特集号
招待論文

GREEにおけるインフラストラクチャのサービス化とその意義

藤本 真樹[†1]

[†1] グリー（株）

IaaSという単語の普及が示す通り，インターネットサービスを支えるインフラストラクチャは，従来のラック，物理サーバを提供する形から，仮想化されたインスタンスをサービスとして提供されることが一般的になりつつある．一方で，2014年現在，IaaSにも課題はある状況であり，かつ，既存のオンプレミスなインスタンスを多数運用している環境では，単純にIaaSを利用すればよいわけではなく，それぞれの利点，課題を把握した上で，既存のインフラストラクチャを変化させていく必要がある．本論文では，GREEという実際に稼働しているサービスにおける，オンプレミスな環境をどのようにサービスとしてのインフラストラクチャへと変化させていくか，という事例を紹介する．

1. はじめに

2014年現在，インターネットサービスを構成するインフラストラクチャ（以下インフラ）は，IaaS（Infrastructure as a Service）という単語の普及が示す通り，急速に「サービス」として提供される形へとシフトしつつある．本論文では，インフラのサービス化とは何かを説明し，そしてその利点と難点がどこにあるのかについて紹介する．また，それらを考慮した上で，これまで独自に構築されてきたインフラをサービスとして提供する形へとシフトする際のアプローチと，その技術的課題についてGREEというサービスの事例を元に述べる．

2. インフラのサービス化

インターネットサービスにおけるインフラという言葉の定義は，その言葉が使われる主体やコンテキストによって実に幅広く変化する．また，そこに仮想化の概念が加わることにより，その定義は一層複雑になってきている．ここでは，まず本論文におけるインフラの概念について定義する．

2.1 インフラとは

インフラという言葉の指す意味は文脈によって変化することが多いが，本論文では，「インフラ」を構成する要素として，以下の4レイヤを中心にとらえて論じていく（図1）．

- ネットワーク
- コンピューティングリソース
- OS
- ミドルウェア

2.2 仮想化されたインフラとは

昨今，仮想化技術が進歩し，物理的なインフラの上位レイヤに，仮想化されたインフラを構築することが可能となっている（図2）．仮想化技術による最大のメリットは，従来，物理的なハードウェアと密接に関連していたインフラの構成要素が，APIを持ちソフトウェアから

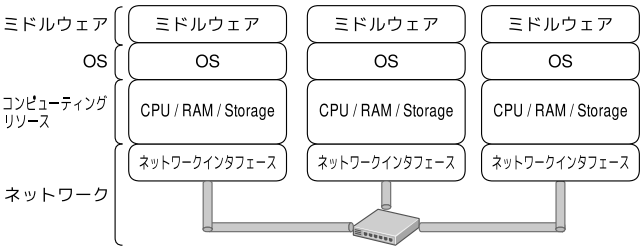

図1　一般的インフラ

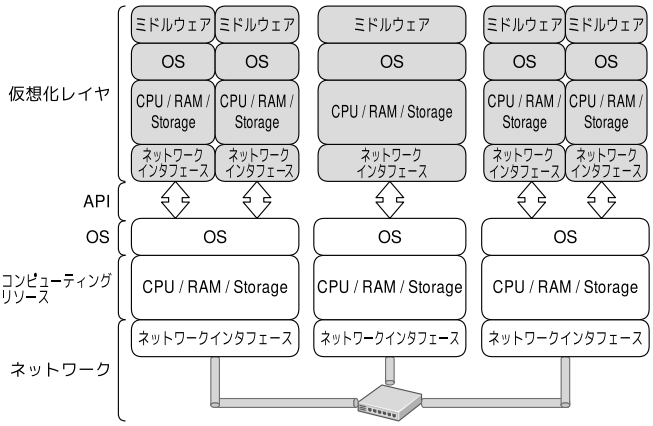

図2　仮想化されたインフラ

操作可能となる点にある．これにより，サーバインスタンスの起動，停止やその構成管理をプログラムから扱うことができるようになる．これは言い換えると，インフラそれ自体がプログラマブルなものとなることを意味している．

2.3　サービス化されたインフラとは

「インフラがサービスとして提供されている状態」とは，仮想化されたレイヤ全体を1つのインタフェースを通じて人間，あるいは機械から操作可能な状態，つまりインフラ全体を仮想化して利用者が扱うことができる状態になっていることといえる（図3）．いずれにしても，インフラをサービスとして提供する際に最も重要な役割を果たしている技術要素は仮想化技術であり，仮想化されたリソースを利用するためのソフトウェアの進歩である．

具体的には，ネットワークレイヤではSoftware Defined Networking，コンピューティングリソースレイヤでは仮想マシンの実装やコンテナの技術が一般的になっている．さらに，OS，ミドルウェアの各レイヤでは，構成管理を行うためのChef[☆1]やAnsible[☆2]といったソフトウェアを利用することで，インフラ自体を1つのソフトウェアとして扱うための技術的基盤が整いつつある．そして，現在（2014年）では，それらの基盤が構築されることにより，アプリケーションのデプロイにインフラの構成要素の管理まで含むことができるようになってきている．その一端はImmutable Infrastructureや，Blue Green Deploymentといった言葉の普及に見てとることができる．

☆1 http://www.getchef.com/chef/
☆2 http://www.ansible.com/home

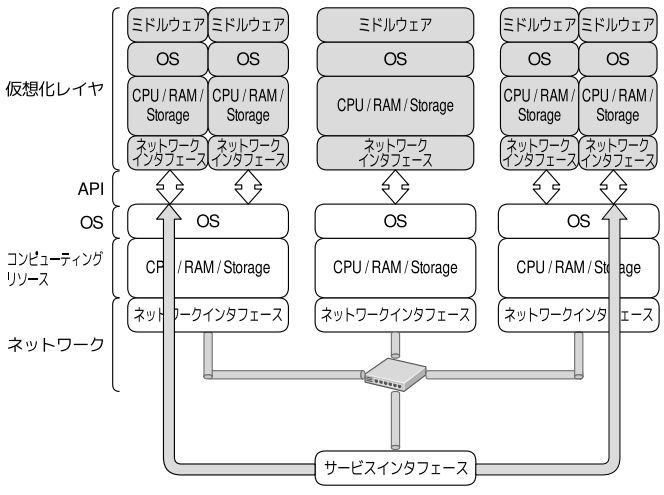

図3　サービス化されたインフラ

2.4　サービス化されたインフラの活用パターン

ここでは「サービス化されたインフラ」の意味付けを明確にするため，アプリケーションのデプロイを例にとり，従来のインフラと，サービス化された，すなわちAPIを持つインフラとでその振る舞いを比較する．

従来アプリケーションのデプロイは，Capistranoに代表されるソフトウェアを利用し，各アプリケーションサーバへのリモートコピーを行うことで実現されてきた．当然この枠組みの中でも，

- リバースプロキシとの連携をすることでデプロイ中におけるサービスの挙動の安定化
- ハードリンクを利用したディレクトリ切り替えによるインスタンス単位でのロールバックの実現
- Twitter, Inc.の開発したMurder[☆3]を代表例とするような，Bittorentを利用したデプロイ高度な並列化と高速化の実現

など，さまざまな取り組みが行われてきた．しかし，これらはいずれも，あくまでインスタンス単位での，アプリケーションのみのデプロイを並列化したものであると考えられる．

一方で，サービスとしてのインフラを利用することで，前述の通りインフラの構成そのものをソフトウェアとして記述し，かつ，それらをアプリケーションのデプロイのたびに構築することができるようになる．これによりアプリケーションのデプロイは以下のようにインフラ単位で実行し，切り替えることが可能となる．

1. 新たなネットワークセグメントを構築する
2. 構築されたネットワークセグメント上に，仮想化されたコンピューティングリソースのインスタンスを必要なだけ起動する
3. 起動されたインスタンスに対して，OS，ミドルウェアの構成変更を行う
4. 構成変更が完了したインスタンスに対してアプリケーションのデプロイを行う
5. デプロイが完了したネットワークセグメントに対してトラフィックの経路を変更する
6. （ロールバックが必要な場合は，n世代前に構築されたネットワークにトラフィックの経路を変更する）

重要な点は繰り返しとなるが，上記1.から4.までの手順により，インフラの構成をソフトウェアとして記述し，管理することが可能となっている点である．そしてこれは各レイヤにおける仮想化技術を導入することにより実現されている．

☆3 https://github.com/lg/murder

以上の通り，サービス化されたインフラにおいては，

- ネットワークセグメントのAPIを通じた構築
- コンピューティングインスタンスのAPIを通じた起動

などが典型例として増加しており，これらはいずれもソフトウェアを介したリソースとして形成されている．これを活かして従来では取り得なかった手法でアプリケーションを構築することが可能となる．次章では，そもそもなぜその必要があるのか，またインフラがプログラマブルになることによる副次的な必要条件と派生する利点について，インターネットサービス「GREE」の状況も例にとりつつ論じる．

3. インフラのサービス化の必要性

インフラ構築の方法としては，本論で紹介する仮想化技術を利用したサービス化されたインフラを利用する方法と，従前どおりの方法を考えることができる．ここで従前どおりの方法とは，たとえばデータセンター，ラック，ネットワーク機器，サーバハードウェアを準備してOSをインストールし，という手順でアプリケーションのインフラを構築することをいう．したがって新規にインフラを構築する場合，あるいは既存のインフラ資産を移行させるといった場合に，これらの技術の中から，どのような技術を最適解として選択していくかという問題ととらえることもできる．

3.1 インフラのサービス化による利点

サービスとしてのインフラは，古くは2006年にリリースされたAmazon EC2[☆4]を端緒として，そこからIaaSという単語とともに普及が始まった．そして，それらについては，ビジネスや経営的な視点，エンジニアリングからの視点などさまざまな論点から，その利点についての議論が行われている．

3.1.1 仮想化による利点 〜構成管理の安定化〜

仮想化による利点としては，「Infrastructure as a Code」という言葉が普及していることが示す通り，インフラ自体をソフトウェアから扱うことが可能となるという側面が大きい．すなわち，インフラ部分をソフトウェアで実現することにより，従来のソフトウェア開発で培われてきた，構成のバージョン管理やレビュー，そしてユニットテストやContinuous Integrationなどを利用することが可能となる．その結果，より安定した，かつ管理コストの低い形でインフラを構築，運用していくことができる．前述のデプロイの例についても，仮想化によるAPI提供とそれに伴うソフトウェアによるネットワークやコンピューティングリソースの管理機能の提供によって可能となるものである．

3.1.2 サービス化による利点 〜投資リスクの低減〜

サービスとしてインフラを利用することにより，一定の条件下で，コンピューティングインスタンスが必要になった任意のタイミングでインフラ全体の構成を変更することができるようになる．ここでインフラの構成変更とは，インスタンスを追加したり，ネットワークセグメントを追加したりといったことを指し，それらが動的に可能であることを意味する．これにより，たとえばトラフィックが見込まれる可能性が一定の確率である場合にも，事前に大規模な投資を行う必要がない，つまり，大規模な投資を行ったが，その多くが実際には不要となった，というようなリスクをコントロールすることが可能となる．

また逆に，あるサービスが縮小の局面を迎えた場合に，任意の単位，任意のタイミングでリソースを縮退させることも容易であり，常に必要最小限，余剰の少ない状態でインフラを保つことができる．これにより増減いずれの場合においても，主にコスト面でのリスクを低減することができる．

また，同様に，任意のタイミングでリソースを増減できることにより，従来より短いタイムスパンでインフラの構成を最適化することができる．つまり，通常，インフラの構成は予想される最大のトラフィックを予見して管理されるが，たとえば夜間トラフィックが低下する際にはリソースを縮退させ，移行ピークに合わせてリソースを増減させることにより，さらに効率的なコストコントロールが可能となる．

3.1.3 まとめ

上記で述べた投資コストに関する利点については，実際には特に自社で物理的なハードウェアを持つことをせずに，外部のIaaSと呼ばれるサービスを利用した場合に明確な利点となる．一方で，自社内で一定規模のハードウェアリソースを持ち，その上で複数のサービスを運用する場合には，即座に物理的なリソースを解放することはできない．しかし，この場合にも，自社内のインフラを受け持つ部門を独立した位置づけにおくことで，あるサービスAが縮退しており，異なるサービスBが伸張しているような局面において，より効率的に物理的リソースを活用するといった方法を採用することは可能であ

☆4 http://aws.amazon.com/ec2/

る．この場合にも，前述の利点の一部は享受することができる．

そして上記以外にも，たとえばインフラの技術的構成要素がサービスとして簡易な形で提供されることにより，専任のエンジニアなしでも運用を考慮することができるなど，人的な側面を含め，多くの可能性が考えられる．いずれにせよこれらの要素は，サービスとしてインフラが提供されることで実現される．そして，それによりソフトウェアからインフラを扱うことにより，任意のタイミングでリソースを追加，削減することが可能となる，という副次的な利点を得ることができる．

3.2　インフラのサービス化による問題点

前述の利点の一方で，インフラのサービス化は現状，発展段階にあるため，その実現の過程でいくつかの問題点が発生することも考慮しなければならない．

3.2.1　コストに関する問題点

問題点の1つ目は，コストに関してである．ここでいうコストとはインフラのサービスに対して支払う金額と実装コストの差分である．たとえばインフラが大規模になりスケールメリットが享受できるようになった場合を考えてみる．この場合，自社で物理的なハードウェアを確保し内部でインフラを構築したほうが安価で運用できるケースも少なくない．実際にグリー（株）においても外部のいわゆるIaaSと呼ばれるサービスを利用していたアプリケーションを，内部のインフラへ移行させたケースもある．GREEが依然として，内部でインフラを運用している要因はやはりこのコスト面の問題によるところが大きい．一方で，内部で完全にサービス化されたインフラを構築する場合，ネットワーク，コンピューティングリソースなどの各スタックのSDN (Software Defined Network) は普及途上であるがゆえに，その実装には，事前のテストや，未成熟なソフトウェアに対するデバッグ，未知の障害への対応など，やはり大きなコストがかかるといった問題も存在する．

3.2.2　遡行性に関する問題点

さらに，外部のサービスを利用した場合には，必然的に多くのスタックがブラックボックスとなる．この結果，それらの安定性やパフォーマンスがコントロールできないという問題に直面することも少なくない．したがって，一定以上のレベルでの安定稼働やパフォーマンスが求められる場合には，この点がリスクとなるケースがある．非常に些細な例ではあるが，たとえばAmazon RDS[☆5]というサービスはRDBMSのスタックをサービスとして提供しているが，ここでMySQL[☆6]のInnoDB Table Engineを利用した場合には，サービスの制約上変更できるパラメータに制限がある．このうちの1つにinnodb_log_file_sizeと呼ばれるパラメータがあり，これが変更できないことによるパフォーマンス上の制約が発生する．これは一例だが，外部のサービスを利用する場合には，こういった制約を前提として考える必要がある．

3.2.3　まとめ

以上のように，内部でインフラをサービス化していく場合には実装面でのコストやリスク，外部のサービスを利用する場合にはブラックボックスに起因するリスクが存在する．ただし，これらにより生まれるデメリットは，「オンプレミス」と呼ばれる従来型のインフラを利用し続けるメリットを上回りつつあると考えられる．それは，現在，大きな潮流として，IaaSのマーケットが伸び続けているという事実からも理解できる[☆7]．そして，新規にアプリケーションを構築し，かつ，外部のサービスを利用する場合には，そのサービスについてのノウハウを蓄積していくことが課題となる．この際には，既存の一定規模以上のインフラが存在している場合に，これをサービスとしてのインフラへどう遷移させていくか，ということがさらに大きな問題となってくる．これについて以降GREEを例にとり述べていく．

3.3　インターネットサービス「GREE」のインフラ

3.3.1　GREEについて

GREE[1]は，2004年12月に設立されたグリー（株）によって開発，運営されているサービスであり，SNSをはじめ，ソーシャルゲーム，ソーシャルプラットフォームなどのサービスが合わせて展開されている．このうち，トラフィックの中核となるのはSNSとソーシャルゲームの各サービスであり，それぞれ数百台，あるいはそれ以上の台数でサーバが稼働し，過去数年に渡り順次拡張され続けている．結果，2014年現在で1万台を超えるサーバが稼働する非常に大きなサービスとなっている．

3.3.2　インフラの概況

これらのサーバは，従来，すべて1物理サーバが1インスタンス，いわゆるオンプレミスなインスタンスとして稼働していた．インフラとしての構成は，それぞれアプリケーションサーバ，データベースサーバ，といった形でロールが定義され運用されていた．そしてそれら

☆5　http://aws.amazon.com/rds/

☆6　http://www.mysql.com/

☆7　http://onforb.es/1lIuQaa

の基本的な構成情報であるIPアドレスや物理的な配置，そしてロールや割り当てられているサービスを管理するために，サーバダッシュボードというシステムを開発し，運用していた（図4）．

また，それぞれのロールについては，一定のリソースがプールされており，かつアプリケーションのアーキテクチャとも連携してシステムとしてのスケーラビリティは確保されていた．しかし，そこではヒューマンオペレーションに依存する個所が多く，まさに従来通りのハードウェア，ソフトウェア両面におけるオペレーションが必要とされていた個所であった．したがって，ハードウェア障害発生時には，自動でのインスタンスの入れ替えは行うことができない前提で稼働していた．

これ以外にも，トラフィックの増減，特にピークタイムや，特定日のトラフィックの伸びが非常に大きい，内部での通信も多く，ノード間，スイッチ間，DC間の通信量を計算しながら，サービス，ロールをインスタンスに割り当てていく必要があるなど，インフラをサービスという形でアプリケーション開発者に提供するには難しい状況にあった．

3.3.3 インフラの戦略

こうした状況において我々は以下の点を問題と認識していた．一方で，これらの問題点はサービス化されたインフラを導入した場合に利点になる側面であると考えることもできる．

- あらゆるインフラのオペレーションに，インフラ側担当者と，アプリケーション側担当者とのコミュニケーションが必要となる．この結果，双方にコミュニケーションコストがかかり，本来集中すべきアプリケーション開発に割く時間が相対的に減少するといった問題やリードタイムが長くなる（インスタンスの追加，構成の変更など）などの問題が発生する．たとえばIaaSであればAPIリクエスト1回で完了するインスタンスの追加起動が，人的なコミュニケーションが発生することも含め，場合によっては1週間という期間を要することもある
- アプリケーショントラフィックの拡大によるコンピューティングリソースを追加する場合には物理サーバを追加していく必要がある．このため仮想化されたインスタンスを追加する場合と比較すると必然的により多くの対応時間を要するといった問題が生ずる．また，逆にリソースを縮小する局面において物理的なサーバが枷となり，効率的なリソースアロケーションが行えないといった問題も発生する
- 同様に，インスタンスごと，あるいはアプリケーションごとのコンピューティングリソースの利用状況の偏差が大きく，効率的にリソースを利用できない

一方で，金額，安定性，マイグレーションのオペレーションを考慮すると大規模なインフラを他のIaaSというカテゴリで整理される外部サービスへ移行するのは合理性を欠くとも考えられる．このため現在のGREEのインフラは（1）内部でインフラをサービス化する，つまり，オンプレミスなインスタンス群を仮想化しつつ，その上位レイヤでAPI，コンソールを提供することで，上記問題を解決する（図1の状態から図3への移行を進める）（2）他のIaaSとの接続を開始し，サービス初期や，突発的なリソース需要への対応についての柔軟性を確保する（図5），という2つの戦略を採用している．本論文ではこのうち（1）に焦点をあて，次章からはその具体的なアプローチについて述べる．

4. インフラストラクチャのサービス化のアプローチ

GREEのインフラをサービスとして提供する前提となるのは，第1章で述べた通り，コンピューティングリソース以下のレイヤにおける仮想化である．このレイヤが仮想化されている状態が構築されれば，以降のレイヤ，すなわちOS，ミドルウェアについては，ChefやAnsibleといった構成管理を行うソフトウェアを導入することで，インフラ全体をソフトウェアで扱うことが可能となり，結果として第2章最後で述べた，内部サービスと外部サービスを混在させることで突発的な需要に対して柔軟性を確保することが可能となる．

4.1 仮想化基盤の選択

本節では，仮想化基盤の導入における利点，難点と，GREEで導入されたOpenStack[2]の選択理由について述べる．

4.1.1 仮想化基盤とは

オンプレミスで稼働しているサーバ群をサービスとして稼働させるには，ネットワーク，そしてコンピューティングリソースを仮想化する仕組み，そしてそれらを人間あるいは機械から扱うためのインタフェースや，監視を始めとした運用を支えるためのさまざまなツール群が必要となる．これらを一括して，かつ連携させつつ提供するのが仮想化基盤であり，オープンソースソフトウェ

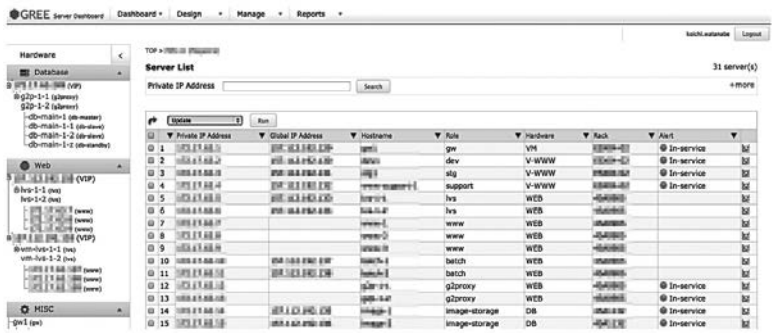

図 4　サーバダッシュボード

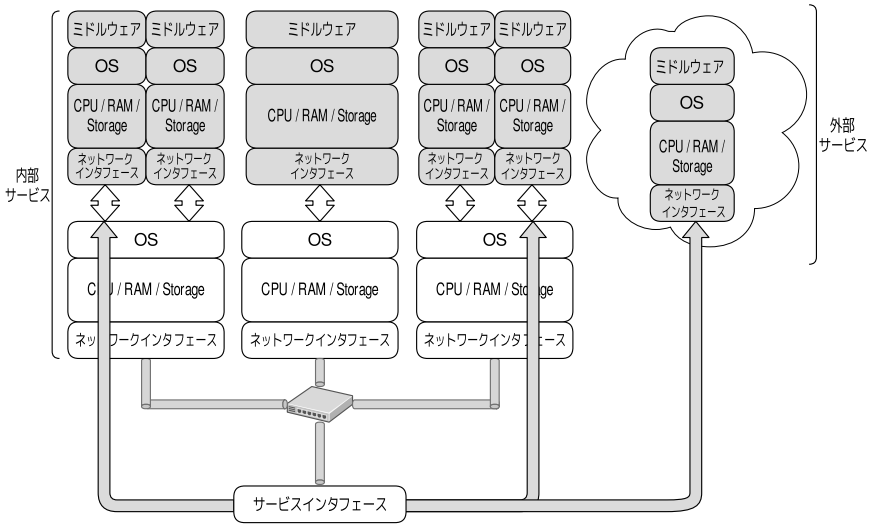

図 5　内部サービスと外部サービスの混在

アとしては，OpenStack や CloudStack[☆8] が有名である．

OpenStack を例にとると，サーバインスタンスの仮想化を行うコンポーネントや，それぞれのサーバインスタンスイメージを行うコンポーネント，ネットワーク管理を担当するコンポーネントなどを組み合わせて利用する形となる．

4.1.2　利点

基本的には，2.3 節にて記述した問題点である（1）サービス提供のリードタイム（柔軟な運用），(2) 効率的なリソースアロケーションを解決することが目標であり，かつ導入の利点である．実際の仮想化基盤の導入に当たっては，これらを改めて整理した上で以下を問題解決のゴールとして設定した．

- ネットワークセグメント構築と，コンピューティングインスタンスの起動と停止，そして構成管理の自動化とそれによる運用効率向上
  - ➢ 自動化によるオンデマンドでのコンピューティングリソース提供
  - ➢ 自動化された，かつ，高速なプロビジョニング
  - ➢ インフラへのソフトウェアテスト導入による安定性と運用効率向上
- コンピューティングリソースの利用率向上

アプリケーションごとに異なる規模に適切に対応したコンピューティングリソースを提供する．これにより適切なコストでの運用を可能とすると同時に，アプリケーションの可用性を保持する．

4.1.3　難点

一方で，仮想化基盤の導入により新たに発生する以下のような問題についても考慮しておく必要があった．

- パフォーマンス劣化

各種コンピューティングリソースへのアクセスにおける，仮想化されたサーバインスタンスを管理する Hypervisor のオーバーヘッド

- リソース制御

Noisy Neighbour 問題，つまり，同一物理サーバ上に複数のコンピューティングインスタンスが存在する場合に

① 発生するパフォーマンス劣化をどう解決するか
②その中でどういったサービスレベルでインフラを提供するかについてのクラスタリングが必要

といったことへの配慮が必要となる．たとえば，高度に安定性が求められるアプリケーションが複数存在する場合には，本来 1 物理サーバに 8 つのコンピューティングインスタンスを起動できるところを 4 つのインスタンスに制限する必要がある，というケースが発生し得るため，リソースの効率的利用とのトレードオフを考えなければならない

- システム全体の複雑化

単体のインスタンスで見ると管理コストは明確に下げられるが，システム全体として俯瞰すると，相互の依存関係が増加する場合もある．つまり「仮想化基盤」という新しい大きなソフトウェアコンポーネントが追加されることにより，複雑性は増加する．これはケースによっては全体としての運用コストの増加や，障害発生時の原因究明コストの増加を招くことがある

☆8　http://cloudstack.apache.org/

これらについては，いずれも前節で述べた利点とのトレードオフであり，さらにはここまで述べてきたインフラのサービス化によって得られる利点や変化とのトレードオフである. GREEでは, 少なくとも中期的視点からは, このトレードオフは十分に成立すると考え，仮想化基盤の導入を進めた．このトレードオフが成立するかどうかは，企業体におけるアプリケーションの性質や数，トラフィックのトレンドも影響する．GREEにおいては

(1) 仮想化基盤ソフトウェアの進歩による安定性と機能向上が継続的に行われるという仮定
(2) アプリケーションライフサイクルの短期化によるインフラ需要のボラティリティの増加，つまり2010年以前と比較したときに，アプリケーションがより短いサイクルで追加され，また，削除される傾向が生まれてきた

という両面を考慮して，仮想化基盤導入によりオペレーションコストを低減させ，1物理サーバインスタンスあたりの稼働率を向上させるメリットは大きいと判断した．次節ではこれについて詳細を述べるが，一方で，同時に考慮しなければならなかった点については，4.2節にて補足を行う．

4.1.4 OpenStackの選択

以上のような，利点，難点を考慮しつつ，GREEでは仮想化基盤としてのOpenStackの導入を行ったが，最適なソフトウェアスタックの選択は状況に応じて変化するため，ここでは主に，OpenStackのどこがGREEの現状に適合したかを述べる．

①Open Source Software

GREEではサービス開始当初以来，ほぼすべてのソフトウェアスタックはOpen Source Software（以下，OSS）で構成されている．このため，文化として障害発生時にソースコードレベルでの調査ができる，必要に応じてパッチを追加することができるOSSが非常に好まれており，これを重要な選定要件とした

②コミュニティの状況

OSSを選択する際には，そのコミュニティの活発さは非常に重要となる．利用する側として当然コミュニティへのフィードバックは行うが，できるだけ多くの開発者，ユーザが活動しているコミュニティを選択することで，ソフトウェアのより早期の成熟，機能の追加といった恩恵を得ることができるためである．この点で，OpenStackはCloudStackなどの他のソフトウェアと比較したとき，開発者数，コミット数といった定量的な値においても他のプロダクトを2013年以降上回っており，大きな利点としてとらえることができる[3]．また，企業としては取引先企業がサポート，または本格的に利用している場合はそれを利用することで初期コストやトラブルシュートのコストを下げることができることが多く，これも1つの要因となる

③API提供

前述の通り，インフラをサービスとして提供するにはAPIの提供が必須であり，これはOSSであることと同様に必要条件の1つである

④サーバハードウェア親和性

2014年現在GREEで標準的に利用されているハードウェアで問題なく稼働することも重要な条件である

⑤疎結合

これは導入状況によるが，GREEのように既存の大規模な資産が存在している前提での導入である場合，コンポーネントごとに柔軟に組合せを変更できる疎結合なスタックであることは重要な要素となる．実際にGREEでも，すべてのコンポーネントを利用しているわけではなく，主に利用しているのは以下の用途，コンポーネントである

- Keystone: 認証
- Nova: コンピューティングリソース管理
- Quantum / Neutron: ネットワーク管理
- Glance: VMイメージ管理
- Cinder: ストレージ管理

上記の①～⑤を主たる理由として仮想化基盤としてOpenStackを選択した．

4.2 固有の問題に対するアプローチ

GREEでのケースのように，すでにオンプレミスなインスタンスが大量に稼働している環境における固有の問題として，

①仮想化基盤導入による相対的なパフォーマンス劣化に関する問題
②一定以上のタイムフレームでオンプレミスなインスタンスと，仮想化基盤が併存することになるため，これら2つのインスタンス群をどのように扱うべきか

という2つの問題を解決する必要があり，本節ではこの問題へのアプローチについて述べる．

4.2.1 仮想化による性能劣化

仮想化されたインスタンスは，同一性能のオンプレミスなインスタンスと比較したときに，当然性能劣化があると考えられており，これについては多くの比較検証が

行われている．ここではグリー（株）で行われた検証についてのサマリを用いて，仮想化による性能劣化への対応方針について紹介する．検証は表1に示すハードウェアを用いて，オンプレミスなインスタンスに対する，仮想化されたインスタンスの相対性能を表2に示すパターンそれぞれにおいて計測する形で行った．

この結果のサマリとしては以下の通りとなる．

- CPU，RAMに関しては仮想化による性能劣化は，実用上誤差レベルと考えられる

ただし，Networkに関してはshort packetにおける性能劣化，Storageにおいては書き込みに関する性能劣化が認められる（図6，図7）．

これを受けて，GREEの仮想化基盤については全面導入ではなく，高いStorageへのI/O性能が求められるRDBMSなどの役割を持つサーバに関しては，引き続きオンプレミスなインスタンスを利用し，仮想化基盤と組み合わせて利用することとした．議論としては，性能劣化を受け入れつつ仮想化によるメリットを享受する，という選択肢も検討されたが，GREEの状況特有の議論として，(1) すでに大量のオンプレミスなインスタンスが存在し，いずれにしても併存させて利用しなければならないこと (2) RDBMSのようなアプリケーション固有のデータを保持するインスタンスに関しては，いわゆるアプリケーションサーバとは異なり頻繁に構築，破棄するものではないため，オンプレミスでの稼働によるオペレーションコストの影響がすくないこと，という2点を受け，上記の結論としている．

4.2.2　オンプレミスインスタンスとの併存

以上の通り，GREEにおいては完全な仮想化基盤を構築するのではなく，それらのインスタンスとオンプレミスなインスタンスを併存させる必要があるため，仮想化基盤自体を前述のサーバダッシュボードから扱うことで，同一なインタフェースを提供し，運用コストを削減する方針をとることとした．

表1　パフォーマンス計測ハードウェア

CPU	RAM	Storage
XeonE5506 (2.13GHz/4Core x 4)	32GB	15Krpm/RAID10

表2　パフォーマンス計測パターン

Pattern-A	Pattern-B	Pattern-C
1Hypervisor: 1VM	1Hypervisor: 2VM (1 running)	1Hypervisor: 2VM (2 running)

5. おわりに

ここまで，インフラのサービス化とは何か，そしてそれがなぜ必要なのか，それが何をもたらすかについてと，実際に稼働しているオンプレミスな環境において，これらをサービスとして提供するために必要な仮想化基盤の導入について記述してきた．前述の通り，仮想化基盤の導入自体のコストは確実に下がり，安定性は増しているが，インフラ全体をサービスとして，ソフトウェアのスタックとして扱うにはまだ解決しなければならない問題は多い．一方でこの概念自体は今後も変わらず重要なものであり続けると想定しており，引き続き改善をつづけていく必要がある．

また，本論文で触れていない大きな要素としてデータストアがあり，これについては前述の要素よりも遥かに複雑な問題を持ち得ることをここに補足しておく．こちらについても，近年開発が活発なCephを始め，商用，非商用含め多くの開発が進行しており，こういったソフ

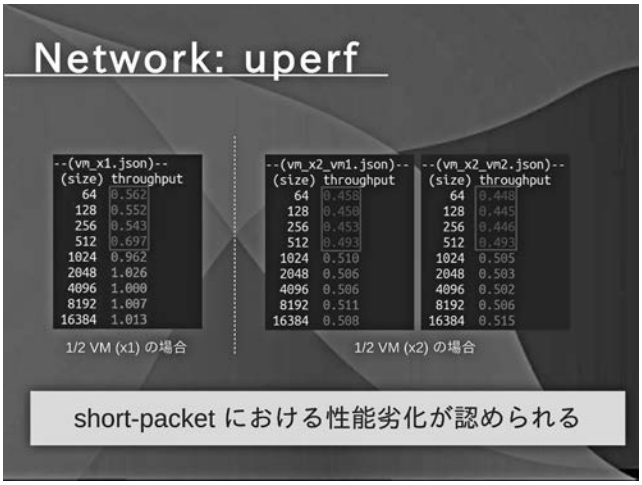

図6　ネットワークパフォーマンス比較

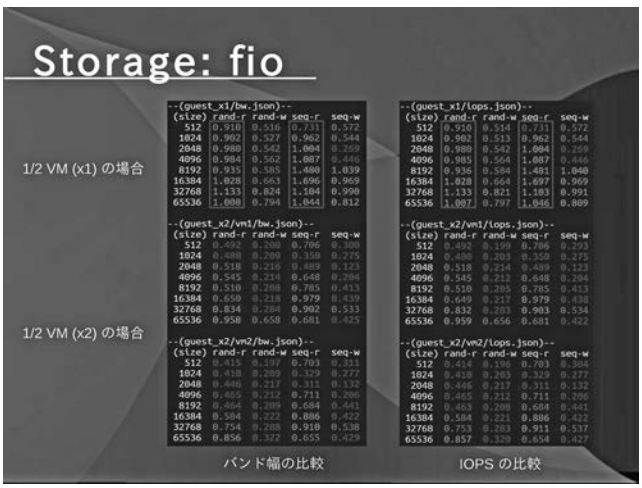

図7　ストレージパフォーマンス比較

トウェアの進化が，インフラのサービス化をより活発にしていくと考えられる．

最後に，直近のGREEのインフラサービスにおける改善点としては，(1) よりサービスとしての形を整えるためのUI/UXの向上 (2) Edge-to-EdgeやL2 Overlay，OpenStack L3 Agentのテストなどのネットワーク面でのテストと改善 (3) Dockerをはじめとするコンテナの導入を進めており，これらと合わせて，外部のIaaSサービスとのハイブリッド構成を採用することが技術上の重要なマイルストーンになっている．これらが実現できれば，オンプレミス，内部のインフラサービス，外部のサービスを有機的に組み合わせることが可能となり，それぞれの特性を活かした競争力のあるインフラを構築することが可能となる．これらを実現する上で，内部のインフラを仮想化，サービス化することは直近のソフトウェアスタックの進歩で現実的となり，有用な手段であると言えよう．

謝辞　本論文におけるプラクティスに尽力されたグリー（株）インフラストラクチャ本部チームメンバに感謝の意を表します．

参考文献

1)【GREE】ソーシャルネットワーキング，ソーシャルプラットフォームサービス : http://gree.net/
2) OpenStack: http://www.openstack.org/
3) CY14-Q1 Open Source IaaS Community Analysis: http://www.qyjohn.net/?m=201404

藤本 真樹（非会員）masaki.fujimoto@gree.net
2001 年，上智大学文学部を卒業後，（株）アストラザスタジオを経て，2003 年（有）テューンビズに入社．PHP 等のオープンソースプロジェクトに参画しており，オープンソースソフトウェアシステムのコンサルティング等を担当．2005 年グリー（株）取締役に就任．

採録決定：2014 年 9 月 5 日
編集担当：平山雅之（日本大学）