクとしてマウントすることはできたものの、仮想計算機からディスクを利用している途中に読み出しあるいは書き込みのエラーが発生することがあり、継続した利用が困難だった。iSCSI の機能の実装によるものか、sheepdog クラスタとのインターフェースによるものか、明確な切り分けができていない。

6.5 まとめ

WIDE クラウドでは、広域で利用できるストレージの仕組みを引き続き検討していく。sheepdog はそのひとつであるが、OpenStack で採用されている SWIFT[4]、分散ファイルシステムとして注目されている GlusterFS[5]などについても、調査検討を進める予定である。

[1] http://corosync.org/doku.php?id=welcome
[2] http://zookeeper.apache.org/
[3] http://www.osrg.net/accord/
[4] http://openstack.org/projects/storage/
[5] http://www.gluster.org/

7 IPv4-IPv6 トランスレータの評価

7.1 はじめに

計算機の速度は年々上昇を続けているものの、単一の計算機の性能向上にだけ頼った方式には限界が見え始めている。近年はネットワークで接続された計算機を統合して利用する、グリッドあるいはクラウドと呼ばれる技術に注目が集まっており、その利用方法やアプリケーションモデル、資源の効率的な運用方法などの研究が盛んである。また、計算機の資源をより細かい粒度で活用するために、ひとつの計算機を仮想的に複数の計算機資源に分割して運用する仮想計算機技術も成熟してきており、今後物理計算機の台数をはるかに越える多数の仮想計算機を用いた計算機ネットワークが構築されると考えられている。

計算機をネットワークに接続する場合、これまでは IPv4 が利用されてきた。しかしながら、IPv4 アドレスの枯渇に伴い、IPv6 への移行が現実のものとなってきている。ただし、IPv4 が枯渇した後も、それまでに分配された IPv4 アドレスが利用できなくなるわけではなく、今後長期間に渡って IPv4/IPv6 のデュアルスタック環境が運用されていくものと考えられている。

計算機資源のネットワーク設定を管理する場合、今後はデュアルスタックでの運用が必須となる。しかしながら、IPv4 と IPv6 は、そのアドレス空間の大きさの差を除いてほぼ同じ機能を提供しているにも関わらず相互に互換性がないため、運用者は類似したネットワークプロトコルを二重に管理する必要がある。

運用管理者としては、サービスに必要なバックエンドネットワークをシングルスタック運用しつつ、外に見える部分はデュアルスタックとして公開することで、運用コストを抑えつつ、今後のインターネットの進化に備えることができるのが理想である。本論文では、仮想計算機資源を提供するシステム (IaaS、Infrastructure As A Service) を対象とし、そのバックエンドを IPv6 のみによるシングルスタック運用としつつ、外部に公開されるサービス部分を IPv4-IPv6 変換ソフトウェアを用いてデュアルスタック化するシステムを提案する。IPv4-IPv6 変換ソフトウェアは特に新しい技術ではなく、IPv6 が提案された初期から存在する。しかしながら、一般的な変換ソフトウェアはひとつの IPv4 アドレスを複数の IPv6 ノードで共有することを前提としており、アドレスの対応状態の管理、セッションの管理などが複雑であり、また通信状態を保持管理しなければならないという特性上、スケーラビリティや耐障害性が問題となる。提案システムでは、対象を IaaS に限定することによって、アドレス変換を一対多から一対一に制限している。これにより、現実的な IaaS サービスを提供しつつ、既存の一対多 IPv4-IPv6 変換ソフトウェアが抱える問題を解決し、シングルスタックでのバックエンド運用を可能としている。

7.2 関連技術

IPv6 の仕様から IPv4 との後方互換性を省くという決断がなされた時点から、すでに両プロトコル間での相互接続問題を解決するための技術の検討は始まっていた。これらの技術は大きく三つにわけることができる。一つ目はアプリケーション層で IPv4 ノードと IPv6 ノードの相互接続性を確保する方法、二つ目はトランスポート層でデータを中継する方法、最後が IP 層でヘッダ変換を行う方法である。

アプリケーション層での接続性の確保はアプリケーション毎に設計された代理サーバーを用いて実現する。IPv4 と IPv6 網の境界に代理サーバーを設置し、特定のアプリケーション (たとえばウェブサービスなど) の要求を受け付ける。代理サーバは受信した要求を解釈し、そこから新たに本来のサーバーへ向けた要求を作成し、適切な IP プロトコルで再送信する。要求に対する応答も、同様の手順で本来の要求元のクライアントへ転送される。代理サーバーはアプリケーションプロトコルを完全に把握しているため、三つの方式の中では最も細かい制御が可能な仕組みとなる。反面、アプリケーションプロトコルが代理サーバーを経由しても問題ないように設計されている必要がある。また、アプリケーション毎に代理サーバーの開発が必要になるなどの短所がある。アプリケーションによっては残る二つの方式で対応可能な場合も多く、本方式は複雑な中間処理を実現したい場合の特殊対応として利用する場合が多い。

トランスポート層での中継では、IPv4 と IPv6 の境界に位置する中継サーバーが、socket レベルで接続を終端、再接続することで両プロトコル間を橋渡しする。代表的な実装として SOCKS64[5] や Transport Relay Translator[6] が挙げられる。SOCKS64 では、SOCKS64 ライブラリを利用することによって、DNS の名前解決とデータ送受信機能を差し替え、通信が SOCKS64 中継サーバーを経由するように調整される。SOCKS64 ライブラリを新たにリンクする必要があるという点で、クライアントの変更が必要になるという短所はあるものの、アプリケーション的には SOCKS64 を意識せず、通常の通信手順を踏むだけで自動的に IPv4/IPv6 変換が行われる。Transport Relay Translator の方は追加ライブラリを必要としない。その代わり、IPv6 ノードが IPv4 ノードと通信する場合には、宛先アドレスとして IPv6 アドレスの一部に IPv4 アドレスが埋め込まれた特殊なアドレス空間を用いる。中継サーバーは、前述の特殊な IPv6 アドレス空間宛ての接続を終端し、IPv6 アドレス中に埋め込まれた IPv4 アドレスを抜き出す。その上で、抜き出された IPv4 アドレス宛に、同じデータを再送信する。特殊な IPv6 アドレスは、通常は手動で計算することはなく、DNS64[7] のような、専用の DNS サーバーを運用することで対応する。

IP 層でのヘッダ変換は、IPv4 における NAT 技術と同等の技術である。この技術は nat64[8] として標準化されている。この方式では、中継サーバーが通信を終端する必要がない。中継サーバーに辿り着いた IP パケットは、事前に定められたアドレス変換規則に則って始点アドレスと終点アドレスが書き換えられる。また同時に IPv4 と IPv6 の間のヘッダ変換も行われる。中継サーバーの機能がトランスポート層にあるのか、IP 層にあるのかが異なるだけで、本質的には二つ目の方法と技術的な差は存在しない。

これらの技術は、IPv6 でのみ構成されたネットワークに接続している IPv6 ノードがインターネット上の IPv4 サーバーに接続するときの問題を解決するために提案されたものである。運用的には、ひとつ、あるいは少数のグローバル IPv4 アドレスが中継サーバーに割り当てられており、多くの IPv6 ノードがそのアドレスを共有して IPv4 インターネットへの接続を確保する形態が想定されている。しかしながら、本論文で想定しているシステムは、IPv6 で構成されたサーバー群に、IPv4 ノードがアクセスするものであり、想定された運用形態が異なっている。IPv6 サーバーを IPv4 ノードに提供するため、グローバル IPv4 アドレスを共有することはできず、IPv6 サーバーアドレスとグローバル IPv4 は一対一で対応づけられることになる。既存の技術でも、本論文で前提としている仕組みを実現することは理論的に可能であるものの、前提とする運用形態が異なるため、実際に運用できる実装は存在しない。

本論文では、IPv6 のみによる管理を前提とし、IPv4/IPv6 変換ソフトウェアを利用したデュアルスタック IaaS サービスの設計を示し、それを実現する IPv4/IPv6 変換ソフトウェアの設計および実装、またその性能評価を行う。

7.3 システムおよびソフトウェアの設計と実装

本節では提案する IaaS システムの全体像と、IPv4/IPv6 変換ソフトウェアの設計を述べる。

7.3.1 IaaS システムの設計と実装

本提案で対象としているのは、仮想計算機資源を提供する IaaS システムである。これは、複数の物理計算機で構成された資源プールから、仮想的な計算機資源を切り出し、サービス運用の部品として利用する仕組みである。サービスによって異なるが、ひとつのサービスは一般的には複数の仮想計算機を用いて構築される。たとえば、ウェブのフロントエンド、ロードバランサー、データベースなどである。あるいは、分散ストレージのようなサービスの場合は、より多くの仮想計算機がフロントエンドの裏側に配置されることも考えられる。デュアルスタックサービスを提供する場合、

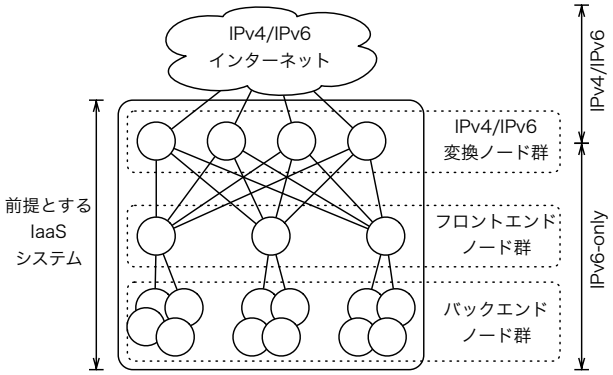

図 13: 提案システム概要図

すべての仮想計算機に IPv4 と IPv6 の両方を割り当てても構わないが、先に述べたように二重管理の問題が発生する。実際、サービスを利用するユーザーから見えるのは、フロントエンドに位置する一部の計算機だけであり、それらの計算機にデュアルスタックでアクセスできれば、それ以外の計算機がデュアルスタックである必然性はない。サービス構築側も、内部のプロトコルがひとつであれば、ネットワーク設計が容易になると同時に、その上で動作させるサービスの開発やテスト工数が削減できる。

また、本提案では IPv4 と IPv6 を一対一で対応させることを前提としている。IPv4 アドレスを複数のサーバーで共有しないため、その消費量については議論の余地があると思われる。しかしながら、サーバーとして運用する場合は、いずれにせよ IPv4 アドレスを占有する必要があり、これまで通りのサービス内容を維持するためには避けられない消費と言える。本提案では、アドレスの割り当てをフロントエンド計算機に制限することにより、無駄な消費を抑えることにしている。

IaaS 内部ネットワークで利用するプロトコルについては、IPv4、IPv6 のどちらでも構わない。今回の実装では将来的な IPv6 への移行を考慮して、IaaS 内部ネットワークは IPv6 ネットワークのみを使う構成としている。

図 13 に提案システムの概要を示す。フロンとエンドとなるノードには、実際には IPv4 アドレスは設定されず、フロントエンドノードの IPv6 アドレスと、それに対応する IPv4 アドレスの情報が IPv4/IPv6 変換ノード群で共有される。アドレス変換は一対一で行わ

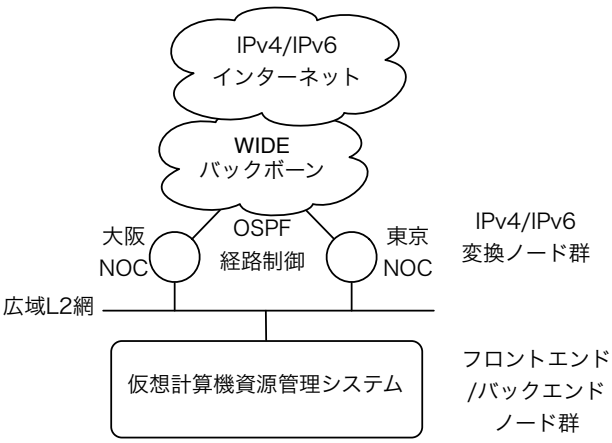

図 14: 提案システムの実構成ネットワーク図

れるので、各変換ノードが通信のセッション情報を保持する必要がなくなり、パケット単位で変換できる。そのため、変換ノードを複数配置することが容易となり、トラフィック増加により変換ノードの負荷が向上した場合でも、変換ノードを追加するだけで対応可能である。また、どこかの変換ノードに障害が発生した場合でも、そのノードを切り離す事で他のノード群でサービスを継続可能となる。図 14 は、我々が実際に構築運用しているネットワーク図である。

最下層に位置する仮想計算機資源管理システムは WIDE プロジェクトによって開発されている WIDE Cloud Controller (WCC)[9] を利用している。WCC は広域 L2 ネットワークを経由して東京と大阪間で接続されており、それぞれの拠点に変換ノードを配置している。変換ノードは OSPF 経路制御プロトコルを用いて、一対一対応された IPv4 アドレスの経路情報を WIDE プロジェクトが運用する L3 ネットワークに広報している。これにより、東京あるいは大阪のどちらかの変換サーバが消滅しても、適切に生き残ったサーバーにトラフィックが向かうようになっている。

7.3.2 変換ソフトウェアの設計と実装

IaaS システム内は IPv6 のみで運用されているため、サーバーにはすべての通信が IPv6 で行われているように見える。IPv4 クライアントからのパケットは一旦変換ノードで受信され、IPv6 に変換される。本ソフトウェアでの変換手順を図 15 に示す。

IPv6 サーバーに対応する IPv4 アドレスは変換テーブルに保持されており、クライアントから入力された IPv4 パケットの終点アドレスに応じて適切な IPv6 終点アドレスへと変換される。IPv6 パケットの始点アドレスは、元の IPv4 パケットの始点アドレス情報を保持する形で、IaaS 内で経路制御されている仮想プレフィックスの下位 32 ビットに埋め込まれる。IPv4 のアドレス空間は、IaaS システム内ではすべてこの仮想プレフィックスに含まれることになる。サーバーからの応答パケットは、上記と逆の手順で変換される。

実装は Linux や BSD で標準的に提供されている tun 仮想ネットワークデバイスを用いた。変換ノードは、IPv6 サーバーに対応付けされている IPv4 アドレスの経路を IPv4 インターネットに広告し、そのアドレス宛のパケットを一旦 tun ネットワークデバイスで受信する。受信されたパケットはユーザー空間でデバイスを開いている変換プログラムに読み込まれ、図 15 の手順に従って変換される。変換の結果生成された IPv6 パケットは、再び tun ネットワークデバイスへ書き出され、IPv6 サーバーへ配送される。逆方向も同様に、変換ノードが仮想プレフィックスの経路情報を IaaS システム内に広告し、すべての仮想プレフィックス宛のパケットを tun ネットワークデバイスで受信、変換し、元の IPv4 クライアントに再転送される。

最後に、本提案で用いる IPv4/IPv6 変換ソフトウェアの制限を述べる。本ソフトウェアは変換ノードでの状態を保持しない設計となっているため、変換のために状態保持が必要となる、フラグメントされた ICMP と ICMPv6 間の変換はサポートされない。また、現在のところ、ペイロード部に IPv4/IPv6 アドレスを含むパケットの変換もサポートしない。後者は実装上の制約であり、理論上は状態を保持しないまま変換可能である。

本ソフトウェアはオープンソースソフトウェア map646 として公開 [10] されており、自由に利用可能である。

7.4 性能計測

本節では map646 の性能を評価する。

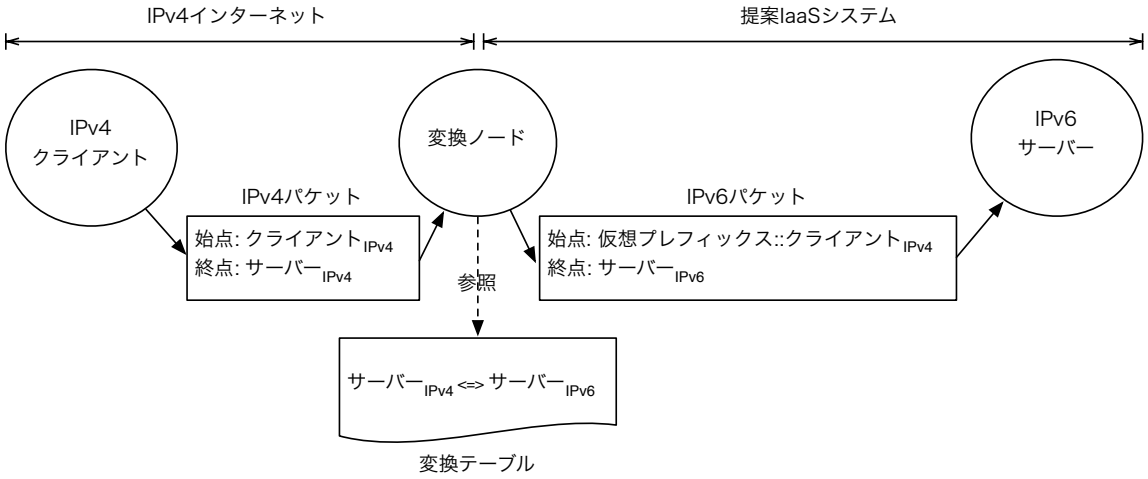

図 15: IPv4/IPv6 変換手順

表 4: 性能評価実験環境: 使用計算機

CPU	Core2 Duo 3.16GHz
Memory	4GB
OS	Linux 2.6.32
Network Interface	Intel 82573L Realtek RTL-8169

7.4.1 評価環境

性能評価実験は次の環境で行った。図 15 に示した、IPv4 クライアント、変換ノード、IPv6 サーバで構成されるネットワークを実際に構築し、遅延時間の測定のために ping、クライアントサーバ間の帯域幅の測定のために iperf を用いて変換ノード上の IPv4/IPv6 変換ソフトウェアの性能を計測した。3 台の端末の詳細は表 4 の通りである。Network Interface の Realtek RTL-8169 は変換ノードの IaaS 側のインターフェイスにのみ使用した。それ以外は全て Intel 82573L を使用した。

iperf では通信プロトコルとして TCP を使用した。比較対象として変換ソフトウェアを用いず変換ノードをルータとして扱い、クライアントと変換ノード間のネットワーク、変換ノードとサーバ間のネットワークを共に IPv4 ネットワークのみにした場合と IPv6 ネットワークのみにした場合でも計測を行った。それぞれの実験の構成を図 16 にまとめた。以降この図に従い map646 を使用した計測を構成 1、IPv4 ネットワークのみを使用した計測を構成 2、IPv6 ネットワークのみを使用した計測を構成 3 と呼ぶ。

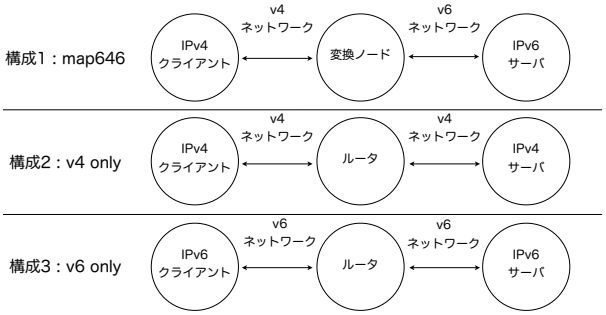

図 16: 実験構成

ping、iperf の結果をそれぞれ図 17、図 18 に示す。計測は全て 5 回ずつ行い図にはその平均値を示した。

図 17 の横軸はそれぞれの構成での結果を示し、縦軸は遅延時間である。構成 1 の遅延時間は 0.172ms、構成 2 は 0.139ms、構成 3 は 0.14ms となった。

図 18 の横軸はそれぞれの構成での結果を示し、縦軸は帯域幅である。構成 1 の帯域幅は 572.4Mbps、構成 2 は 593.4Mbps、構成 3 は 584.2Mbps となった。

7.4.2 類似技術との性能比較

次に類似技術として nat64[8] を用いて、map646 との性能比較を行った。nat64 は IPv6 ネットワーク内にあるクライアントが変換ノードを通して IPv4 ネット

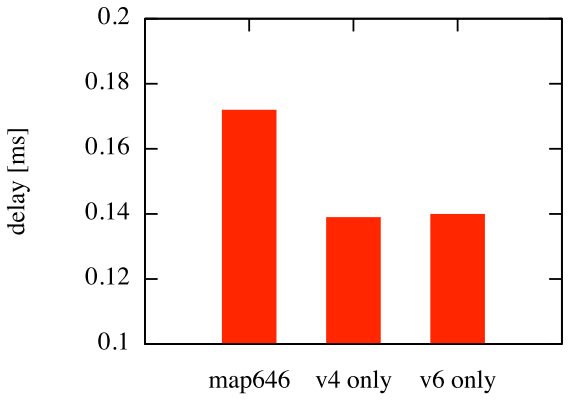

図 17: ping による遅延時間の測定

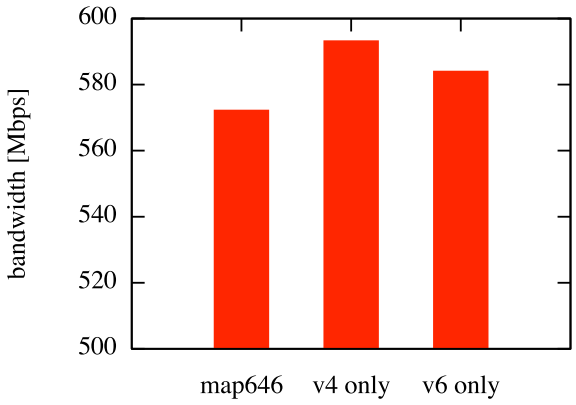

図 18: iperf による帯域幅の測定

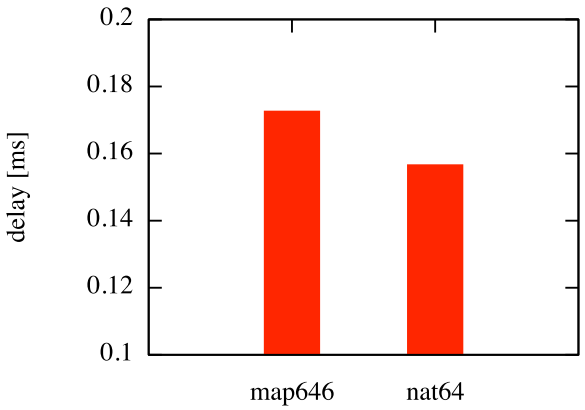

図 19: nat64 との比較: ping による遅延時間の測定

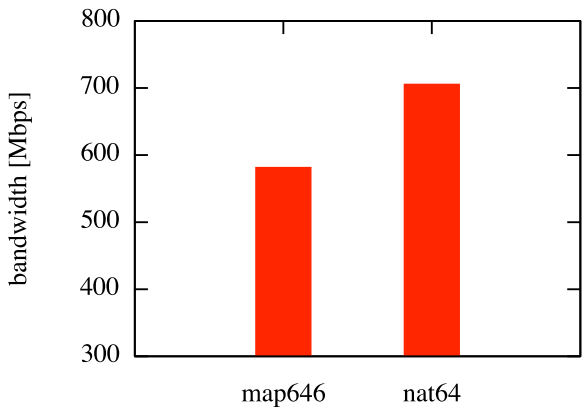

図 20: nat64 との比較: iperf による帯域幅の測定

ワーク内にあるサーバにアクセスするための技術で、クライアントは IPv6 サービスプレフィックスである 64:ff9b::の下位 32 ビットにアクセスしたい IPv4 アドレスを埋め込み変換ノードへパケットを渡す。変換ノードでの IPv6 パケットから IPv4 パケットへの変換は、セッション情報の管理とポート変換によって行われるため、変換に必要な IPv4 アドレスは一つだけで済む。今回の比較実験では nat64 の実装として Ecdysis[11] を使用した。Ecdysis は nat64 のオープンソースの実装で、Fedora をベースにパッケージ化された LiveCD が配布されている。Fedora 内で nat64 はカーネルモジュールとして組み込まれている。

nat64 での想定利用環境に合わせ、IaaS システム内を IPv4 ネットワーク、IaaS システムの外を IPv6 ネットワークとして 7.4.1 章で行ったものと同様の計測を行った。ping、iperf の結果をそれぞれ図 19、図 20 に示す。

図 19 の横軸は map646 での変換を行った場合、nat64 を使用した場合の結果を示し、縦軸は遅延時間である。map646 を使用した際の遅延時間は 0.172ms、nat64 を使用した場合は 0.156ms となった。

図 20 の横軸は map646 での変換を行った場合、nat64 を使用した場合の結果を示し、縦軸は帯域幅である。map646 を使用した際の帯域幅は 582.4Mbps、nat64 を使用した場合は 706.4Mbps となった。

7.4.3 実ネットワークでの性能計測

最後に 7.4.1 章で行ったものと同様の計測を WIDE プロジェクトが運用する WIDE Cloud 内で行った。比

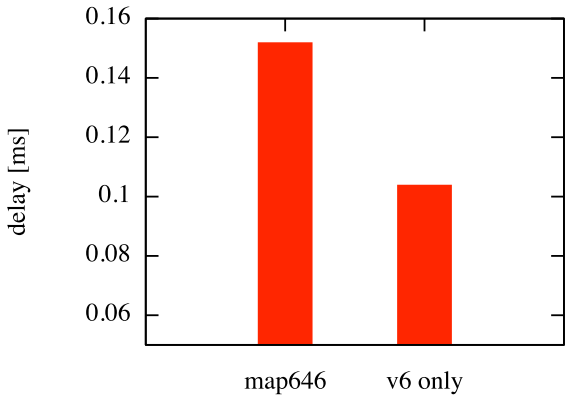

図 21: 実ネットワークでの性能計測: ping による遅延時間の測定

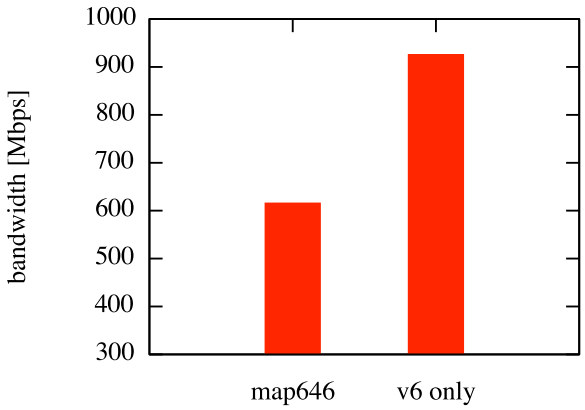

図 22: 実ネットワークでの性能計測: iperf による帯域幅の測定

表 5: 実ネットワーク環境: 使用計算機

CPU	Xeon 2.33GHz * 2
Memory	4GB
OS	Linux 2.6.26
Network Interface	Broadcom NetXtreme II

較対象として変換ソフトウェアを用いず IPv6 ネットワークのみを利用した場合での計測も行った。WIDE Cloud は実運用されている IaaS であるため、アドレスの付け替えが必要になる IPv4 での検証は行えなかった。ping と iperf の結果をそれぞれ図 21、図 22 に示す。図 21 の横軸は map646 での変換を行った場合、IPv6 ネットワークのみの場合を示し、縦軸は遅延時間である。map646 を使用した際の遅延時間は 0.152ms、IPv6 ネットワークのみを利用した場合は 0.104ms となった。図 22 の横軸は map646 での変換を行った場合、IPv6 ネットワークのみの場合を示し、縦軸は帯域幅である。map646 を使用した際の帯域幅は 617Mbps、IPv6 ネットワークのみを利用した場合は 927Mbps となった。WIDE Cloud 内で変換ソフトウェアが稼動しているサーバのスペックを表 5 に示す。

7.4.4 考察

7.4.1 節と 7.4.3 節の遅延時間の計測より、map646 を使用する際の時間コストは 0.03-0.05ms とわかった。変換ソフトウェアではコンフィギュレーションファイルから変換ハッシュテーブルを作成し、tun インターフェイスに到着したパケットの宛先アドレスをキーとして変換ハッシュテーブルから変換先のアドレスを取得するように実装してある。今回コンフィギュレーションファイルに記述してある変換対は実験環境では実験に用いた 1 組、実ネットワークとして利用した WIDE Cloud では約 30 組であったためハッシュテーブルの検索には大した時間はかからず、得られた時間コストの大半は IP ヘッダの変換に費やされていると考えられる。ハッシュテーブルに格納する変換対が増えたときの時間コストへの影響の調査は今後の課題とする。

次に 7.4.1 節の実験では map646 を使用した際の帯域幅の減少は 4%に留まっているが、実ネットワークを用いた 7.4.3 節では帯域幅の減少が 34%にまで広がってしまっている。7.4.1 節と 7.4.3 節での IPv6 ネットワークのみを使用したときの結果を示す図 18 と図 22 を比較すると、7.4.3 節では 7.4.1 節の約 1.6 倍の帯域幅が得られている。この差は各環境で使用した PC の NIC の性能差と考えられ、7.4.1 節で map646 を使用した際の帯域幅の減少が小さかったのは NIC の性能が帯域幅のボトルネックとなっていたからと考えられる。

最後に 7.4.2 節の計測により、類似技術である nat64 に比べ map646 は帯域幅では 18%、遅延時間では 0.016ms パフォーマンスが劣ることがわかった。これは map646 がユーザ空間で動くプロセスとして実装されているのに対し、今回使用した nat64 は Linux のカーネルモジュールとして実装されてためと考えられる。

7.5 結論

仮想化技術の進歩により、より多くのサーバーを運用されるようになりつつある。今後 IPv4/IPv6 のデュアルスタックインターネットが普及していくことが予想され、サーバー側も両プロトコルに対応する必要があるものの、運用コストを考えると通信プロトコルはひとつに集約できた方が望ましい。本提案では、サービスバックエンドの構築を IPv6 のみのシングルスタックで実現し、利用者に公開するポイントのみを IPv4/IPv6 変換機構を用いてデュアルスタック化する IaaS システムを提案し、そのために必要な IPv4/IPv6 変換ソフトウェアの設計、実装、評価を行った。IPv4 アドレスと IPv6 サーバーの対応を一対一に限定することにより、変換ソフトウェアが保持すべき状態をなくし、負荷状況に応じて柔軟に変換ノードを増減でき、またひとつの変換ノードに障害が発生しても、自動的に他の変換ノードがサービスを継続できるスケーラブルかつ耐障害性の高いシステムが実現できた。変換ソフトウェアの実装は tun 仮想ネットワークデバイスを用いてユーザー空間プログラムで実現されている。カーネルモジュールとして実装されている変換ソフトウェアと比較した結果、34%の性能低下が認められたため、今後は高速化を目指す改良を進め、ネットワークの本来の性能を生かしきるような改良を進めていく。

7.6 今後の課題

性能をより向上されるために map646 をカーネルモジュールとして実装していく。またハッシュテーブルに格納する変換対が増えたときのパフォーマンスへの影響を調査していく。また今回は TCP での性能評価しか行わなかったが、UDP を使用した際の評価、ウェブサーバへのアクセスなど実アプリケーションを使用した際の評価も行っていく。さらに変換ノードを通過するパケットの統計的な情報を得たいという IaaS の管理面からの要求に応えるための機能を map646 上に実装し、さらに運用が容易な IaaS システムの実現を目指したい。

8 LISP を用いたネットワークマイグレーション

第 3 章で説明した VPN を用いたサーバ間で接続するコストを抑制するため、NEPL とは異なるアプローチとして LISP(Locator ID Separation Protocol)[2] を用いた VM のネットワーク移動透過性提供手法を展開した。LISP を用いることで各 VM への到達性をホストルートと似た形で 1 対 1 のマッピングをおこない、L2 を拡張する VPN に対する依存性から解放されることを目指す。

8.1 LISP 概要

LISP(Locator ID Separation Protocol) は現状のグローバルインターネットにおける BGP 経路数抑制を目的として、IP アドレスが示す 2 つの識別子である Routing Location(RLoc) と Endpoint ID(EID) を異なる識別子としてとりあつかうプロトコルである。RLoc と EID は通常の IP アドレスによって表現される。LISP は既存のインターネット上に ETR(Egress Tunnel Router)、ITR(Ingress Tunnel Router)、Map Server(MS)、Map Resolver(MR) を追加することによって構成されるミドルボックスソリューションである。LISP では通常 XTR(ETR と ITR の総称) 配下に EID Prefix を展開し、IP アドレスとして EID が割り当てられた End Point が設置される。

8.1.1 LISP のデータプレーン

ITR が End Point の送信したパケットの宛先 IP アドレスをキーとして MR に対して検索をかけ (Map-Request)、MR からの返答である (Map-Reply) によって宛先 EID に対応する RLoc を学習する。ITR は End Point から受け取ったパケットを送信元 IP アドレスを ITR、宛先 IP アドレスを RLoc アドレスとし、LISP パケットとしてカプセル化して送信する。

ETR は受信した LISP パケットのカプセル化を解除し、通常の IP 経路表上でダイレクトコネクトされている収容 EID Prefix のエントリに対してパケットを転送する。なお、一度検索された EID に対応する RLoc には TTL がセットされており、パケットカプセル化とパケット転送の対応表にキャッシュされる。また、ITR 上

で Map-Request をおこなった結果 EID に対する RLoc が存在しない場合、LISP 以外のネットワーク (通常はグローバルインターネット) との通信と定義し、ITR は通常の IP 経路制御によって構築された経路表を参照してパケット転送をおこなう。グローバルインターネットから LISP サイトへの通信は、PXTR(Proxy XTR) が通常の IP 経路制御によって EID Prefix を広告することでトラフィックを引き寄せ、PXTR を ITR とした LISP 通信を EID に対しておこなう。

8.1.2 LISP のコントロールプレーン

XTR は収容している EID Prefix と ETR 自身が持つ RLoc を Map 情報として MS に対して登録する (Map-Regist)。MS は BGP によって Map-Regist された EID と RLoc のマッピング情報を他の MS と MR に対して広告する。ITR は End Point から受信したパケットの宛先 IP アドレスに基づいて MR に Map-Request をおこない、MR は BGP により通知された EID を持つ MS に対して Map-Request を転送する。この BGP によって構築されたネットワークを ALT(Alternative LISP Topology) と呼ぶ。Map-Request を受信した MS は、Map-Request を該当する ETR に対して転送し、ETR は検索をおこなった ITR に対して Map 情報を Map-Reply として返答することで、ITR は EID に対する RLoc を学習する。

8.2 分散クラウド環境における LISP の位置づけ

本研究で提案する分散クラウド環境における要素を LISP に対応させた場合、VM は EID を IP アドレスとして割り当てられる End Point である。また、VM のデフォルトルータは XTR となり、IaaS 事業者の Local Network から XTR に割り当てられた IP アドレスが RLoc となる。本研究で提案する分散クラウド環境では、VM がライブマイグレーションによって異なる拠点へ移動する。また、VM 単体にネットワークの移動透過性を確保する技術を要求しないため、全体のアーキテクチャとしてネットワークによる移動支援型のレイヤ 3 モビリティ技術であると言える。

8.3 分散クラウド環境における LISP の問題点

LISP は通信の種別として大きく分けて、1)LISP サイトから LISP サイトへの通信、2) グローバルインターネットから LISP サイトへの通信、3)LISP サイトからグローバルインターネットへの通信の 3 種類がある。この中で EID を持つ End Point が移動することで問題になるのが、LISP サイトから LISP サイトへの通信である。LISP サイトから LISP サイトへ通信をする際、送信元 End Point を収容する ITR は Map-Request によって通信先の LISP サイトの EID を収容する ETR が持つ RLoc を検索し、検索結果を Map Cache として保持しパケット転送テーブルにキャッシュする。そのため、通信相手である End Point が移動した場合、Map 情報に対する TTL がタイムアウトするまでキャッシュ不整合がおこる問題がある。現在 IETF で議論されている LISP の Internet Draft では EID と RLoc の Map 情報の TTL は分単位で規定されているため、キャッシュ不整合によって VM の拠点移動の際に長時間の通信断が発生する。

8.4 LISP を用いた分散クラウドアーキテクチャの提案

本研究では分散クラウド環境において、VM のライブマイグレーションによる Map 情報のキャッシュ不整合を解決するアーキテクチャを提案する。この問題を解決するため、本研究では CP(Content Provider) が管理するネットワークにおいて、LISP ネットワークとグローバルインターネットからのトラフィック着信に対するアンカーとして代表 ETR を定義し、CP の LISP ネットワーク内に限定した VM の EID と RLoc を管理する Map Server を定義した。

図 23 に代表 ETR(図中 D-ETR) を用いたネットワークアーキテクチャと VM マイグレーション時のプロシージャを示す。代表 ETR は LISP ネットワークとグローバルインターネットに対し CP が委譲するネットワーク Prefix を集約して広告し、LISP サイトとインターネットからのトラフィックを代表 ETR に引き寄せる。そのため、LISP ネットワークからは CP に割り当てられた EID Prefix は集約経路のみが存在する。これにより他の LISP サイトの XTR が CP の持つ EID に対

する RLoc を解決した場合、必ず代表 ETR の IP アドレスを通知されるため、実際に VM が収容されている XTR の RLoc を解決する必要が無く、キャッシュの不整合を回避する。

VM のマイグレーションは Local MS に対して Cloud Controller から通知され、代表 ETR は各 VM のホスト単位で記述された EID と RLoc の Map 情報を持つ Local MS と BGP peer を確立し、VM の移動に追従した最新の Map 情報を BGP Update によってプッシュ型通知される。その手順は以下の通りである。

1. VM のライブマイグレーションは IaaS 事業者が VM を管理・運用する Cloud Controller が VMM に対してシグナリングによって開始される。
2. VM は拠点 A から拠点 B へライブマイグレーションされる。
3. Cloud Controller は VM のライブマイグレーションと連動して MS に Map-Regist によって通知をおこない、MS に対して常に VM の EID と RLoc の最新 Map 情報を提供する。
4. MS は代表 ETR に対して BGP peer を確立し、Cloud Controller から受け取った Map-Regist 情報を BGP Update によってプッシュ型通知をおこなう。

これらの手順を経て、LISP ネットワークもしくはグローバルインターネットより代表 ETR に配送されたトラフィックは、代表 ETR 上にある最新の Map 情報を基に EID に対応する RLoc に対してパケットを LISP パケットとしてカプセル化して適切な XTR へと転送される。

8.5 現在の利用状況と今後の課題

現在 WIDE プロジェクトでは LISP Beta Network[12] と呼ばれる LISP の普及を目的としたプロジェクトに参加し、LISP Beta Network の仮想 AS(AS3943) が提供するアドレスブロックより IPv4 と IPv6 の Prefix 委譲を受け、WIDE クラウド上の LISP 実験網に展開している。現在 WIDE クラウドではこれらの Prefix から切り出された IP アドレスを利用して一部の VM が稼働している。また、VM のマイグレーションに伴う EID-RLoc Mapping 情報の変更と BGP によるプッシュ型通知を利用した実験をおこなっている。

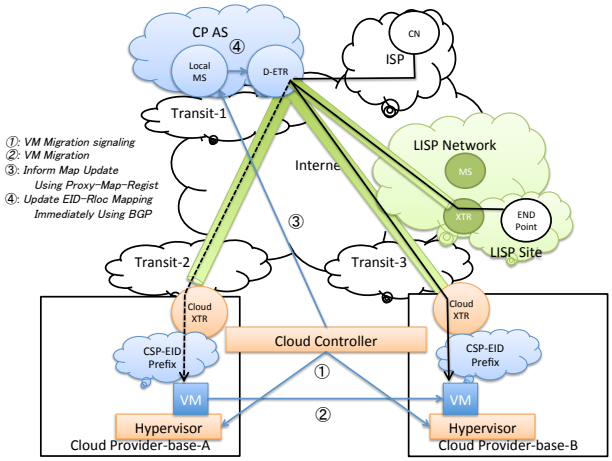

図 23: LISP を用いた分散クラウド環境のアーキテクチャ

また、本研究グループでは Linux 上における XTR と MS の実装がほぼ完了させた。そのため今後の課題としてはこれらの実装を WIDE クラウド上の各ハイパーバイザー上に展開し、WCC との連携インターフェースを持つことで、WIDE クラウドの全利用者に対して適用可能な環境を整えていく予定である。

9 IETF の活動

WIDE クラウドワーキンググループでは、広域連邦型クラウド運用におけるネットワーク資源の透過的な利用を実現するための要求事項とその実装例を検討している。検討した内容はインターネットドラフトとしてまとめられ [13]、IETF に提出されている。

[13] では、ネットワーク資源の透過的な運用に必要な要求事項として以下の点に注目した。

- 異なるデータセンタであっても、同一のユーザには同一のネットワーク資源を提供すること
- ネットワーク資源が、あるデータセンタから別のデータセンタへ移動する際には統一した移動ポリシが適用できること

- ユーザに提供されるネットワーク資源は、サービス中断を避けるため、高可用性を有すること

また、ネットワーク資源のデータセンタ間移動を実現する手法として、以下のふたつの移動モデルを提案した。

- ホストベース移動モデル
- ネットワークベース移動モデル

ホストベースの移動モデルでは、ユーザが利用している仮想計算機自身がネットワーク移動のための機能を持ち、ネットワークベースの移動モデルではデータセンタネットワークがその機能を提供する。後者の場合、ユーザの仮想計算機はネットワーク資源の移動に関与する必要がない。前者を実現する例としては、IP 層の経路制御を用いた方法、Mobile IP[14, 15] などの移動通信プロトコルを仮想計算機が有する方法などが挙げられる。後者を実現する方法は、広域 VLAN、vxlan[16] などを用いたデータリンク層での解決方法、NEMO[1]、LISP[2] などを用いた IP 層での移動通信技術などが挙げられる。

データセンタにおける資源透過性の話題は、近年の IETF でも盛んに議論されている。IETF ワーキンググループとしては、Layer 2 Virtual Private Networks、Layer 3 Virtual Private Networks、Operations and Management Area Working Group などに跨がる話題となっており、また台湾の IETF82 以後、データセンタ技術に関係する話題を取り扱う `dc@ietf.org` メーリングリストも設置された。

WIDE クラウドワーキンググループでも、引き続き連邦型クラウド技術における資源透過性の議論を継続していく。

10 まとめ

WIDE クラウドワーキンググループでは、広域に広がる組織を前提とした、連邦型クラウドサービスの運用技術を研究開発している。2010 年度に引き続き、WIDE クラウド管理ソフトウェアの開発、WIDE クラウドネットワークの拡張を継続している。WIDE クラウドサービスの応用として、夏の甲子園中継の映像配信サービスのサービスホストとして、仮想計算機を用いた動的な負荷適応の実証実験を実施した。また、広域での仮想計算機マイグレーション技術として、ストレージのマイグレーション技術の実証、分散型ストレージ技術の検証を実施した。広域マイグレーションでは、ストレージに加えてネットワークの透過性も必要であり、こちらに関しては LISP 等の IP モビリティを利用した透過的なネットワーク資源提供方法の研究、IPv4-IPv6 ヘッダトランスレータを用いたアドレス変換によるネットワーク透過性の提供技術の実証をおこなった。近年は IETF でも仮想計算機の管理に関する話題が注目されており、WIDE クラウドワーキンググループでも、実験環境での運用経験を踏まえた要求事項を提案している段階である。今後も、規模性にすぐれた広域分散環境でのクラウド運用を可能とする技術の研究開発を継続していく。

参考文献

[1] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, and Pascal Thubert. *Network Mobility (NEMO) Basic Support Protocol*. IETF, January 2005. RFC3963.

[2] Dino Farinacci, Vince Fuller, Dave Meyer, and Darrel Lewis. *Locator/ID Separation Protocol (LISP)*. IETF, December 2011. draft-ietf-lisp-17.

[3] Kazutaka Morita. Sheepdog, December 2011. http://www.osrg.net/sheepdog/.

[4] National Information of Information and Communications Technology. Hokuriku Research Center (StarBED), December 2012. http://www.starbed.org/.

[5] Hiroshi Kitamura. *SOCKS-based IPv6/IPv4 Gateway Mechanism*. IETF, April 2001. RFC3089.

[6] Jun ichiro itojun Hagino and Kazu Yamamoto. *An IPv6-to-IPv4 Transport Relay Translator*. IETF, June 2001. RFC3142.

[7] Marcelo Bagnulo, Andrew Sullivan, Philip Matthews, and Iljitsch van Beijnum. *DNS64:*

DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. IETF, April 2011. RFC6147.

[8] Marcelo Bagnulo, Philip Matthews, and Iljitsch van Beijnum. *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*. IETF, April 2011. RFC6146.

[9] WIDE project. WIDE Cloud Controller, August 2011. http://wcc.wide.ad.jp/.

[10] Keiichi Shima and Wataru Ishida. map646: Mapping between IPv6 and IPv4 and vice versa, August 2011. https://github.com/keiichishima/map646/.

[11] Viagénie. Ecdysis: open-source implementation of a NAT64 gateway, August 2011. http://ecdysis.viagenie.ca/.

[12] LISP Beta Network Project. LISP Beta Network Project. http://www.lisp4.net/beta-network.

[13] Keiichi Shima, Yuji Sekiya, and Katsuhiro Horiba. *Network Portability Requirements and Models for Cloud Environment*. IETF, October 2011. draft-shima-clouds-net-portability-reqs-and-models-01.

[14] Basavaraj Patil, Phil Roberts, and Charles E. Perkins. *IP Mobility Support for IPv4*. IETF, August 2002. RFC3344.

[15] David B. Johnson, Charles E. Perkins, and Jari Arkko. *Mobility Support in IPv6*. IETF, June 2004. RFC3775 (Obsoleted by RFC6275).

[16] Mallik Mahalingam, Dinesh G. Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. *VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. IETF, August 2011. draft-mahalingam-dutt-dcops-vxlan-00.