OpenStack クラウド構築のための準備

ハイパースケール（超大規模）環境用
プラットフォーム＆インフラの基盤づくり

デル テクニカル ホワイトペーパー

作成： Rob Hirschfeld、Greg Althaus

ご協力： Rackspace Cloud Builders 製品管理ディレクター、Bret Piatt 氏

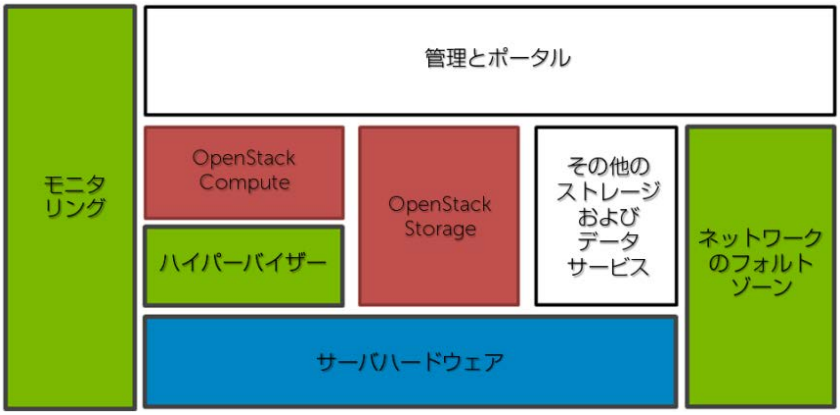
クラウド運用にもたらされる、オープンな API とベストプラクティス

目次

要旨

一般に、クラウドインフラストラクチャを立ち上げ、オンライン稼働にまでこぎつけるには、多大な労力と時間を要します。プライベートまたはパブリックサービスプロバイダとして看板を掲げるためには、まず、プラットフォームを選択し、ハードウェアを購入し、ネットワークを設定し、運用サービスをセットアップし、次に、これらをすべて統合しなければなりません。また、いくつもの作業を経て、やっと販売対象のアプリケーションをインストールするまでに至ります。

本ホワイトペーパーは、オープンソースの OpenStack と Dell Power Edge C シリーズハードウェアを使用したクラウドインフラを立ち上げる時、決定しなければならない諸項目を順に説明していきます。下図は、本書で説明するトピックスを図示したものです。赤は OpenStack、青はハードウェア、緑は操作と構成を示し、白い部分は将来デルから発行されるホワイトペーパーが対応する予定です。これらを参考にすれば、最終的にトライアルシステムの設計準備が整い、ハイパースケール（超大規模）クラウドの基礎を固めることができます。



始めの一歩

クラウド化を真剣にお考えですか？デルは、本書で説明している必須項目を中心とした「スタートキット」を開発中です。

このキットには、基本的なハードウェア仕様とツールが含まれており、サーバの開梱後、数時間で使用可能な OpenStack クラウドを立ち上げられるよう、工夫されています。

詳細は、OpenStack@Dell.com までお尋ねください。

プラットフォームの選択

本書では、インフラストラクチャのプラットフォームに Ubuntu 10.10 と OpenStack Bexar リリースを選択したものとして説明を進めます。基本的な概念はどの超大規模クラウドインフラでも同じですが、1 つのプラットフォームに絞った方が参照例を示しやすくなるため、このような前提にしました。OpenStack がオープンソースクラウドとして注目されているのは、次の理由からです。

- Amazon と Rackspace という、二大パブリックコンピュートクラウド API （Application Programming Interfaces）をサポート
- KVM と Xen という、二大オープンソースハイパバイザーをサポート
- Windows、Linux、その他の x86 ベース OS を使ってゲストを実行することが可能
- 複数のサイト（NASA、Rackspace、その他）で超大規模（1000 ノード以上）な導入計画が進行中
- 真のオープンかつコミュニティ主導の開発により、必要に応じた修正、サポート、機能拡張が可能
- 国際的な大規模コミュニティによって新機能が続々と追加

OpenStack には、イノベータ達の理想が各所で具現化されています。たとえば、既存のエコシステムをサポートしますし、クラウドサービスの基礎コンポーネントも提供でき、今後の方向性を決定付けるチャンスもあります。この基盤をしっかりと固めておけば、その上に完全なクラウドソリューションが構築できます。追加のサービスや機能拡張については、本書の最後に説明します。

最近、「オープンソースクラウドの OpenStack を試してみたい」というお客様達が、デルに支援を求めてくるようになりました。デルの OpenStack スタートキットは、特にトライアルをターゲットとしており、基本的な OpenStack クラウドを素早くセットアップして、構成方法を迅速に習得することができます。

OpenStack には 3 つの主要コンポーネントがあり、それぞれ、「Compute（コンピュート）」（Nova）、「Object Storage（オブジェクトストレージ）」（Swift）、「Image Service（イメージサービス）」（Glance）と呼ばれています。本書が主に取り上げるのは、OpenStack 実行環境の準備です。OpenStack を手動でインストールする全手順については、他の資料を参照してください。

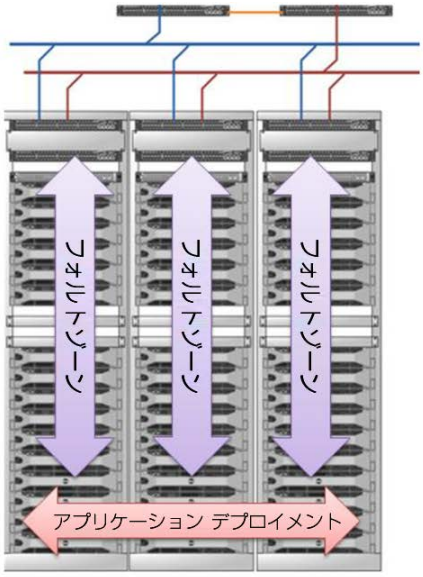
注： デルは、OpenStack を使った概念実証（PoC）にご興味のあるお客様を積極的に探しています。デルの協力のもとに PoC を実施すると、本書でご紹介する多くの構成およびサービスが活用でき、さらに、Rackspace Cloud Builders から提供されるサービスも活用可能です。詳細は、OpenStack@Dell.com までお尋ねください。

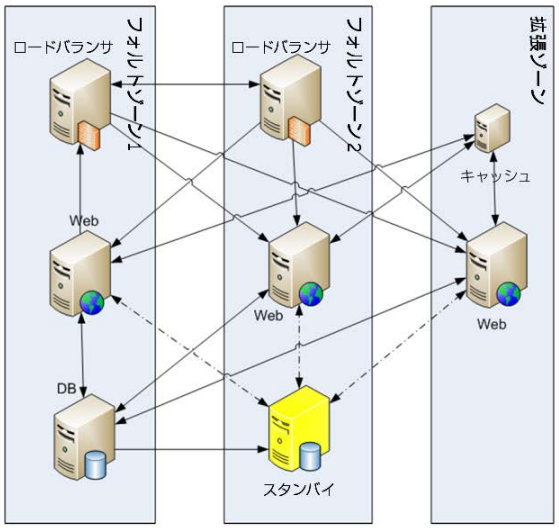
基本的なハイパースケール設計パターン

フォルトゾーン

超大規模クラウドを構築するには、従来のエンタープライズ仮想化インフラにまつわる固定観念を打破しなければなりません（いわば「革命的」な思考です）。これは、ほとんどのエンタープライズシステムより、シンプル化、統一化、密度のレベルを格段に上げることを意味します。

これらの超大規模システムで定石となるのが、「今までハードウェア側で実現していた冗長性を、ソフトウェアとアプリケーション側に移す」という考えです。実際、数千台ものサーバを運用していると、障害など日常茶飯事となるため、「障害は起きるもの」という前提でシステムを取り扱います。





大規模な拡張を達成するため、各コンポーネントは、ネットワーク、電源、ディスクの冗長性といった機能性を故意に縮小しています。サーバは、1 本のネットワークパスと 1 台の電源装置だけを搭載し、ドライブも非 RAID 構成です（これを、「JBOD＝Just a Bunch of Disks、単なるディスクの集まり」と呼びます）。したがって、配電ユニット（PDU）やラックスイッチが故障すると、いくつものサーバがダウンすることになります。このリスクに対応するため、システムをいくつかの「フォルトゾーン」と呼ばれるグループに分けて、アプリケーションとデータをフォルトゾーン間でストライピングしています（RAID のデータストライピングに似た手法）。これにより、複数のコンポーネント障害の影響を切り離すことができます。

下記のとおり、このアプローチには多大なメリットがあります。

- コンポーネント（ディスク、サーバ、ネットワーク）を冗長化しない選択ができるため、総所有コスト（TCO）が削減できます。
- ネットワークのルーティングと構成がよりシンプルになります。
- データセンターの物理的なレイアウトが簡潔になります。

「ハイパースケールクラウド」とは？

ハイパースケールシステムは、1 つの管理インフラで数千台ものサーバを運用するために設計されました。これほど多数のシステムでは、ハードウェア障害が日常的に起こり得ること、手動のステップは実用的ではないこと、小さなコストの違いが全体では大きな差となって現れることから、これまでとは異なる管理手法が必要です。

「塵も積もれば山となる」コストの一例を挙げてみましょう。たとえば、6 ドライブアレイで RAID 5 を構成していたシステムを、RAID 10 に変更した場合、合計ストレージ容量は 40 % も減少してしまいます。言い換えれば、同じだけのストレージ容量を得るために、66 % も多くのディスク数を購入しなければなりません（6 ドライブではなく、10 ドライブが必要）。

- ディスク、ネットワーク、電源を冗長化しないため容量が最大限に利用でき、密度が増します。
- セットアップと導入プロセスが定型化および合理化できます。

ここで重要なのは、基幹ネットワークについては依然、ハードウェアのフォルトトレランスパスを使用した冗長構成が必要であることです。

このアプローチでインフラストラクチャを構築する場合、アプリケーションも、フォルト「ゾーン」トレランス対応で導入する必要があります。この RAIN（Redundant Arrays of Inexpensive Nodes）を使用したアプリケーションストライピングについては、別のプレゼンテーション資料とブログで詳しく述べています。

フラットなエッジの重要性

「フラットなエッジ」は、ハイパースケールクラウドの重要な設計原則の 1 つです。「フラット（平坦）」とは、なるべくクラウドインフラに階層を作り込まないようにする、という意味です。たとえば、ブレードをフレーム集約型ネットワークに導入し、これを VLAN 経由で SAN に接続する場合は、階層化設計となり、コンポーネントが縦方向に積み重ねられます。一方、ローカルディスクを使い、スイッチに直接接続したシングルノードは、すべて同じコンポーネントを扱うことになりますが、1 つの「フラット」なレイヤとなります。また、「エッジ」とは、クラウドの最下位層（または「葉」）のことです。構造をフラットにすると、ほとんどのコンポーネントが自己内包型となるため、多くのエッジが形成されることになります。複雑化を抑えながら規模を大きく広げるには、クラウドをエッジ中心に構築する必要があります。こうすることで、ネットワークトラフィックの経路指定、レプリケーションデータの保存先、仮想マシン（VM）のスロットリング条件などが、それぞれ個別に決定できるようになりますし、インテリジェンスのオーバーヘッド負荷を（「中央集中化したオーバークラウド」に一括制御させるのではなく）クラウドの各コンポーネントに効果的に分散させることができます。

注： VLAN を使ってテナントごとにセグメント化する方法は、「エッジ」型設計とは逆の例となります。その理由は、スイッチング階層側で VLAN（制限付きのリソース）を構成して、エッジコンポーネントが生成したトラフィックを管理するようにしなければないからです。

「フラットなエッジ」などの概念は、超大規模クラウドの運用に基づくものです。多くの場合、ハイパースケールの設計要件は、従来型データセンターが目指すものと相反します。それは両者で、中核となる前提条件が異なるからです。

Dell データセンターソリューション（DCS）グループは、何年もの間、この規模のクラウド構築でお客様を支援してきました。これらのハイパースケールデータセンターで培ってきた革新技術が少しずつ下方に流れ出てきて、今や、中規模環境でも効果的に応用されています。

ハードウェアの選択

クラウドハードウェアを選択する際は、運用形態に合わせたフォルトトレランス手法と足並みを揃える必要があります。ハイパースケールクラウドに要求されるのは、高度にモジュール化された高密度なソリューションです。RAID システムが相互交換できるコモディティディスクの使用にこだわるのと同様、クラウドも相互交換可能なユーティリティサーバを使って構築されます。ここで鍵となるのは、冗長化するために十分な拡張性があること、そしてさらに重要なのは、自在に増設できるモジュール型設計であることです。

シンプル化にはこの「モジュラリティ」が欠かせません。クラウドが数百ノードを数えるようになると、ノード管理、すなわち、6 台ずつのグループで専用 SAN にリンクし、次に、8 組またはそれ以上の NIC チーミングペアを異なるクロス接続スイッチに繋げる、といった管理が難しくなってきます。システムそのもので既に難しいのですから、これらを設計し、文書化し、維持することがどれだけ困難か、想像に難くありません。

原則としてハイパースケールクラウドは、意図的に共有物理インフラを縮小しています。その理由は、共有物理インフラの場合、構成、管理、トラブルシューティングが難しくなるからです。また、共有しているために、障害の影響が多数のシステムに波及してしまうのも弱点です。一方、ハイパースケールモデルの場合、個々のコンポーネントで障害が起きやすくなったとしても、その影響は小さく、他から切り離しやすいうえ、修正もはるかに簡単迅速です。

これまでの経験則から、ノードは次の 4 種類のうち、いずれか 1 つの性能カテゴリに振り分けられます。

- **「コンピュート」**ソリューションは、仮想マシンベースのクラウドほど一般的ではありませんが、特定の分析システムでは典型的に採用されています（興味深いのは、多くの分析法がディスクおよびネットワーク指向であることです）。クラウドアプリケーションは実際、スケールアップではなく、スケールアウト寄りとなります。
- **「ストレージ」**ソリューションは、扱いに注意が必要です。このソリューションには、大規模なデータをはるかに容易に中央集中化でき、すべてのデータをローカルノードにまで引き込む必要のない IP ネットワークベースの iSCSI SAN か NAS ストレージをお勧めします。注： 非常に大容量のストレージと多数の VM を必要とするソリューションは、クラウドアプリケーションに向かない可能性があります。
- **「ネットワーク」**ソリューションは、演算（コンピュート）集中型のシステムに見えることがありますが、システムに大容量の RAM と CPU を詰め込まない限り、ネットワークバンド幅の壁に突き当たることはありません（詳細は後述）。リモートストレージは、ネットワーク容量のニーズを増やす主な原因となるため、ローカルストレージの利用を高めれば、ネットワークにまつわる制限を解決できる場合があります。
- **「バランス」**ソリューションは、たとえごくベーシックな VM 配置でも、VM を十分に分散させてリソース利用を均すことができるため、優れた折衷案となります。いずれライブマイグレーションが標準提供されれば、さらに容易になります（OpenStack Cactus リリース前に提供される予定）。

これらの 4 カテゴリを出荷中の Dell PowerEdge C サーバモデルに当てはめると、コンピュートには、バランス重視のサーバが最も幅広いアプリケーションに対応できますし、ストレージには、ストレージノードが最適な選択となります。トライアルシステムには、バランスノードを推奨します。その理由は、クラウドの成長に合わせて、すぐ別の用途に変更できるからです。

フォーカス	デル モデル	サイズ	コア数	RAM	ディスク /ノード	ディスク /コア	ネット /コア
コンピュート	PEC 6100 4 スレッド（上図）	2 U	32	192	6	3:16	1:2
バランス	PEC 6100 2 スレッド	2 U	16	96	12	3:4	1:2
ストレージ	PEC 2100	2 U	8	48	24	3:1	1:2
ネットワーク	PEC 2100 +10 Gb NIC	2 U	12	48	24	2:1	～2:1

前提条件：

ノードあたり 48 ギガバイト（GB）（実際の RAM は増減可）

- 2.5 インチのドライブはスピンドル数が増える。一方、3.5 インチのドライブは、より大容量かつ割安だが、IOPS（1 秒あたりの I/O 処理件数）は落ちる
- 「ディスク/コア」比率は、RAID を構成*しない*ドライブを想定。RAID を採用した場合、数が減少する
- 後述の「ネットワーク構成」ガイダンスに従い、ノードあたり 4 NIC を搭載

ターゲット比率を決めると、それがハードウェアの主な選択基準となります。たとえば、コンピュートをプランニングする際、VM あたり 1 コアとすると（控え目な密度）、バランス重視のシステムでは、VM あたりほぼ 1 スピンドルを得ることになります。これで、各 VM に専用 I/O チャネルを効果的に作成することができ、十分なストレージが確保されます。もっと高密度をターゲットにしても構いませんが、「1 コアクラスの VM にしておけば、ほぼ専用のリソースが確保される」ということを覚えておくと便利です。フラットなエッジを心がけると、VM レベルでこの種の切り分けが促進されます。それは、最大限に細分化することで相互依存性がなくなるからです。

RAID：
「使うべきか、使わざるべきか。それが問題だ」

コンピュートノードでハードウェア RAID を使用すると、ユーザ データの安全性が一層強化されます。したがって、ユーザが、重要なデータにネットワークストレージを使用する予定がないとき、または、複数のノードに拡張する予定がないとき（または、そのように強制するとき）は、RAID の重要性が増します。

RAID のデメリットは、使用できるストレージ容量が減少し、コストが増え、複雑化することです。VM I/O が均一でない場合、RAID は、JBOD 構成より性能が低下することもあります。

最終的には、運用能力とリスク対応能力に応じて、RAID を採用するか否かを決定してください。

ストレージハードウェアは、コアあたりのディスク比率を十分に高くできないことがあります。Swift のようなソリューションでは、最も電力効率の良い CPU と最大容量のディスクを検討するようお勧めします。オブジェクトストアは通常、フロント側でキャッシュされるため、使用頻度の高いファイルは、実際のストレージノードをヒットしません。

それでは、ストレージとコンピュートシステムを結合させた構成法から見ていきましょう。このモデルでは、同じノードがコンピュート機能とストレージ機能の両方を兼ねます。この構成の場合、一見すると、ネットワーク最適化ノードが適していると思われるかもしれませんが、10 ギガビット（Gb）ネットワークは、他の異種混在（ヘテロジーニアス）システムに比べると、依然、極めて高額なため、私達の一貫した見解は、「兼用ノードは妥協する点が多過ぎて、結局、高額になってしまう」ということです。ただし、1 つだけ例外があり、小規模なパイロット（試行）目的では、兼用システムをお勧めします。これなら最も柔軟性に富みますし、クラウドインフラの使用を学ぶのにも適しています。

何を設計するにせよ、注意すべきなのは、例外への対応が発端となって、設計全体が次善策に陥らないようにすることです。たとえば、100 GB ディスクの VM を何台かホストする必要があるとき、これが原因となって、インフラストラクチャ全体に「大容量ストレージ重視」を強要してはなりません。この場合、まず、高速なローカルディスクを使用した 20 GB の VM を前提とし、次に、例外対策用の二次ドライブとして、シングルシェアの iSCSI SAN か NAS ターゲットをセットアップする方が賢明です。サービスプロバイダの場合これらの例外はプレミアム機能となります。

ハイパースケールクラウドは、基本的にマルチ・テナントです。互いに関連性のない作業を混在させられる能力は、大規模クラウドのロードバランスも可能にします。需要は動的に発生し得るため、伸縮自在なリソース機能がこれに対応します。

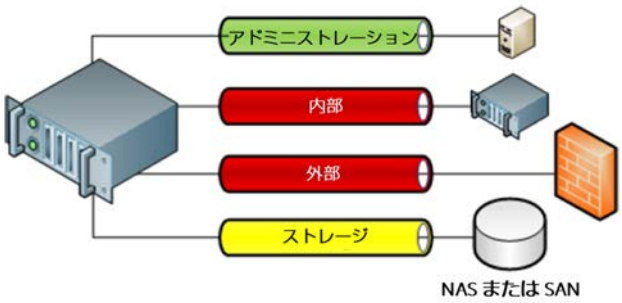
マルチ・テナントという前提は、要件のパラドックスを生み出します。テナント同士を「切り離す」必要があるのに、積極的な「混在」も進めなければなりません。この矛盾を解決するのが—当然と言えば当然ですが—「コンピュート、ネットワーク、ストレージの仮想化」です。

ネットワーク構成

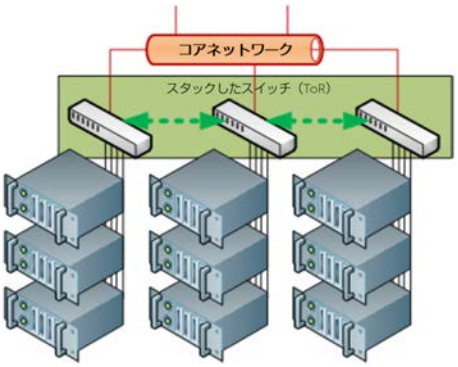
ハイパースケールクラウドにとってネットワークがいかに重要であるかは、語っても語り切れませんが、いずれにせよ、その重要性が複雑化に繋がってはなりません。クラウドネットワークのキーワードは「シンプル化」と「フラット化」です。この設計目標を達成するには、エンタープライズのネットワークトポロジと正反対の選択が必要となります。

典型的なトポロジ

ハイパースケールクラウドのベストプラクティスとして、主に 3 つの論理ネットワークが挙げられ、さらに、もう 1 つ加わります。実用的には、通常、これらのネットワークを直接、物理 NIC にマッピングしますが、マッピングは必須ではありません。



1. **アドミニストレーション**ネットワークが、クラウドワークロードを実行する各ノードと、クラウドインフラ管理を接続します。このネットワークは制限されており、VM にアクセスすることはできません。詳細は「運用インフラストラクチャ」セクションをご覧ください。
2. **内部**ネットワークは、VM とクラウド内のサービス（オブジェクトストアなど）を接続します。このネットワークは通常、大量のクラウドトラフィックを搬送しますが、内部で消費されるバンド幅については顧客に課金されません。
3. **外部**ネットワークは、VM とインターネットを接続します。この接続は、顧客に課金できるよう使用量が測定されます。
4. 中央ストレージを採用し、他のネットワークで発生する大量転送の影響を切り離したいときは、**ストレージ**ネットワークの使用を推奨します。ストレージトラフィックを VM から切り離す場合、ストレージをアドミニストレーションネットワークに連結する、という手段も考えられます。



ネットワークをセグメント化する理由はいくつかありますが、第一の理由は、バンド幅の分散化です。利潤を生みだすネットワーク（外部）は、他のネットワーク処理から支障を受けないようにしなければなりません。また、セグメント化は、IP 管理の強化にも有効です。しかし、意外かもしれませんが、セキュリティは、セグメント化の理由になりません。マルチ・テナントクラウドでは、内部ネットワークにアクセスすることのできる VM 内に、信頼できないユーザが侵入し得る、という仮定も捨てきれないため、侵入者を除外するには、もっと強力な別手段が必要です。

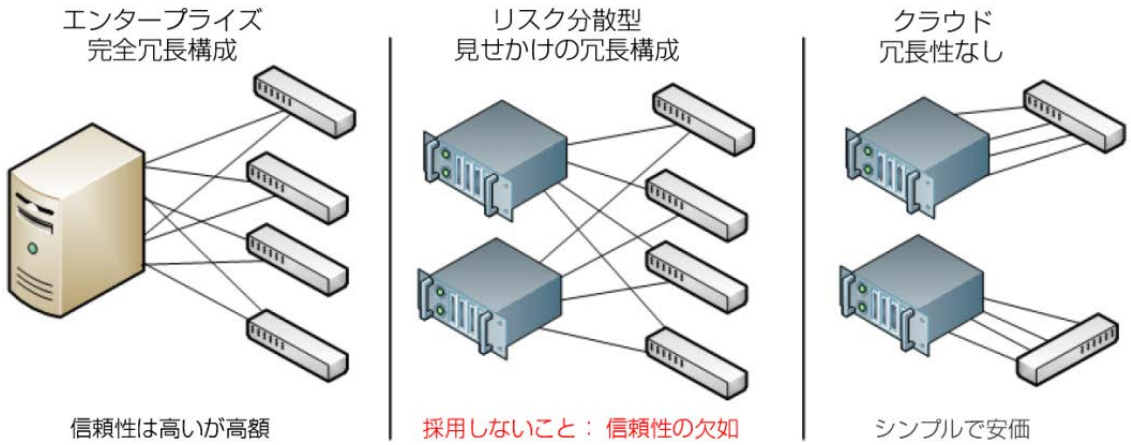
いずれ 10Gb ネットワークのコストが下がれば、これらの論理ネットワークを 1～2 個の物理 10Gb NIC にマッピングすることがトレンドになると思われます。1 つの 10Gb ポートが、4 NIC 構成の 2 倍以上のトラフィック[1]を搬送できるため、単一インタフェースに絞り込むことは理に適った設計です。さらに、単一の高速 NIC 設計なら、バンド幅も柔軟に利用できます。たとえば、1 つのネットワークの使用量が容量の 80 % まで急伸しても、依然として他のネットワークに 1Gb のバンド幅が残されます。ここで忘れてはならないのは、運用管理ネットワークは、マザーボードの IPMI（Intelligent Platform Management Interface）と管理にアクセスしなければならないため、通常、二次インタフェースには載せられない、という点です。

設計ガイドライン

すべてに適合できる 1 つの万能なトポロジなど無いため、クラウドネットワークの構築にまつわるいくつかの基本ルールを重要な順にご紹介します。以下のルールは、クラウド接続基盤をしっかりと固めるのに役立ちます。

ルール 1： コスト重視

容量の一部が未使用のまま残されてしまうのは、経費の無駄使いです。アイドル状態のバックアップリンクや、十分に活用されていないバンド幅には、二倍以上のコストがかかります。複雑化もコストを増やす原因です。互いに重なり合う VLAN、複雑なルーティング規則、洗練されたアクティブ/アクティブ型のパスには、高額な管理ツールや手動での対応が求められます。通常の企業データセンターなら、クラウドのように自在に伸縮させる必要がないため、完全冗長型のネットワークに投資するのも理解できますが（通常、8 個以上のインタフェースを 4 台のインタリーブスイッチに接続する必要がある）、クラウドの場合、コスト上の理由から、余剰な（そしてより高額な）機材を購入、インストール、管理することは不可能です。



見せかけの冗長構成について

スイッチとサーバの一対一マッピングは、リスクが大きいように見えるかもしれませんが、実際には、サーバのネットワーク負荷を 4 台のスイッチに分散させるより、信頼性が 3 倍以上向上します。

冗長性をきちんと確保すれば話は別ですが、システムにむやみにコンポーネントを加えることは、信頼性を落とす原因となりかねません。

[1] 10 Gb NIC はより広いバンド幅を提供しますが、パケット数に制限があるため、全容量を活用し切れない場合があります。たとえば、VM が大量の小さなパケットを使って重い負荷を送信すると、1 つの 10Gb NIC では飽和する可能性がありますが、複数の 1Gb NIC なら処理できる場合があります。

ハイパースケールクラウドネットワークには、フォルトゾーンが含まれるため、よりシンプルなスイッチを採用できます。ここで推奨する構成は、各サーバをそれぞれ 1 台のスイッチのみに接続する方法です。この方法なら、スイッチ数もポート数も少なくて済み、ワイヤの長さも短くなりますし、高度なパスを張りめぐらす必要もありません。ノードに接続するコンポーネント数が増えれば増えるほど、ノードの信頼性は低下します。この場合、信頼性を向上するには、高額で複雑な冗長機材を追加するしかありません。ハイパースケールクラウドは、システムレベルの冗長性に加え、より低コストなリソースをフォルトトレランス向上策として利用しています。

ルール 2： フラットなネットワークを維持

「4,096」とは、大きな数です。これは、ユーザが特別なことをしなくても、ほとんどのネットワークがサポートする VLAN の最大数です。論理ネットワークを作成し、ブロードキャストドメインを管理するには、いくつかの VLAN が必要となりますが、VLAN を使用してテナントトラフィックをセグメント化すると、拡張性が低下します。現在、デルが推奨している密度は、ラックあたり 36 ノードです。各ノードで 32 VM ずつ（コアあたり 4 VM）サポートする場合、ラック 1 台で 1,152 台もの VM を維持し、ほぼ 2,500 個の IP アドレスを割り当てなければなりません。このような高密度システムで階層型ネットワークと VLAN を管理することは現実的ではありません。したがって、クラウドネットワークはできる限り階層化せず、フラット（平坦）を保つべきです。

クラウドネットワークでデルが提唱する参照用設計では、スタッキングを使用して、論理的なラック上（Top-of-rack）スイッチを構築します。このスタッキングは、短距離 14Gb ネットワーキングを使用し、すべてのスイッチを効果的に管理することができます。これにより、ラック内のノード間通信は極めて高速かつシンプルになり、スタックされたスイッチ間で、スイッチごとのコアルーターに渡される 10Gb アップリンクを共有できます。この方法では、すべてのトラフィックをコアにルーティングするという犠牲を払わずとも、各スイッチが分割されたフォルトゾーンを形成することができます。

ルール 3： エッジでのフィルタリング

VLAN は拡張性に乏しいため、何か別の方法を使って、テナント間の混信を防止しなければなりません。その解決策とは、ノードレベルで、エッジのトラフィックフィルターを設定することです。これには、クラウド管理システム（OpenStack Nova）を使用して、各導入 VM にネットワークへのアクセスルールをセットアップする必要があります。このルール設定で、同じテナントからの VM 間通信を許可し、その他のトラフィックはブロックするような制御が必要です。現在のところ、この種のフィルタリングにお勧めするツールは Linux IPTables ですが、今後、OpenFlow や Open vSwitch を使用した新しい手法にも期待が寄せられています。

ルール 4： フォルトゾーンの設計

クラウドを構築される方は、ご自分のネットワークトポロジ内で、フォルトゾーンがすぐ特定できるようになる必要があります。前述のとおり、フォルトゾーンの目的は、障害の及ぶ範囲を限定すること、そして、設計をシンプルにすることです。新人のデータセンター技術者から、高度に自動化されたクラウド管理システムまで、すべてがトポロジを理解する必要があります。

ルール 5： ローカルトラフィックのプランニング

一般的な傾向として、クラウドアプリケーションは、従来の階層型設計に比べて、内部通信の多いスケールアウトアーキテクチャとなります。この方式は、フォルトゾーン間に処理を分散させることで信頼性を確保しますが、そのために大量の内部ネットワークトラフィックが生じます。コアネットワークを経由して、この内部トラフィックをスイッチ間にルーティングさせる必要があるときは、コアバンド幅をオーバーサブスクライブすることが可能ですが、外部通信に支障を与えかねません。しかし、幸いなことに、内部通信は主に各テナントの VM 間で発生するため、予測可能です。対策として、追加のアウトバウンドリンク、Top-of-Rack 型のスタッキングスイッチ（上記「ルール 1」）、テナントのクラスタリングを加えることで、当該トラフィックのほとんどが同じコアスイッチに集約されるため、問題が軽減されます。

複雑さをテストする「30 秒ルール」

コンピュータサイエンスを学んだ方であれば、「複雑性」を計算するアルゴリズムをご存知だと思います。しかし、データセンターが複雑か否かを判断する場合、このアルゴリズムはあまり実用的ではありません。

デルが提唱する「複雑性ルール」に博士号は不要です。

まず、あるデバイスを思い浮かべてください。次に、それが故障すると、どこに影響が及ぶか答えてください。30 秒以内に答えが出なければ、あなたの設計は複雑過ぎます。

ルール 6： ロードバランサ（負荷分散器）の提供

最後のルールは、クラウドユーザによる健全なアーキテクチャの形成に役立ちます。低コストで使いやすいロードバランサを提供すると、顧客のアプリケーションのスケールアウトを促すことができます。クラウドプロバイダは、ハイパースケールで高くなりがちなエッジ障害のリスクを軽減するために、フォルトゾーン間にいきわたらせることのできる、スケールアウト型アプリケーションが必要です。パブリッククラウドによっては、コアサービスの一環としてロードバランサも組み込んでいるか、VM にロードバランサを事前構成して容易に利用できるようにしています。顧客にアプリケーションのスケールアウトを奨励しない場合は、プロバイダ側のヘルプデスクや運用チームのスケールアウトを計画してください。

運用インフラストラクチャ

デルがクラウドの立ち上げで学んできた重要な経験則の 1 つは、「ハードウェアやソフトウェアと同様、運用能力も成功を左右する要因である」ということです。プランニングするのが 1,000 ノードのパブリッククラウドであろうと、6 ノードのラボであろうと、運用に必要となる中核コンポーネントに変わりはありません。これらのサービスは、コアネットワークサービスから、モニタリング、プロビジニング、アクセスへと順番にレイヤを積み重ねていきます。

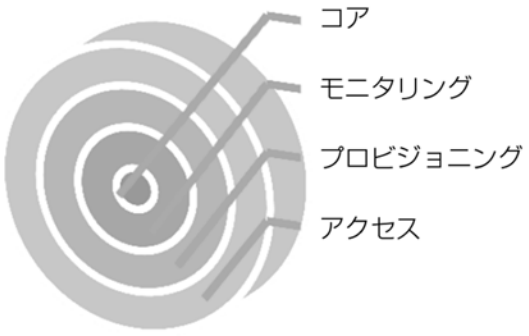
アドミニストレーションサーバ

導入する特定のサービスに飛び移る前に、まず、インフラストラクチャのごく一部（100 ノードにつき 1 台）をアドミニストレーション（アドミン）サービスとして割り当てることが重要です。私達のデプロイメントスクリプトはすべて、このサーバを最初に構成するよう記述されており、ここから、残りすべてのインフラストラクチャが依存するオペレーションサービスが提供されます。このサーバは、厳格に内部インフラの管理のみに使用されるため、外部 API やポータルを実行することはありません。立ち上げ時は、これがイメージサーバとなり、かつ、デプロイメントマネージャにもなります。立ち上げ工程が完了したら、要塞ホストとして、または、モニタリングシステムとして利用することも可能です。クラウドの運用がはるかに容易になるため、デルでは、たとえ最小のシステムでも、アドミン専用のサーバを設置しています。以降で説明するとおり、アドミンサーバは実に働き者です。

コアサービス

コアサービスは、ほとんどの基本アクセスとインフラの調整をサポートします。インフラ全体がデータセンターなみの運用能力を期待するため、これらのサービスはクラウドの運用に欠かせません。シングルノードをターゲットとした SQL サーバとは異なり、クラウドソフトウェアは、「ネット」で使えるすべてのサービスとネットワーク接続を網羅した、インターネットデータセンター内の運用を想定しています。

以下に、コアサービスの一覧を示します。



通常稼働時の DHCP 使用については、何度か評価が揺れました。最初は依存関係が 1 つ増えること、また、それを維持しなければならないことに抵抗がありました。しかし最終的には、DHCP がブートアップ構成を提供するうえ、PXE 統合を維持できることから、アドミンネットワークに対する DHCP の有効性を認めることとなりました。現在運用中のインフラストラクチャは、DHCP と PXE を使用し、パッチイメージ経由でブートすることにより、BIOS のアップデートを自動化しています。

サービス	名前	説明
アドレスの アロケーション	静的、 または、 DHCP	アドミンネットワークで DHCP を使用すると、ノードアドレスの中央管理が可能になり、また、IP アドレスに囚われない構成情報の伝達もできます。他のセグメントでは、VM 中心のネットワーク管理サービスとの衝突を避けるため、静的アドレスを使用しています。
ドメイン名	DNS	ノードには、自分自身、他ノード、アドミン、クライアントの名前を解決する能力が必要です。クラウド DNS サーバを使用すると、外部への依存を排除することができます。最終的には、クラウドに大量の DNS 処理が発生することになるため、ドメイン内で名前を制御する能力が必要です。
時間の同期	NTP	システムは、通信に対して証明書を発行するため、ほんのわずかな時間差でも、問題のトラブルシューティングが困難になります。
ネットワーク アクセス	要塞ホスト	推奨： ネットワークの分割を作成（または解決）する場合、要塞ホストを構成すれば、アドミンネットワーク（プロダクション）へのアクセスを制限したり、プロダクションネットワーク（機密ラボ）へのアクセス権を許可したりすることができます。
ネットワークの インストール	PXE	推奨： わずかな情報を、メディアから多数のサーバ上にインストールすることは実用的ではないため、ラボのインストールに限らず、PXE は必要です。
アウトバウンド （発信） ルーティング	SMTP	推奨： ほとんどのクラウドコンポーネントは、警告するときやアカウントの作成時に電子メールを送信します。電子メールを送信することができないと、ハングアップやエラーを引き起こす原因となるため、電子メールのルーティングを検討するようお勧めします。

プロビジョニング

ハイパースケール最大の課題は、当然のことながら、システムをオンラインにする（または、プロビジョニングする）ときに、また、導入後パッチレベルを維持するときに、膨大な反復作業が必要になることです。特に、OpenStack のような、新機能やパッチが随時登場する動的なプロジェクトでは、このアップデート作業が一層煩雑になります。現にデルのクラウド開発ラボでは、週一回のペースで、全システムの再構築をスケジュールしています。

また、インストールに遅れが出ぬよう、Puppet や Chef といったデプロイメントツールの習得にも投資しています。デルのクラウドオートメーションは、アドミンに Chef サーバを使用し、ノードイメージには Chef クライアントが含まれています。PXE を使用してノードに OS をロードしたら、次に Chef クライアントがそのノード固有の構成をサーバから取得します。構成スクリプト（Chef（シェフ）の専門用語では「レシピ」または「クックブック（料理本）」）は、正しいパッケージをインストールするだけでなく、当該ノードに必要なカスタマイズ構成とデータファイルもロードします。たとえば、Swift データノードには、然るべきリング構成ファイルを渡す必要があります。

真剣にクラウドの立ち上げをお考えなら、相互接続システムとしてのデプロイメントオートメーションを理解してください。私達は、この記述を「メタ構成」と呼んでいます。運用チームは、各 Swift リングにどのドライブを割り当て、各スケジューラにどの Nova ノードを関連付けるのか、熟考したうえで決定する必要があります。デルのクラウドインストーラは、トライアルシステムをシンプル化するため、お客様固有のインフラに基づいた推奨事項を提示します。しかし、クラウドはそれぞれ固有なため、最終的にはお客様が然るべき時間をかけて、インフラのフォルトゾーンと依存関係をマッピングする必要があります。

モニタリング

ノードのプロビジョニングが完了したら、運用チームはシステムの稼動を維持しなければなりません。数百台ものノードと数千個ものスピンドルを稼働する環境では、障害や性能の競合など、日常的な出来事です。もちろん、クラウドソフトウェアは設計に障害対応が組み込まれているため、重大な事態を回避しますが、それでも運用チームはエラーを見つけ、修理しなければなりません。エッジ障害がパニックを引き起こすことは稀ですが、利用可能な容量が劣化することから、収益は打撃を受けてしまいます。優れたクラウド設計とするには、計画的（パッチの適用）および計画外（障害）の停止を見越した対応策が必要です。

このニーズを満たすには、クラウドインフラのパフォーマンスとヘルスを両方モニタリングするシステムの設置が欠かせません。この件は広く理解されているため、競争の激しい市場でもあります。既に運用中のインフラがある場合、その既存システムを活用するようにしてください。デルの自動化された OpenStack ラボシステムでは、自動化したデプロイメントスクリプトにオープンソースツールの Nagios（ヘルス）と Ganglia（パフォーマンス）を組み込んでいます。私達がクラウドのテストを始めたとき、これらの機能がすぐに利用できると非常に便利であることを実感しました。

基盤が整ったら： 次は OpenStack の準備

これまでは、ハイパースケールガーデンの苗床の準備を中心に説明してきました。フォルトゾーン、サーバ、ネットワーク、運用インフラのすべてが、クラウドサービスの豊かな実りに貢献します。

これらの準備作業が完了したら、次は、クラウドをブートアップして、クラウドソフトウェアのインストールを始める番です。

以降では、OpenStack クラウドのインストール前に検討すべき設計項目について説明します。OpenStack の完全なインストール手順は本書の対象外となるため割愛しますが、ここではスタートポイントを示すことで、お客様がクラウド導入の次の段階に進められるようにします。

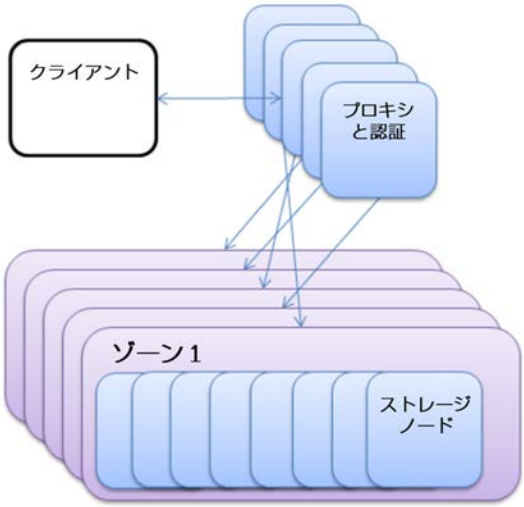
お急ぎですか？

デルは、クラウドの立ち上げ工程を自動化し、本書に記載された各 OpenStack コンポーネントをインストールするツールを開発しました。

これらのツールは、デル の OpenStack スタートキットの一環としてお客様に提供する予定です。

Object Storage（Swift）の導入

OpenStack のストレージコンポーネントである Swift は、非常に大規模なデータを低コストで保存するよう設計されています。簡単に説明すると、Swift とはデータセンターレベルの RAID システムであり、冗長性が個々のノード単位ではなく、フォルトゾーン単位にまで繰り上がったもの、と考えられます。このように Swift は、アクセスを確保するためにデータのコピーをフォルトゾーン間にわたって分散させるため、導入環境には少なくとも 3 つのフォルトゾーン（Swift 用語では「リング」）を作成する必要があります。



Swift には、「プロキシ」と「ストレージ」という 2 種類のサーバロール（役割）があります。

プロキシノードは、Swift API と認証サービスを提供します。これらはネットワーク集中型の Web サーバです。

ストレージノードは、プロキシによって更新・取得されるデータのコピーを維持します。これらはストレージ集中型のサーバです。

Swift 用のネットワークは非常にフラットです。プロキシは、すべてのストレージノードを通してデータを取得するため、すべてのストレージフォルトゾーンにアクセスできる必要があります。ストレージノードは、各フォルトゾーンと通信してデータのコピーを分配します。

Swift の導入を計画するときは、その拡張や移行が比較的容易であることを頭に入れておいてください。拡張時にノードを追加すると、Swift はその新しい容量でバランスを取りますし、移行が必要になったら単に離脱ノードを取り外すだけで、Swift が「失われた」コピーを新しいノードに複製してくれます。

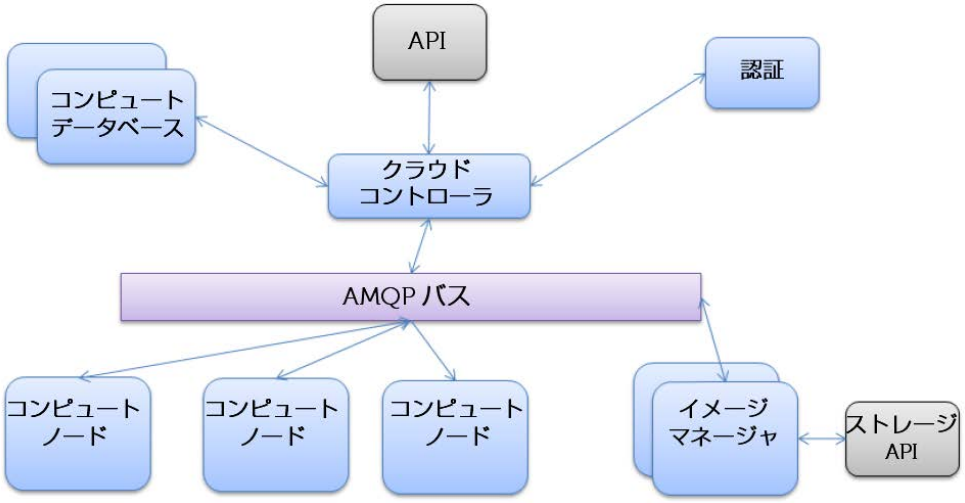
Swift の主な機能は、安価な長期ストレージを提供することにあるため、高速なオブジェクトまたはブロックストレージ機器は意図していません。これらの機能には、それぞれに特化したストレージサービス、たとえば、SQL、NoSQL、SAN、NAS、AMQP などが必要になることもあります。Swift を、OpenStack Glance や CDN （コンテンツ配信ネットワーク）などのキャッシュと共に、フロントエンドに設置することもできます。

OpenStack の設計策定時に助言をしたい方は、定例の「OpenStack Design Summit」に是非ご参加ください。同サミットのディスカッションに「青写真（Blueprint）」を提出して貢献することが奨励されています。

Compute（Nova）の導入

OpenStack Compute の「Nova」は複雑なうえ、急速に進化しているため、オンラインから入手できる最新資料を確認するよう強くお勧めします。

Nova の最も基本的なレベルでは、ユーザ向けにリソース（VM）を調整する管理＆オーケストレーションレイヤがあります。このレイヤは、複数の API（Amazon、Rackspace/OpenStack）とハイパバイザー（現在、KVM、XenServer、Hyper-V）をサポートし（VMware は近日対応予定）、メッセージバス（AMQP）を使用するリソースレイヤのコンピュートノードと調整を図ります。



「プラグ可能なスケジューラ」というのが、Nova の VM オーケストレーションの根底にあるロジックです。Nova には、「Scheduler」という名のコンポーネントが含まれる予定で、これが、VM をシステム内のどこに配置するのか決定します。Scheduler の仕事は、基本的な負荷分散を実行することです。Scheduler のオープン版は、最も利用されている RAM アルゴリズムか、ラウンドロビン（順繰り）方式を採用すると見られています。各ホストは、その固有の要件に応じて、Scheduler を拡張します。現在の設計ガイダンスでは、1 つの Scheduler ごとに約 100 ノードのリソースクラスタを管理するよう提唱しています（ラックあたり 36 ノードなら、108 ノードにマッピングされます）。Scheduler は、クラスタ間の調整を図ります。

その他のサービス

OpenStack は、ハイパースケールクラウドにインフラ基盤を提供しますが、トータルソリューションではありません。クラウドを運用するには、その目的に応じて、コンポーネントを追加する必要があります。コンポーネントを追加することで、たとえば、ソフトウェア開発者のサポート、内蔵システムとの統合、事前構築済みテンプレートの提供、顧客の操作能力の拡張などが可能になります。

以下に、検討できるコンポーネント例を示します。

- アプリケーションサポートコンポーネント： たとえば、構造化データベース（SQL）などのデータストレージサービス、テーブルストレージ（NoSQL）、キューイングサービス（AMQP）、コンテンツ配信ネットワーク（CDN）、アプリケーションプログラミングプラットフォーム（PaaS）などがあります。
- インテグレーション： 請求、認証、VPN トンネリングとの統合は、いずれも、顧客が内部システムに接続するのに役立ちます。
- 事前構築済みのテンプレートと、OVF（オープン仮想化フォーマット）や同種のテクノロジーを使ったイメージのアップロードは、相互運用性が高まりますし、顧客は他のクラウドからの作業を再使用できるようになります。
- オペレーションサービス： ロードバランサ、ファイアウォール、セキュリティサービス、バックアップ、アクセスモニタリング、ログ収集などを提供して、顧客側から運用の負担を取り除きます。これは顧客にとって多大なメリットとなりますし、プロバイダ側は「規模の経済」が活かせます。

OpenStack が提供する基盤以上に規模を拡張するチャンスは、無数にあります。オープンなクラウドインフラプラットフォームに投資することで、サービスとパートナーのエコシステムを拡張することができますし、共有プラットフォームを構築すれば、重複作業が削減されます。また、大規模なエコシステムを整備すれば、技術革新と投資が促進され、難問にも対処できるようになります。

まとめ

超大規模データセンターを設計するときは、運用上の課題をまったく別の角度から捉える必要があります。大量のリソースは管理に固有の困難が伴いますが、ローカルな冗長構成に頼るのではなく、広くリソースを分散させることで、この問題を解決することができます。

論理構成は、物理レイアウトと同様に重要です。シンプル化から逆行するすべての小さな要因が、大規模な環境では、極端な複雑化を助長してしまいます。したがって、自動化し、モニタリングする手法を見つけてください。

この強力なクラウドプラットフォームを迅速に評価するため、デルは、購入してすぐに利用できるソリューション開発に投資してきました。クラウド向けに最適化された業界随一のデルハードウェアに、デルのクラウドインストールオートメーションを組み合わせることで、お客様は、洗練されたインフラソリューションを確実に構築することができます。

デルは OpenStack コミュニティに積極的に貢献しています。それは、お客様にオープンな API と実用的なクラウド運用能力を提供し、経済的なインフラを実現する潜在能力が、OpenStack にはあると信じているからです。

参照先

Dell と OpenStack の詳細： www.dell.com/openstack