WIDE Technical-Report in 2011

WIDEクラウドWG 2010年度活動報告

wide-tr-cloud-report-2010-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on WIDE documents, please contact to board@wide.ad.jp

Title: WIDE クラウド WG 2010 年度活動報告
Author(s): 石橋 尚武, 岡本 慶大, 島 慶一, 関谷 勇司
Date: 2011-1-4

WIDE クラウド WG 2010 年度活動報告

石橋 尚武* 岡本 慶大† 島 慶一‡ 関谷 勇司§

2011 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。活動内容としては、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 Cloud システムの運用と、それを用いた研究開発である。

今年度の主な活動を以下に列挙する。

- クラウドアーキテクチャシンポジウム開催 (2 章)
- 甲子園インターネット中継 (3 章)
- ライブマイグレーション性能評価 (4 章)
- ネットワーク可搬性 (5、6 章)
- NAT64 の実装と運用 (7 章)

以降、それぞれの活動の詳細を各章にて報告する。

2 クラウドアーキテクチャシンポジウム開催

2010 年 7 月 2 日に、慶應義塾大学三田キャンパスにて WIDE プロジェクト主催のクラウドアーキテクチャシンポジウムを開催した。本節では、その開催報告を行う。

2.1 開催目的

本クラウドワークショップは、「クラウド」というキーワードが氾濫する昨今の状況において、そのアーキテクチャを技術的にとらえることで、従来のサービスと何が異なり、何が技術的な問題点となるのかを探り、クラウドに関する理解を深めるために開催された。

WIDE プロジェクトが主催するため、クラウドという言葉が意味する社会的な側面ではなく、技術的な観点から、その利点と欠点、ならびに運用上の問題点と改善すべき課題に関して、発表者から提言をしてもらい、それを議論する形式にて行われた。

技術課題の解決に関しては、WIDE プロジェクトが今後果たすべき役割、ならびに標準化等の貢献に関して、どのように進めていくべきかの議論を中心としてまとめられた。

本シンポジウムは、WIDE プロジェクトのスポンサー企業ならびに WIDE メンバーに対して参加の案内が送付され、当初予定通りの 50 名以上が参加する盛況なシンポジウムとなった。

2.2 プログラム

本シンポジウムのプログラムを、以下に示す。

- WIDE プロジェクト代表挨拶 (代表 江崎浩 (東京大学))
- 第一部 : クラウドコンピューティングのアーキテクチャ (13:00 – 14:45)
  - クラウドコンピューティングのアーキテクチャ(村井純 (慶應義塾大学))
  - クラウドのセキュリティアーキテクチャ (門林雄基 (奈良先端科学技術大学院大学))
  - IaaS アーキテクチャの現状 (関谷勇司 (東京大学))
- 第二部 : テクニカル・アップデート (15:00 – 16:45)

*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
§東京大学

  - セキュリティとインタークラウド (門林雄基 (奈良先端科学技術大学院大学))
  - WIDE クラウドワーキンググループの活動 (関谷勇司 (東京大学))
  - StarBED とクラウドコンピューティング (三輪信介 (独立行政法人情報通信研究機構))

- 第三部：クラウド技術開発をガイドする (17:00 – 19:00)
  - パネル：技術とベストプラクティス (座長: 江崎浩 (東京大学))
  - パネル：ポリシと標準化 (座長:村井純 (慶應義塾大学))

2.3 シンポジウムにおける議論

第一部は、クラウドコンピューティングのアーキテクチャに関しての発表と議論が行われた。一口に「クラウド」と言っても、その運用形態は様々であり、その形態に応じてアーキテクチャも異なることが報告された。また、クラウドコンピューティングを利用したサービスと、既存のサービスとの違いについて発表があり、その違いから発生するセキュリティ上の問題点に関する発表も行われた。

発表後の議論は、アーキテクチャの開放性、透明性、イノベーションに関するものから始まり、いまのクラウドはクローズドアーキテクチャの抱え込みサービスになっている点が懸念である、という点が指摘された。また、クラウドの利点を生かして運用されている商用クラウドというものが少なく、既存技術の組み合わせで構築されたアーキテクチャの上で、既存のサービスの低コスト版を展開しているのみにすぎない、という意見が出された。

今後は、SLA(Service Level Agreement)、ならびに信頼性、冗長性の確保が技術的な課題となり、オープンアーキテクチャを基にしたクラウドアーキテクチャの設計が望まれるという議論がなされた。

第二部では、より技術的な観点からの現状報告とその問題点に関する議論が行われた。まず、インタークラウドを構築するにあたってのセキュリティ課題に関する発表が行われ、クラウドのセキュリティというものが認識されておらず、今後問題となるであろう課題に関する報告が行われた。次に、WIDE プロジェクトで構築、運用している WIDE Cloud の技術詳細に関する発表が行われた。最後に、NICT が構築、運用している StarBED をクラウドとして利用するための実験や検証に関する報告が行われた。

発表後の議論においては、より積極的な広域分散と耐障害性を追求した技術開発が必要であるという点と、オープンな技術でかつセキュリティを確保したインタークラウドを実現するための手法が課題であるとの指摘があった。また、クラウドの管理技術に関する議論も行われ、現状の技術では信頼性を担保するための監視技術や管理技術が確立されていないことも議論された。

第三部においては、パネルセッションにて議論が行われた。このパネルセッションでは、クラウドアーキテクチャはより上位層のアプリケーション、サービスを含めたものに発展していく傾向にあり、その組み立て方によっては、「砂上の楼閣」となってしまうという懸念が指摘された。これは、基盤の技術を固めずに既存の技術を継ぎ接ぎして構築されているクラウドアーキテクチャを比喩するものである。これに関して、オープンアーキテクチャの技術を提唱し、きちんと標準化を行っていくことが重要である、という指摘が出され、WIDE プロジェクトは技術開発と標準化を通して世界に貢献することが必要であるとの認識が共有された。

2.4 シンポジウムのまとめ

本シンポジウムは三部構成で行われ、第一部、第二部では「クラウド」の現状把握とその技術詳細、問題点の把握が行われた。これを受けて第三部では、クラウドの健全な発展とそのために WIDE プロジェクトが今後なすべき課題に関する議論が行われた。WIDE プロジェクトはその技術開発と標準化、運用技術の蓄積において、クラウドのガラパゴス化を防ぎ、オープンアーキテクチャとしての健全なる発展に貢献することが必要である、との結論が導かれた。

この議論をうけ、WIDE クラウド WG では、信頼性、冗長性の確保、ならびに安定運用のための技術開発や標準化に積極的に取り組む所存である。

3　甲子園インターネット中継

3.1　プロジェクト概要

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。　今年度は、Web コンテンツの一部を WIDE Cloud を利用して配信を行うことで、期間限定の大規模配信におけるクラウドコンピューティング活用について検証した。

3.2　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 の入力から静止画の切出しと圧縮を行い sentryc へ提供

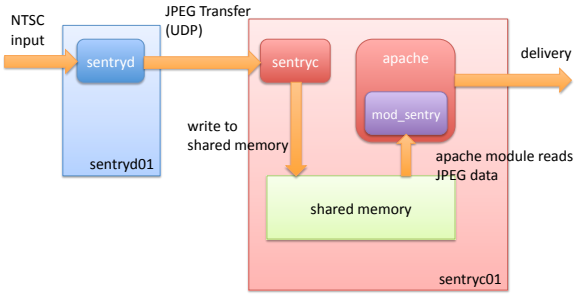

図 1: sentry サーバプログラムの動作

sentryd と sentryc の動作モデルを図 1 に示す。sentryd は入力された映像をパラパラ漫画の要領で 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ようになっている。

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といった環境下でも、試合進行や球場の雰囲気を伝えることができる。

sentryd から sentryc へのデータ転送は、プライベートネットワークを利用しており、sentryd はインターネットへの接続点は持っていない。

3.3　昨年度までの課題とクラウドによる解決

昨年度までの構成では、アクセス集中時に sentryc 系のサーバ能力不足や対外線の逼迫により、一部のユーザへ画像が配信されないような場合があった。sentryd 側で JPEG 画像の圧縮率を上げる、パラパラアニメの更新間隔を伸ばすなどの対策を行ったが、その分ユーザへの配信クオリティが落ちてしまうという問題もあった。そこで、sentryc 系サーバの一部を WIDE Cloud 上の VM とすることで、外部の計算機リソースやネットワーク帯域を利用するという運用を行った。

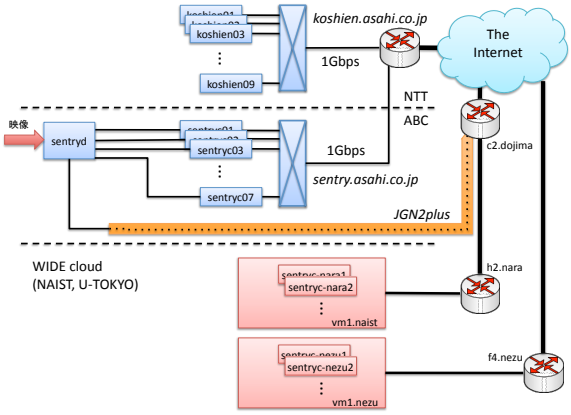

図 2: 2010 年度甲子園システムネットワーク概要図

今年度の甲子園システムのネットワーク構成を図 2 に示す。WIDE Cloud 上の VM は、奈良 (NAIST) と根津 (東大) に配置し、1VM 毎に 1 グローバル IPv4/v6 アドレスを割り当てた。

従来プライベートネットワークで運用していた sentryd は、cisco2.dojima との間を JGN2plus 経由で接

続することで、インターネット経由で WIDE Cloud 上の sentryc へも画像転送を可能とした。

実計算機である sentryc.asahi.co.jp と、WIDE Cloud の振り分けは、koshien が配信する JavaScript の参照先サーバを書き換えることで実現した。koshien シリーズは本年度は 9 台で運用したので全体の 1/9 ずつのトラフィックを振り分けられる。さらに、koshien シリーズの負荷に余裕がある場合は前段のロードバランサのウェイトを変更することで細かいトラフィック調整が可能である。

3.4 ベンチマーク

実際の運用前に、WIDE Cloud 上の VM での Apache ベンチマークを行った。Hypervisor は CPU による変化を調査するため、Intel Xeon と AMD Opteron のサーバを利用した。Hypervisor の構成を表 1 に示す。

また、VM の GuestOS として これまでの甲子園での運用実績のある CentOS 5.5 と、Hypervisor と同じ OS である Ubuntu 10.04 をインストールし、OS の違いによる性能差を比較した。また、仮想 NIC について、準仮想化ドライバを利用する virtio と、Intel Pro 1000 のエミュレーションである e1000 の比較も行った。VCPU は 1、割り当てたメモリは 4GB、利用した Apache のバージョンは 2.2.16 で、MPM は event を利用し、mod_sentry を組み込んである。

ベンチマークは httperf を利用し、mod_sentry によって画像が提供される URL を対象とした。sentryd の提供するコンテンツサイズは 25KB とした。

表 2: Apache ベンチマーク結果

構成	レスポンスレート (s)
Intel+CentOS+virtio	2153
AMD+CentOS+virtio	1107
Intel+CentOS+e1000	1188
Intel+Ubuntu+virtio	1886

表 2 にベンチマーク結果を示す。最も性能が高かったのは、Intel Xeon のサーバ上で CentOS と virtio を利用した組み合わせであった。AMD のパフォーマンスが伸びないのは、kvm_amd カーネルモジュールの実装の問題である可能性が高い。virtio が e1000 より速いのは、準仮想化によるエミュレーションのオーバーヘッド削減の効果によるものである。

Linux カーネルのバージョンが古いにもかかわらず CentOS (Linux 2.6.18) が Ubuntu (Linux 2.6.32) より速い理由は、タイマ割り込み間隔などのカーネルの設定に起因すると考えられる。

3.5 運用報告

甲子園システムの運用は、開幕の 8 月 7 日から行われた。WIDE Cloud では、奈良と根津にそれぞれ 4 台ずつの VM を準備した。決勝戦でのトラフィックは甲子園システム全体で 650Mbps を記録した。また、最大トラフィックは準決勝で 800Mbps を記録している。

3.5.1 実戦投入

WIDE Cloud 上の VM の実戦投入は甲子園 2 日目第 3 試合から開始され、koshien のうち 1 台の JavaScript の参照先を奈良の VM へ変更することでトラフィックが分散されることを確認した。VM の TCP セッション数の変化を図 3 に示す。14:30 付近からセッション数が増加しているのがわかる。

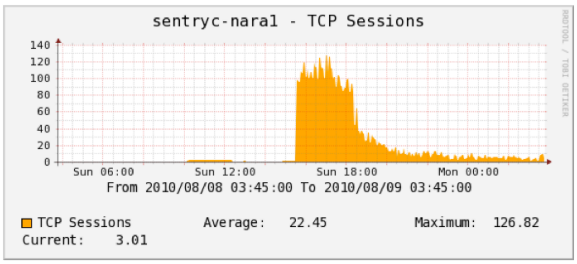

図 3: sentryc 切替時の TCP セッション数の変化

随時、VM の追加、削除を行ったが特に問題は無く、大会 8 日目は準備した 8VM を全て投入した。この日の奈良の Hypervisor におけるネットワークトラフィックを図 4 に示す。4VM の合計となっており、最大 80Mbps、根津と合わせて約 150Mbps、約 8000TCP セッションとなった。

3.5.2 リバースプロキシ導入

koshien と sentryc の紐付けにおけるさらに細かいウェイト調整や、AMD の Hypervisor の利用、グロー

表 1: Hypervisor の構成

	vm1.naist	vm2.naist
CPU	Xeon L5520 (2.26GHz) x2	Opteron 2387 (2.8GHz) x2
Memory	36GB DDR3	64GB DDR2
OS	Ubuntu 10.04 amd64	
Hypervisor	KVM 84	

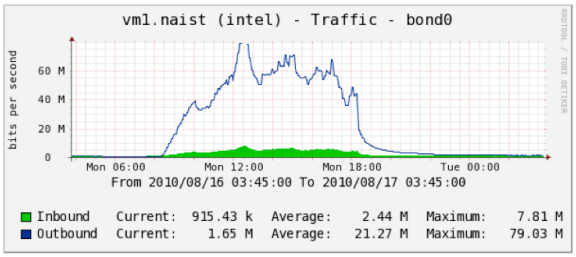

図 4: 4VM 合計ネットワークトラフィック

バル IPv6 アドレスのみを持つ VM の追加などを目的に、リバースプロキシを導入した。リバースプロキシがボトルネックにならないよう、実計算機を利用した。表 3 にリバースプロキシのサーバ構成を示す。

表 3: リバースプロキシのサーバ構成

CPU	Xeon X5570 (2.8GHz) x2
Memory	24GB DDR3
Network	10 Gigabit
OS	CentOS 5.5
Rev.Proxy	Apache 2.2.16 (event MPM) mod_proxy_balancer

mod_proxy_balancer のセッティングを図 5 に示す。ここに示しているのは、guest1 および guest2 を Intel Xeon の Hypervisor 上、guest3 および guest4 を AMD Opteron の Hypervisor 上に配置した状態で、負荷を均等にかけたい場合の設定である。

大会 14 日目第 1 試合において、トラブルに見舞われたものの、リバースプロキシのみで 166Mbps、甲子園システム全体で約 800Mbps を記録した。

3.5.3 トラブル

全 VM 投入後、大会 11 日目のピーク時間帯に VM や Hypervisor の応答が著しく悪化する問題が発生した。問題発生時の Hypervisor の CPU 使用率のグラフを図 fig:cloud:koshien-sentryc-trouble に示す。30 分程の間、監視マシンから snmp データが取得できていない様子がわかる。CPU 使用率やメモリ使用量、ネットワークトラフィックはまだ頭打ちになっていないので、各 VM から発行されるネットワーク割り込みの処理によって Hypervisor の処理が逼迫し、パケットに応答できなくなった可能性が考えられる。

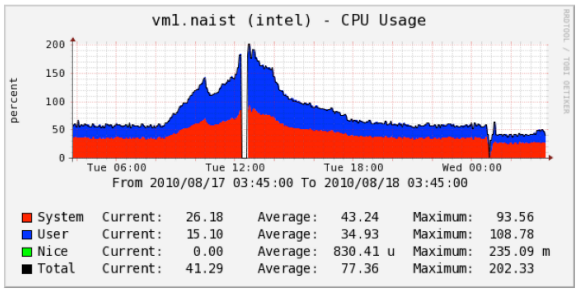

図 6: VM 応答悪化時の Hypervisor の CPU 使用率

さらにリバースプロキシ導入後にも応答悪化、画像が途中で切れるなどの問題が発生した。こちらは、奈良に設置したリバースプロキシのバックエンドに根津の VM を配置したのが原因で、バックエンドを奈良の VM のみにしたところ解消した。

また、Ubuntu 10.04 の net-snmp に問題があり、80Mbps 以上のトラフィックを snmp で取得できないという問題や、Internet Explorer において sentryc の切替がうまくいかない問題などがあることも分かった。

図 5: mod_proxy_balancer の構成ファイル (例)

```
<Proxy balancer://cluster>
    BalancerMember http://guest1.naist.wide.ad.jp/sentry-handler loadfactor=10
    BalancerMember http://guest2.naist.wide.ad.jp/sentry-handler loadfactor=10
    BalancerMember http://guest3.naist.wide.ad.jp/sentry-handler loadfactor=2
    BalancerMember http://guest4.naist.wide.ad.jp/sentry-handler loadfactor=2
</Proxy>
```

3.6 考察と課題

今回の実験は、期間限定の大規模配信基盤としてクラウドを利用するという試みであり、おおむね良好な結果を得ることができたと考えている。特に、昨年度は複数回行う必要があったパラパラアニメの更新間隔の調整が今年度は 1 度だけで済んだため、クラウドを利用することでクライアントへの配信クオリティを向上できたと結論づけられる。

運用面では、検証不足な点がいくつか散見された。ベンチマークほど実戦では性能が出ない点、同一ワークロード多重化による性能劣化、リバースプロキシの構成法などである。

今後、これらのトラブルの原因究明や、実戦により近いベンチマーク手法、VM のチューニングについて検討を行う必要がある。

また、クライアントのソースアドレスや AS によって配信元の VM を変更する、つまりクラウドを CDN の一種として利用する方法についても検討を行いたい。

4 ライブマイグレーション性能評価

4.1 背景

Xen や KVM などの仮想化技術のひとつにゲスト OS を停止させずにハイパーバイザ間を移動させる技術であるライブマイグレーションがある。ライブマイグレーションは LAN 内で行われることを想定し設計されている技術である。本計測では、ライブマイグレーションを長距離広帯域下で行った際の挙動を計測し評価を行う。

4.2 予備実験

Xen においてメモリの汚れ方によってライブマイグレーションの挙動がどのように変わるのかを実験するに当たって予備実験を行った。2 台の ThinkPad X61 に Fedora 14 と Xen4.0.1 をインストールし Gigabit Ethernet スイッチに接続し、memtest+86 の CD イメージをブートディスクとする仮想マシンを作成し 128MB のメモリを割り当てた。メモリにシーケンシャルに書き込みを行う Test 0 とランダムに書き込みを行う Test4 をそれぞれ実行しながらライブマイグレーションを行った。

Test0、Test4 におけるメモリ転送の各イテレーションにおける転送されたページと汚れたページとスキップされたページの数をそれぞれ図 4.2(a)(b) に示す。Test0 では早い段階で転送ページ数と汚れたページ数の差が収束し 13 回目でイテレーションを終了しているのに対し、Test4 ではページの転送がページの汚れ方に追いついていないため収束せず、30 回目で強制的に終了されている。それぞれのメモリの転送速度と汚れ速度を図 4.2(c)(d) に示す。ランダム書き込みを行う Test4 ではシーケンシャルな Test0 に比べて平均約 3 倍の速度でページが汚れているのがわかる。シーケンシャルな書き込みであれば同じページに何度も書き込みが行われるのでページの汚れ方は緩やかになる。一方でランダムで書き込む場合はページの一箇所でも書き込みが行われればページは汚れたと判断されるためページの汚れ方は激しくなるからである。

Xen ではメモリの転送速度は 500Mbps に制限されているなど、Xen のメモリ転送のイテレーションに関係する各種の値はソースコードにハードコードされ、経験的な値に基づいていると考えられるため、環境に応じて最適な値を見つけることによってマイグレーションを高速化する余地がある。

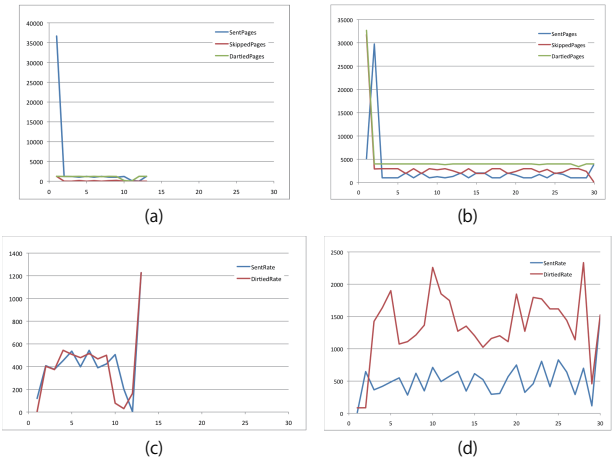

図 7: ライブマイグレーション予備実験の結果

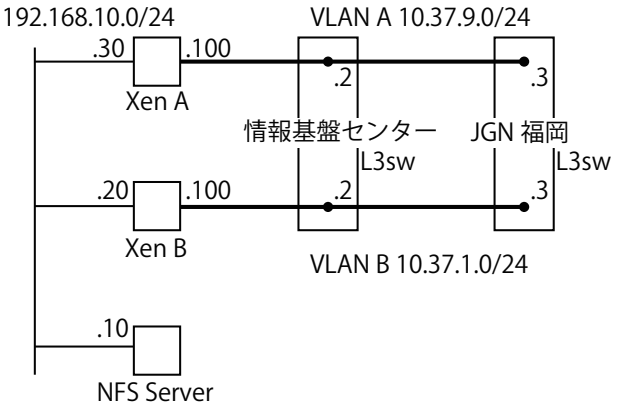

図 8: Xen におけるライブマイグレーション実験時のネットワーク構成

4.3 性能評価

実際の性能評価を行うにあたって、JGN2plus の回線を利用し性能評価を行っている。機器の構成は Dell PowerEdge R410 を 3 台用いた。Xeon 2.27GHz を持つマシン 2 台に Debian lenny と Xen3.2.1 をインストールし、Xenon 1.87GHz のマシンに Ubuntu10.04 をインストールし，NFS サーバとして構築した。ネットワークの構成を図 4.3 に示す。ネットワークは NFS でストレージを共有している LAN はすべて 1G を使用し、JGN2plus の回線では帯域は 10G の回線、Xen のサーバでは 1G の NIC を使用している。福岡での折り返しでは RTT が 40ms、基盤センターでの折り返しでは RTT が 0.2ms である。

実験の手順は下記を予定している。

- Xen A, Xen B 間でライブマイグレーションを行う。
- ライブマイグレーションにかかった時間、イテレーションの回数を計測し、送信パケットを dump する。
- 1 、 2 の 手 順 を ゲ ス ト OS の メ モ リ (128MB,256MB,512MB,1GB,2GB,4GB,8GB) 、折り返し地点（基盤センター、JGN 福岡)、メモリの汚し具合（汚していない状態、自作メモリ汚しプログラム起動時、memtest86+）をすべての組み合わせで繰り返し行う。

現在は JGN2plus の回線を用い数回計測を行い、本計測にむけたチューニングを行っている。予備計測では TCP のウィンドウサイズがボトルネックとなり帯域を十分に使用出来ていなかったため、本計測ではデフォルトのウィンドウサイズを最大値に設定する。

メモリを汚す際のプログラムに関しても、現在は 128MB のファイルを用意し、UNIX システムコールである mmap() を用いてそれをメモリ上に allocate した上でランダムな位置にランダムな値を書き込むプログラムを使っている。ただし、sleep() やランダム値生成時のオーバヘッドが原因で単位時間あたりのメモリの汚れ具合が正確ではないことがわかったため、rdtsc を用いたプログラムに変更行っている。以上のチューニングをもとに今後、本計測を行う。

5 クラウド運用におけるネットワーク可搬性の要求事項

近年の仮想化技術の向上により、様々なインターネットサービスを、いわゆるクラウド環境において提供することが可能になってきた。仮想化技術のひとつに、仮想計算機をハイパーバイザー間で移動させる、マイグレーション機能があるが、この技術は同一リンクに接続されたハイパーバイザー間での移動に限定されており、インターネット上の任意の場所にあるハイパーバイザーへの移動は考えられていない。本章では、インターネットを経由した異なるリンクへの移動の必要

性と、それを実現する運用技術案に言及することで、クラウド運用におけるネットワーク可搬性を考える。なお、本アイデアはインターネットドラフトとしてまとめてあり [1]、IETF Clouds BoF にて議論されたものである。

5.1 背景

本章ではデータセンター間およびクラウドサービス間でのネットワーク可搬性に関する要求と手法をまとめる。仮想化技術の発展は、インターネットサービスに変化をもたらした。PaaS (Platform as a Service) とよばれる技術は、仮想化された計算機によって構成された基盤の上にインターネットサービスを構築する。

現状の PaaS は単一データセンターで単一運営主体によって構築されているが、これを分散し、さらには複数の事業者間で連携させたいという要求は拡大しつつある。こうすることによって、PaaS を構成する一部の機器あるいはデータセンターに障害が発生した場合でも、他のデータセンターへ仮想計算機を移動させる等することにより、サービスの継続可能性が向上するためである。

サービスを停止あるいは影響を与えること無く、仮想計算機をデータセンター間、クラウドサービス間で移動させるためにも、ネットワーク可搬性技術が必要となる。

5.2 ネットワーク可搬性要求

以下に、仮想計算機をデータセンター間やクラウドサービス間で移動させるためのネットワーク可搬性に関する要求事項を挙げる。

- **データセンターやクラウドサービスをまたがるユーザーサービスネットワーク**: 各々のユーザーには、他のユーザーと干渉しない固有のサービスネットワークが提供されなければならない。これはクラウドサービス間をまたがる場合でも守られなければならない。通常、こういったネットワークは VLAN や VPN サービスを用いて実現されるが、特に異なるサービス事業者によって運営されるデータセンター間やクラウドサービス間の連携を考えると現実的ではない。VLAN はレイヤー 2 の技術であり、データセンター間を直結できる回線を持っている必要がある。VPN は VLAN よりも柔軟な構成が可能だが、クラウドサービスの管理主体が異なる場合、容易には相互接続できない。
- **ユーザーサービスネットワークの高可用性**: VLAN や VPN を用いる場合、それがインターネットへの単一障害点にならないようにする必要がある。データセンター間、クラウドサービス間でのネットワーク可搬性を考えるとき、ユーザーサービスネットワークの高可用性を提供できなければならない。

5.3 ネットワーク可搬性モデル

ここでは 2 種類のネッワーク可搬性モデルを考える。ひとつはホストベースのモデル、もうひとつはネットワークベースのモデルである。

5.3.1 ホストベース可搬性モデル

このモデルでは、各ホスト (ゲスト計算機) がネットワーク可搬性の機能を持つ。各々のホストは、なんらかのホスト移動通信プロトコルを備えていなければならない。仮想計算機が、あるハイパーバイザーから、異なるネットワークに属するハイパーバイザーにマイグレートした場合、仮想計算機自身が移動通信プロトコルを発動し、現在接続されているネットワーク環境に応じて適切に自身のネットワーク環境を整えなければならない。

5.3.2 ネットワークベース可搬性モデル

このモデルでは仮想計算機が利用しているネットワーク環境が、移動の前後で変化しないことが保証される。仮想計算機は、ネットワーク環境の変化に気がつかないため、移動の前後でなにかアクションを起こす必要はない。同じネットワーク環境を提供するため、クラウドサービス基盤側は、異なる場所に同じネットワークを提供する機能を持たなければならない。

5.4 実装手法

本節ではネットワーク可搬性を実装する手法をいくつか提案する。5.4.1 節では広域 VLAN を用いた場合の可搬性の実装例を、5.4.2 節ではホストベースの可搬性技術として、Mobile IPv6[2] を用いる例を、5.4.3 節ではネットワークベースの可搬性技術として、NEMO BS[3] を用いる例を紹介する。

5.4.1 広域 VLAN によるネットワーク可搬性

ネットワーク可搬性を実現するひとつの方法として、レイヤー 2 で透過的なリンクをすべてのクラウドサービスに提供する手法が考えられる。近年の高速広域でのネットワークの性能を考えると、広域にレイヤー 2 ネットワークを展開することも現実的となってきている。

この場合、ハイパーバイザーの構成は単純になる。すべてのハイパーバイザーは、広域に展開された同一レイヤー 2 ネットワークに接続されることとなり、単一データセンターにおけるハイパーバイザー運用と変わりがない。仮想計算機は、ハイパーバイザーのブリッジ機能によって、広域レイヤー 2 ネットワークに接続することになる。

仮想計算機のマイグレーションに関して特に注意する必要もない。すべてのハイパーバイザーが同一のリンクに接続しているので、既存のマイグレーション技術を用いることができる。

5.4.2 Mobile IP ベースネットワーク可搬性

ホストベース実装では、すべての仮想計算機が Mobile IP 機能を備えている必要がある。加えて、Mobile IP の運用に必要なホームエージェントもクラウドシステムの一部として運用されなければならない。

各々の仮想計算機はハイパーバイザーが提供するローカルネットワークに接続し、Mobile IP の手順に従って自身の気付けアドレスをホームエージェントに登録する。Mobile IP の具体的な動作については、ここでは言及しないので、詳しくは RFC3775[2] を参照してほしい。

仮想計算機がマイグレートされると、接続先のネットワーク環境がマイグレートした先のハイパーバイザーによって変化する。仮想計算機は、ネットワークの変化を Mobile IP 機能によって検知し、適切に移動処理を行わなければならない。

Mobile IP と同様、本方式を採用する場合はホームエージェントが単一障害点となる。ホームエージェントの多重化などによる高可用性の運用は、本文書の範囲を越えるのでここでは言及しない。

5.4.3 NEMO ベースネットワーク可搬性

NEMO ベースのネットワーク可搬性技術では、RFC3963[3] を用いたルータの移動通信プロトコルを活用する。クラウド運用者は、RFC3963 に準拠した移動ルータを複数運用し、それぞれの移動ルータに固定プレフィックス (Mobile Network Prefix, MNP) を割り当てておく。移動ルータは、クラウドサービスのユーザーからは原則隠されており、ユーザーは透過的に固定プレフィックスを利用する。

移動ルータ自身も仮想計算機として実現し、マイグレートの際には、移動ルータとその固定プレフィックスに接続しているユーザーの仮想計算機が同時に同じハイパーバイザーへ移動する。移動した後、移動ルータが接続するハイパーバイザーのネットワークは、以前のネットワークとは異なるが、NEMO BS の手順を踏むことにより、新しいネットワーク環境に対応可能となる。一方、固定プレフィックスに接続しているユーザーの仮想計算機は、移動が発生した事実を知る必要はなく、移動前と同じ環境で継続して運用可能である。

ひとつの固定プレフィックスには、ユーザーの仮想計算機を必要なだけ接続することができる。ただし、仮想計算機のマイグレーションは移動ルータ単位となるため、同じ固定プレフィックスに接続された仮想計算機は、マイグレーションの際には必ず同時に同じ宛先ハイパーバイザーに移動しなければならない。仮想計算機の移動に細かい粒度を持たせたい場合は、ひとつの移動ルータに収容する仮想計算機の数を減らす必要がある。逆に、通常統一セグメントで運用されるようなサービス、たとえば、ウェブフロントエンドとデータベースなどは意図的に同じ移動ルータに収容して効率を上げることができる。

固定プレフィックスを保持しているネットワークは、実際にはハイパーバイザー内部に仮想的に作り出されたネットワークとなる。移動元と移動先のハイパーバイザーでは、移動する仮想計算機 (移動ルータやユー

ザーの仮想計算機) から透過的にアクセスできる名称で仮想ネットワークが提供されている必要がある。

Mobile IP の場合と同様、高可用性を実現するためには、ホームエージェントの問題を解決する必要がある。

6 WIDE Cloud におけるネットワークベース可搬性技術の運用

WIDE Cloud では、運用開始初期から広域 VLAN ベースのネットワーク可搬性を提供してきた。5 章で分類したとおり、VLAN ベースの可搬性では、異なる管理ドメイン間でのネットワーク可搬性を実現しにくいという欠点がある。そこで現在、NEMO を用いたネットワークベースの可搬性技術を試験運用中である。本章では、その運用の概要を解説する。

6.1 トポロジ

図 6.1 に NEMO ベースのネットワーク可搬性を実現した WIDE Cloud のネットワークトポロジを示す。図に示してあるのは、ネットワーク可搬性に関係する部分のみであり、完全な WIDE Cloud のトポロジとは異なることに注意して欲しい。

ホームエージェントは WIDE の根津 NOC に設置してあり、2001:200:0:1c01::/64 に接続している。移動ルータは現時点では 10 台が運用されており、NEMO のホームネットワークとして 2001:200:d00::/64 を用いている。各移動ルータには 2001:200:d00:1::/64 から 2001:200:d00:a::/64 までのプレフィックスを固定プレフィックスとして割り当てている。移動ルータを受け入れることができるハイパーバイザーは、現在根津 NOC に 3 台、小松 NOC に 2 台が稼働中であり、それぞれのプレフィックスは 2001:200:0:1c0a::/64 と 2001:200:0:1400::/64 となっている。

移動ルータはいずれかのハイパーバイザーの中で仮想計算機として動作しており、ハイパーバイザーが提供するブリッジネットワーク経由でインターネットに接続している。根津から小松、あるいはその逆に移動した場合、ハイパーバイザーの提供するネットワーク環境に変化が生じるが、移動ルータの NEMO 機能により、適切にホームエージェントとのトンネルを張り直し、移動ルータが収容している仮想計算機には影響が出ない運用となっている。

7 NAT64 の実装と運用

WIDE Cloud は IPv6 での運用を前提として設計してある。特に、6 章で紹介したネットワーク可搬性は NEMO BS を基にしているため、IPv6 しかサポートされない。しかしながら、現状を考えると、IPv6 のみでの運用だけでは不十分である事は明らかであり、IPv4 への接続性を提供するのは必須である。そこで、ネットワークの運用は IPv6 のみとして運用コストを下げつつ、IPv4 への後方互換性を提供する手段として IPv4 と IPv6 のヘッダ変換サービス (NAT64) を提供している。

7.1 設計と実装

ここで提供する NAT64 サービスは、WIDE Cloud での運用に特化した割り切った設計となっている。最大の関心はスケーラビリティである。WIDE Cloud 内には、将来的には対外的なサービスを担うノードを仮想計算機として運用する予定にしている。そのため、NAT64 変換部分がボトルネックとならないように、負荷に応じて変換サーバを増減させることができる必要がある。これを実現するため、IPv4 と IPv6 の対応は 1 対 1 とし、一般的な NAT64 に見られるような、複数の IPv6 クライアントがひとつの IPv4 アドレスを共有する機能は省いてある。こうすることにより、NAT64 サーバは IPv6 ノードと IPv4 ノードの対応関係を静的に知るだけで済むようになり、変換の状態を保持する必要がなくなる。すべてのヘッダ変換はパケット単位で実行されるため、たとえ一連のデータストリームが複数の NAT64 サーバに分散したとしても、変換結果に不備が生じる事が無い。これにより、負荷に応じて NAT64 サーバを任意の位置に配置して入力トラフィックあるいは出力トラフィックを分散することができる。

本サービスのために開発した NAT64 プログラムは、オープンソースソフトウェアとして公開済み [4] である。

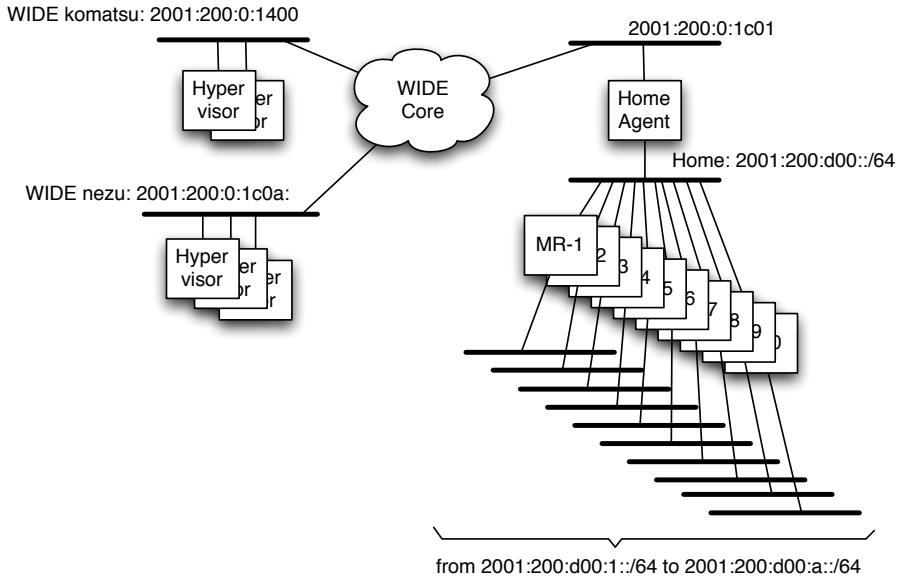

図 9: NEMO ベースのネットワーク可搬性を実現した WIDE Cloud のトポロジ

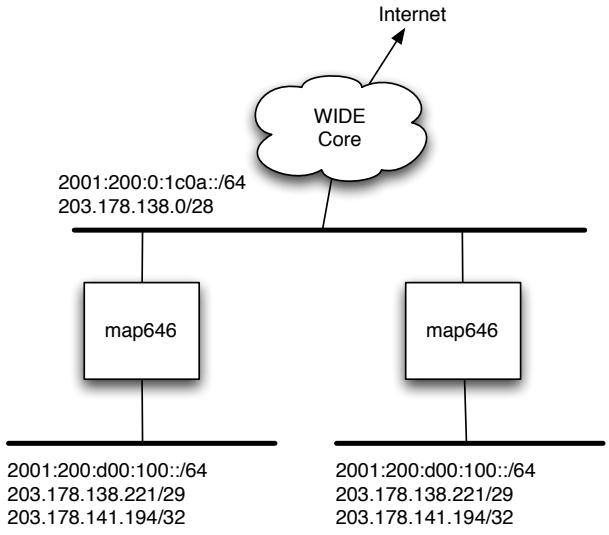

図 10: WIDE Cloud における NAT64 サービスのトポロジ

7.2 トポロジ

図 7.2 に WIDE Cloud における NAT64 サービスのトポロジを示す。現在のところ、2 台の NAT64 サーバが稼働しており、どちらかに障害が発生しても他方がサービスを引き継ぐ形をとることで、冗長性を確保している。

WIDE Cloud 内の IPv6 ノードからは、すべての IPv4 ノードのアドレスが 2001:200:d00:100::/64 の空間にマップされ、あたかも IPv6 ノードと通信しているように見える。

WIDE Cloud の外へは、限られた IPv4 アドレスが公開され、それらは 1 対 1 で WIDE Cloud 内の IPv6 サービスホストに対応づけられる。WIDE Cloud 外の IPv4 ノードは、WIDE Cloud の公開 IPv4 アドレスにアクセスすることで、内部の IPv6 サービスホストのサービスを享受する。

7.3 運用サーバ

現時点で、WIDE Cloud の NAT64 サービスを利用して運用されている公開サーバは以下の通りである。

- www.kame.net
  KAME プロジェクトのホームページ。

- www.internetconference.org
  インターネットコンファレンスのウェブサーバ。インターネットコンファレンス 2010 開催を機に、物理計算機から WIDE Cloud 環境に移行された。

その他、2010 年 10 月 23 日に開催された、厚生労働科学研究成果発表シンポジウムでは、イベント用の一時ウェブサーバとして、4 台の WIDE Cloud サーバが NAT64 サービスと共に用いられ、運用された。

8 まとめ

今年度は WIDE Cloud システムの本格運用が開始され、システム自体の開発はもちろん、それを活用した研究開発や、仮想計算機を貸し出してのイベントサポートなどが行われた。また、研究開発の方向性を議論する場として、WIDE プロジェクトとしてクラウドシンポジウムを開催し、クラウド技術に対する取り組みを紹介するとともに、実際にクラウドサービスに関わっている人たち、また関わってほしいと思っている人たちを巻き込む場を設けた。

イベントとしては、3 章で報告した甲子園中継をはじめ、厚生労働省の厚生労働科学研究成果発表シンポジウム、インターネットコンファレンス 2010 のウェブサーバーホスティング、KAME プロジェクトのウェブサーバーホスティングなどを実施した。

研究活動としては、広域での安定運用を実現するために必要となるライブマイグレーションの性能を評価する活動を実施し、よりよい広域マイグレーションに向けての技術検証を継続している。また、広域マイグレーションに必要な、資源マイグレーションのひとつとして、ネットワーク資源のマイグレーションに関する要求項目をまとめた。

運用面では、WIDE Cloud の操作をウェブベースで実現する WIDE Cloud Controller (WCC) の運用開始、5 章で議論したネットワーク可搬性の実証実験、また完全 IPv6 運用に向けた NAT64 プログラムを開発し、運用を開始した。

来年度は、クラウドシンポジウムなどを通じての研究の方向性の確立、ネットワーク可搬性の実運用、効率的なライブマイグレーションに関する研究、広域ストレージに関する研究など、今後のクラウド技術の根幹となる項目の研究開発に引き続き注力していく予定である。

参考文献

[1] Keiichi Shima and Yuji Sekiya. *Network Portability Requirements and Models for Cloud Environment*. IETF, September 2010. draft-shima-clouds-net-portability-reqs-and-models-00.

[2] David B. Johnson, Charles E. Perkins, and Jari Arkko. *Mobility Support in IPv6*. IETF, June 2004. RFC3775.

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

[4] Keiichi Shima. map646: Mapping between IPv6 and IPv4 and vice versa, December 2010. https://github.com/keiichishima/map646/.