WINDOWS AZURE の紹介

本ホワイトペーパーは「Introducing the Windows Azure Platform」David Chappell 著の翻訳です。原文はこちらをご参照ください：
(http://go.microsoft.com/?linkid=9682907)

DAVID CHAPPELL

2010 年 10 月

SPONSORED BY MICROSOFT CORPORATION

目次

WINDOWS AZURE の概要

クラウド コンピューティングの時代が到来しました。インターネット経由でアクセスできるデータ センター内のマシンでアプリケーションを実行し、データを保存することに、多くのメリットが期待されています。ただし、アプリケーションは実行場所にかかわらず、いずれかのプラットフォーム上に構築されます。組織のデータ センター内で実行されるようなオンプレミスのアプリケーションの場合、こうしたプラットフォームは通常、オペレーティング システムおよびいずれかのデータ保存手段 (および、場合によってはその他の多くの要素) で構成されます。クラウド内で実行されるアプリケーションにも、これと同様の基盤が必要です。

Windows Azure の目標は、こうした基盤を提供することです。Windows Azure Platform の一部である Windows Azure は、クラウド内で各種アプリケーションを実行し、データを保存するための基盤です。図 1 に、この概念図を示します。

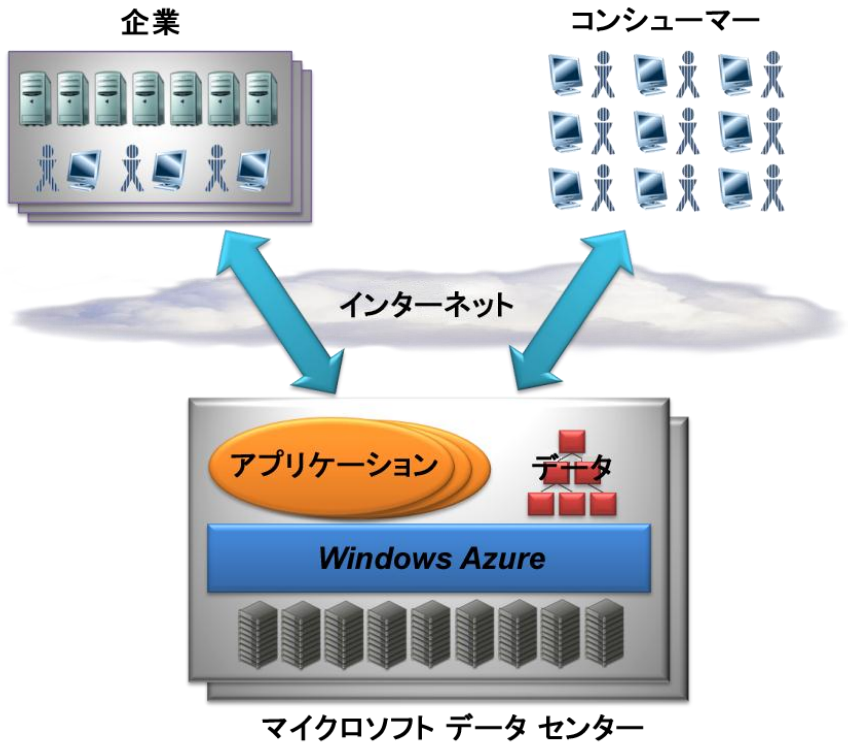


図 1: マイクロソフトのデータ センター内で実行され、インターネット経由でアクセスされる Windows Azure アプリケーション

Windows Azure では現在、マイクロソフトが提供したソフトウェアをユーザーが各自のコンピューターに自分でインストールして実行することはありません。ユーザーは Windows Azure をサービスとして利用し、インターネット経由でアクセスできる、マイクロソフトが保有するマシン上でアプリケーションを実行し、データを保存します。こうしたアプリケーションでは、企業やコンシューマーのいずれか、または両方にサービスを提供する場合があります。以下は、Windows Azure 上に構築できるアプリケーションの種類の一例です。

- 独立系ソフトウェア ベンダー (ISV) は、ビジネス ユーザーを対象とするアプリケーションを開発することがあります。これは多くの場合、"サービスとしてのソフトウェア (SaaS)" と呼ばれるアプローチです。Windows Azure では、設計目的の 1 つにマイクロソフト製 SaaS アプリケーションのサポートが含ま

れているため、ISV 各社は Windows Azure を基盤としてさまざまなビジネス向けクラウド ソフトウェアを開発することができます。

- ISV は、ビジネス向け以外に、コンシューマーを対象とする SaaS アプリケーションも開発することができます。Windows Azure は非常にスケーラブルなソフトウェアをサポートするよう設計されているため、大規模なコンシューマー市場への展開を計画している企業にとっては、新しいアプリケーションのプラットフォームとして最適な選択肢といえます。

- 大規模な企業では、Windows Azure を使用して、自社の社員が使用するアプリケーションを構築し、実行できます。この場合は、大規模なコンシューマー向けアプリケーションが不要になる可能性があるだけでなく、Windows Azure の優れた信頼性と管理性によるメリットも得られます。

各種のクラウド アプリケーションおよびデータをサポートするため、Windows Azure には 図 2 に示す 5 つのコンポーネントが含まれます。

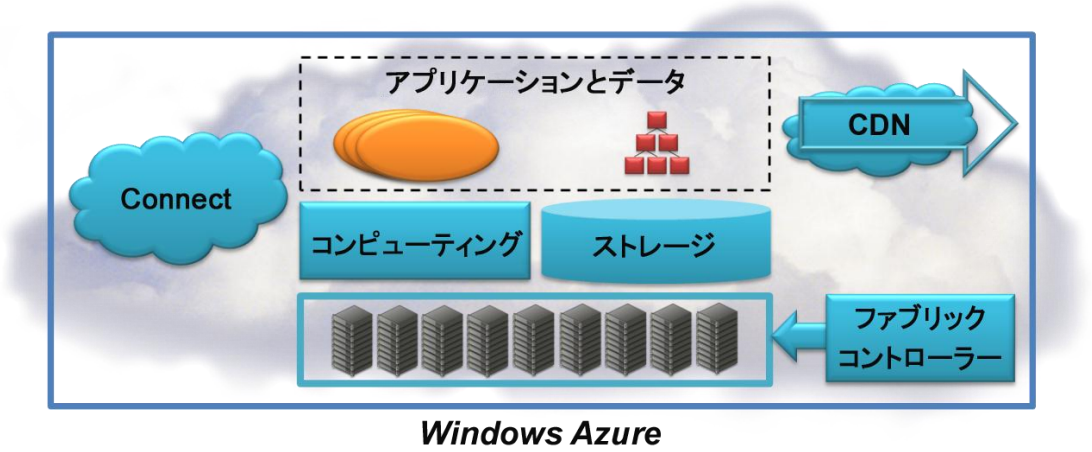


図 2: Windows Azure の 5 大コンポーネント - コンピューティング、ストレージ、ファブリック コントローラー、CDN、Connect

各コンポーネントの概要は次のとおりです。

- コンピューティング: クラウド内で各種アプリケーションを実行します。こうしたアプリケーションは Windows Server 環境とほぼ環境で実行されますが、Windows Azure プログラミング モデルは、オンプレミスの Windows Server モデルとまったく同じというわけではありません。

- ストレージ: クラウド内にバイナリ データおよび構造化データを保存します。

- ファブリック コントローラー: アプリケーションの展開、管理、監視を実行します。また、プラットフォーム全体のシステム ソフトウェアの更新もファブリック コントローラーによって行われます。

- コンテンツ配信ネットワーク (CDN): Windows Azure ストレージ内のバイナリ データに対するグローバルなアクセスを、対象のデータのキャッシュ コピーを世界中に保持することで高速化します。

- Connect: 社内コンピューターと Windows Azure アプリケーションとの間で、IP レベルの接続を確立できます。

以降では、それぞれのテクノロジについて紹介します。

コンピューティング

Windows Azure コンピューティングでは、多種多様なアプリケーションを実行できます。ただし、実行する処理の内容にかかわらず、アプリケーションは 1 つ以上の "ロール" として実装する必要があります。Windows Azure は通常、各ロールの複数の "インスタンス" を実行し、ビルトインされた負荷分散機能を使用して、全インスタンスに要求を分散します。図 3 に、このしくみを示します。

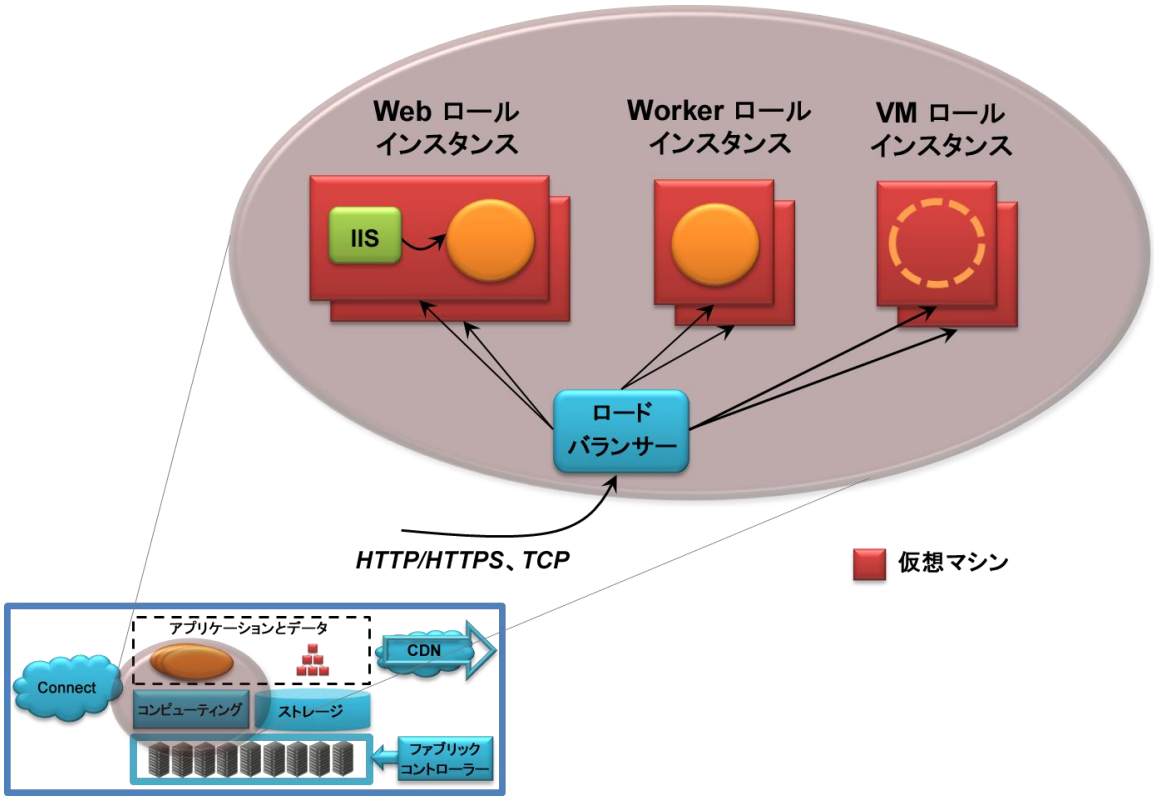


図 3: 複数の Web ロール インスタンス、Worker ロール インスタンス、VM ロール インスタンスの組み合わせからなる Windows Azure アプリケーションの実行

現在のバージョンの Windows Azure で、開発者が選択できるロールは次の 3 種類です。

- Web ロールでは主に、Web ベース アプリケーションの作成の利便性が重視されています。各 Web ロール インスタンス内には、あらかじめ構成されたインターネット インフォメーション サービス (IIS) 7 が含まれるため、ASP.NET や Windows Communication Foundation (WCF) などの各種 Web テクノロジを使用したアプリケーションを円滑に作成できます。開発者はまた、ネイティブ コードのアプリケーションを作成することもできますが、その際、.NET Framework を使用する必要はありません。すなわち、PHP や Java といった、マイクロソフト以外のテクノロジをインストールし、実行することができます。

- Worker ロールは、さまざまな Windows ベースのコードを実行するために設計されています。Web ロールとの最も大きな違いは、Worker ロール内には構成済みの IIS が含まれないため、Worker ロールが実行するコードが IIS によってホストされない点です。Worker ロールは、たとえばシミュレーションの実行、ビデオの処理の実行、またはその他ほぼすべての用途に使用できます。ユーザーが Web ロールを通じてアプリケーションとのやり取りを行い、その後 Worker ロールにタスクを渡して処理するのが一般的です。前述のとおり、開発者は、.NET Framework を使用するか、Windows 上で動作する他のソフトウェア (マイクロソフト以外のテクノロジを含む) を使用するかを、自由に選択できます。

- 各 VM ロールは、ユーザーが指定した Windows Server 2008 R2 イメージを実行します。VM ロールは特に、オンプレミスの Windows Server アプリケーションを Windows Azure に移行するような場合に役立ちます。

アプリケーションを Windows Azure にアップロードするには、Windows Azure ポータルを使用できます。アプリケーションと同時に構成情報が送信され、実行する各ロールのインスタンス数がプラットフォームに伝達されます。その情報に基づいて、Windows Azure ファブリック コントローラーがそれぞれのインスタンスに対応する仮想マシン (VM) を作成し、VM 内で適切なロールのコードが実行されます。

図 3 で示すように、アプリケーションのユーザーからの要求には、HTTP、HTTPS、および TCP といった各種のプロトコルを使用できます。ただし、受信されたこれらの要求は、ロールの全インスタンスへと負荷分散されます。ロード バランサーが特定のロール インスタンスとアフィニティを構築することは許可されていない (スティッキー セッションがサポートされない) ため、同一のユーザーによる複数の要求は、必ずしもロールの同じインスタンスに送られるとは限りません。これは、Windows Azure の各ロール インスタンス自体が、要求間で自身のセッション状態を維持できないことを意味しています。代わりに、クライアント固有のセッション情報はすべて、Windows Azure ストレージに書き込まれるか、SQL Azure (Windows Azure Platform の別のコンポーネント) に格納されるか、他のいずれかの方法によって外部で維持されます。

開発者は、複数の Web ロール インスタンス、Worker ロール インスタンス、および VM ロール インスタンスを任意に組み合わせて Windows Azure アプリケーションを作成できます。アプリケーションの負荷が高くなった場合は、Windows Azure ポータルから、アプリケーションで使用するいずれかのロールのインスタンスを追加するよう要求できます。負荷が減少した場合は、実行するインスタンス数を減らすことができます。また、Windows Azure が提供する API を利用すれば、このような処理をすべてプログラムで実行できます。実行インスタンス数の変更に手動での操作は不要ですが、負荷に応じて、プラットフォーム側でアプリケーションの規模が自動的に調整されるわけではありません。

Windows Azure アプリケーションの開発には、すべての Windows アプリケーションと同じ言語およびツールを使用できます。たとえば、ASP.NET と Visual Basic、または WCF と C# を使用して、Web ロールを記述できます。同様に、こうした .NET 言語の 1 つを使用して Worker ロールを作成し、.NET Framework を使用せずに C++ から直接実行するほか、Java を使用することもできます。また、Windows Azure には Visual Studio 向けのアドインが用意されていますが、この開発環境を必ずしも使用する必要はありません。たとえば、PHP をインストールしている開発者は、別のツールを使用してアプリケーションを記述することもできます。

Windows Azure アプリケーションの監視およびデバッグのため、各インスタンスでログ記録 API を呼び出し、アプリケーション全体の共通のログに情報を書き込むことができます。また、アプリケーションのパフォーマンス カウンターの収集、CPU 使用率の測定、エラー発生時のクラッシュ ダンプの保存などを行うようにシステムを構成することも可能です。この情報は Windows Azure ストレージに保存され、開発者はこの情報を検証するコードを自由に作成できます。たとえば、Worker ロール インスタンスで 1 時間に 3 回のクラッシュが発生した場合に、アプリケーションの管理者に電子メールが送信されるようなカスタム コードを作成できます。

コードの実行機能はクラウド プラットフォームの基本条件ですが、それだけでは十分ではありません。アプリケーションには永続的なストレージも必要です。次に説明する Windows Azure ストレージ サービスは、このニーズへの対応を目的としています。

ストレージ

アプリケーションでは多種多様な方法でデータが処理されます。そのため、Windows Azure ストレージ サービスには複数のオプションが用意されています。図 4 に、これらのオプションを示します。

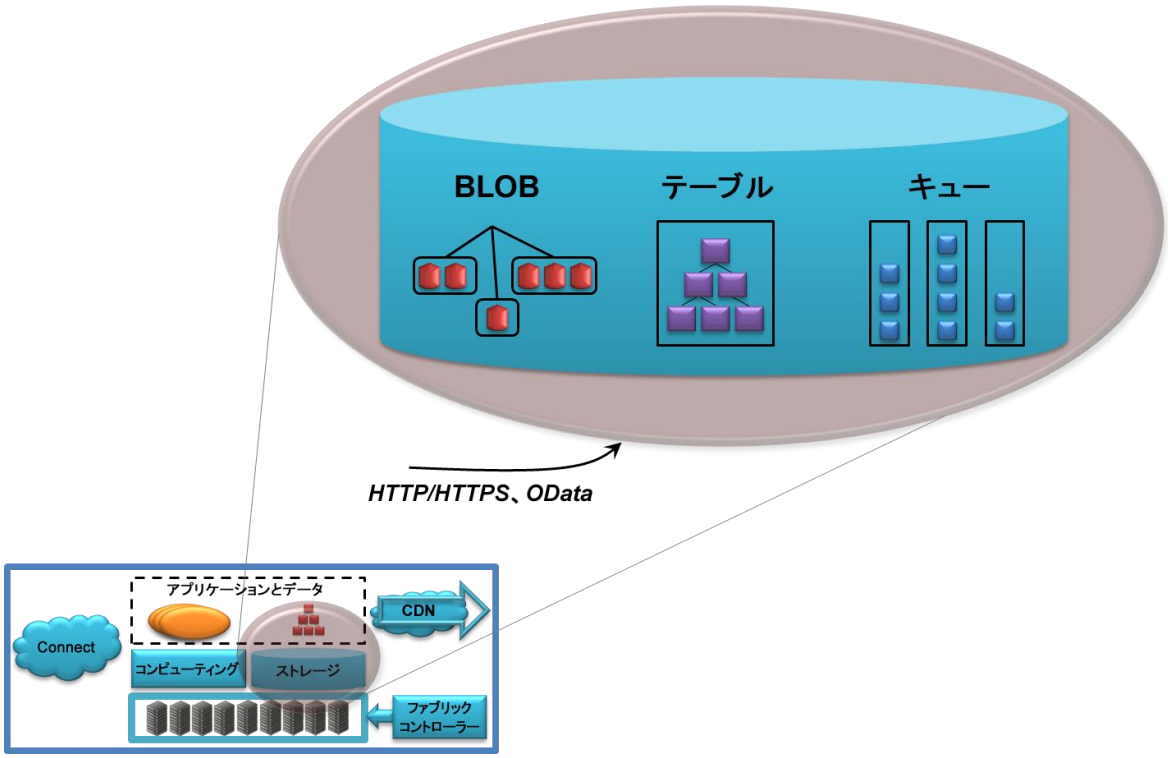


図 4: Windows Azure ストレージが提供する BLOB、テーブル、キュー

Windows Azure ストレージにデータを保存する最も簡単な方法は、BLOB を使用することです。BLOB にはバイナリ データが格納されます。図 4 からわかるように、1 つ以上の BLOB が各 "コンテナー" に収容される、というシンプルな階層構造が存在しています。BLOB は最大 1 テラバイトの大容量をサポートし、JPEG 形式の写真の撮影場所や MP3 ファイルのアーティスト情報などをメタデータとして関連付けることもできます。BLOB は "Windows Azure ドライブ" の基盤ストレージとしての役割も果たします。Windows Azure ドライブは、Windows Azure のロール インスタンスから、ローカルの NTFS ファイル システムと同様に永続的ストレージへとアクセスできるようにするメカニズムです。

BLOB が最適な状況もありますが、構造がシンプルすぎるため、適さない状況もあります。Windows Azure ストレージでは、より詳細なレベルでのアプリケーションのデータ処理を可能にするため、テーブルを提供しています。名前で誤解されやすいのですが、これはリレーショナル テーブルではありません。各テーブル内のデータは、実際には "プロパティ" を含む "エンティティ" のグループ内に格納されています。アプリケーションがテーブルのデータを照会する際は、SQL ではなく、OData によって定義された規則を使用します。このアプローチでは、多数のマシンにデータを分散して規模を拡大することにより、標準的なリレーショナル データベースよりもはるかに効率的にストレージをスケールアウトできます。実際に、1 つの Windows Azure テーブルを何十億というエンティティで構成し、数テラバイトのデータを保持することも可能です。

BLOB とテーブルはいずれも、データの保存とアクセスに重点を置いています。Windows Azure ストレージの 3 つ目のオプションであるキューは、それとは目的がまったく異なります。キューは主に、Web ロール インスタンスと Worker ロール インスタンスとの間で、非同期的な通信手段を提供する役割を果たします。たとえば、Windows Azure の Web ロールによって実装された Web インターフェイス経由で、ユーザーが多くのコンピューター処理を要するタスクの実行を要求したとします。この要求を受信した Web ロール インスタンスは、実行すべき作業を記述したメッセージをキューに書き込むことができます。すると、このキューを待機している Worker ロール インスタンスによってメッセージが読み込まれ、指定したタスクが実行されます。結果は他のキューから返される場合や、別のいずれかの方法で処理される場合があります。

データの格納方法 (BLOB、テーブル、またはキュー) にかかわらず、Windows Azure ストレージで保持されるすべての情報は 3 回複製されます。このレプリケーションによってフォールト トレランスが実現され、1 つのコピーが失われても致命的な結果にはなりません。このシステムは強力な一貫性を備えているため、自身が書き込んだばかりのデータをアプリケーションがすぐに読み込んでも、直前に書き込んだ内容を確実に再取得できます。また、Windows Azure では、全世界の同じ地域に属する別のデータ センターにあるデータのすべてについて、バックアップ コピーが保持されています。メイン コピーを保持しているデータ センターが利用不能になった場合や物理的に破壊された場合でも、このバックアップには引き続きアクセスできます。

Windows Azure ストレージには、Windows Azure アプリケーション、オンプレミスのアプリケーション、ホスティング サービスや他のクラウド プラットフォームで実行されているアプリケーションからアクセスできます。いずれの場合も、Windows Azure の 3 つのストレージ形式ではすべて、図 4 で示すように REST の規則に沿ってデータが識別され、提供されます。各 BLOB、テーブル、キューはすべて URI を使用して命名され、標準の HTTP 処理を通じてアクセスされます。.NET クライアントでは、アクセスに Windows Azure で提供されるライブラリを使用できますが、必須ではありません。アプリケーションから RAW HTTP を呼び出すこともできます。

BLOB、テーブル、キューを使用して Windows Azure アプリケーションを作成すると、確実に利便性が高まります。リレーショナル ストレージを使用する必要のあるアプリケーションでは、代わりに Windows Azure Platform の別のコンポーネントである SQL Azure を使用できます。Windows Azure (または、それ以外の場所) で実行しているアプリケーションでこのテクノロジを使用すると、クラウド内のリレーショナル ストレージに、使い慣れた SQL ベースでアクセスできます。

ファブリック コントローラー

すべての Windows Azure アプリケーションと、Windows Azure ストレージ内のすべてのデータは、マイクロソフトのデータ センターにあります。このデータ センターでは、Windows Azure 専用の一連のマシンとそれらで実行されるソフトウェアが共に、ファブリック コントローラーによって管理されています。図 5 に、この概念図を示します。

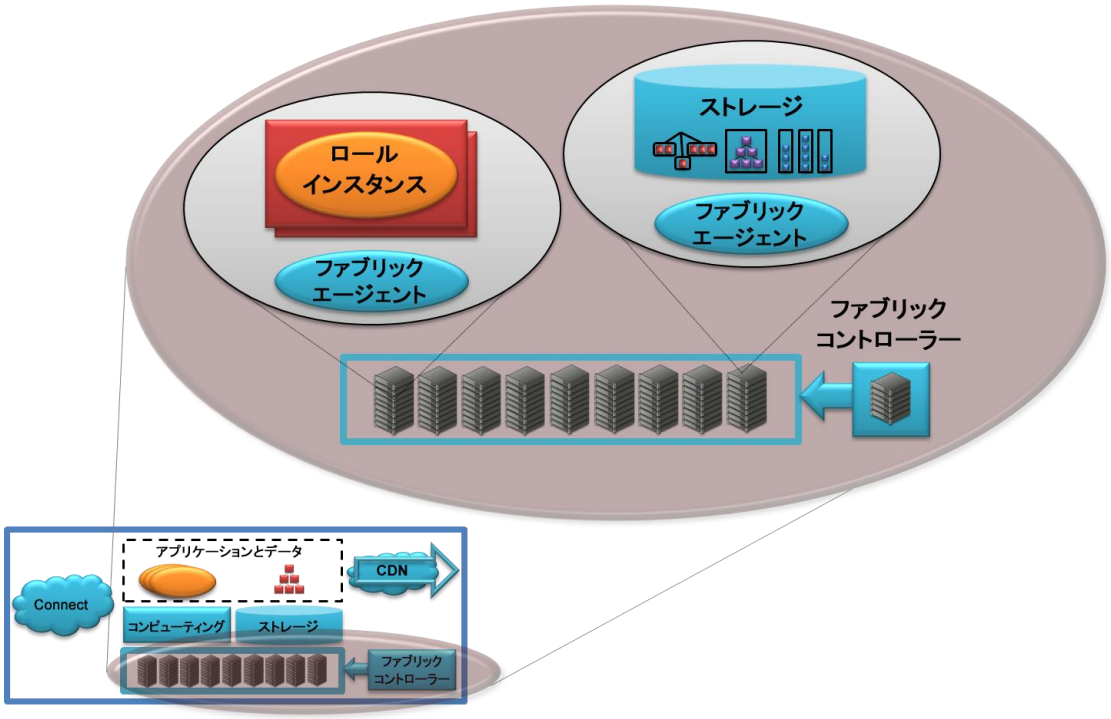


図 5: ファブリック エージェントを介して Windows Azure アプリケーションと通信するファブリック コントローラー

ファブリック コントローラーとは、複数のマシンからなるグループの全体にわたって複製された分散アプリケーションです。ファブリック コントローラーは、コンピューター、スイッチ、ロード バランサーなど、環境内の全リソースを所有します。すべてのコンピューター上の "ファブリック エージェント" と通信可能なため、ファブリック内の全 Windows Azure アプリケーションも認識できます (興味深いのは、ファブリック コントローラーが Windows Azure ストレージを単なるアプリケーションの 1 つとして認識する点です。そのため、データの管理とレプリケーションの詳細はコントローラーからは確認できません)。

こうした広範な対象を認識することで、ファブリック コントローラーは多くの有益な機能を提供するようになります。たとえば、実行中のアプリケーションをすべて監視し、最新の状況を常に把握できます。さらに、新しいアプリケーションを実行する場所を決定し、物理サーバーを選択してハードウェアの利用率を最適化できます。こうしたファブリック コントローラーの処理に不可欠なのが、各 Windows Azure アプリケーションと共にアップロードされた構成情報です。このファイルには、Web ロール インスタンスの数や Worker ロール インスタンスの数など、対象のアプリケーションの要件が XML ベースで記述されています。ファブリック コントローラーは新しいアプリケーションを展開すると、この構成ファイルを使用して作成すべき VM の数を決定します。

それらの VM を作成後、ファブリック コントローラーは各 VM を監視します。たとえば、アプリケーションが 5 つの Web ロール インスタンスを必要とする場合、そのうちの 1 つがダウンしても、ファブリック コントローラーが自動的に新しいインスタンスを起動します。同様に、VM を実行しているマシンがダウンした場合も、ファブリック コントローラーが別のマシン上で対象のロールの新しいインスタンスを起動し、必要に応じてロード バランサーを再設定して、この新しい VM を参照します。

現在 Windows Azure には、選択可能な VM のサイズが 5 種類あります。選択できるオプションは、次のとおりです。

- 極小 (XS): シングルコアの 1.0 GHz CPU、768 MB のメモリ、20 GB のインスタンス ストレージ

- 小 (S): シングル コアの 1.6 GHz CPU、1.75 GB のメモリ、225 GB のインスタンス ストレージ
- 中 (M): デュアル コアの 1.6 GHz CPU、3.5 GB のメモリ、490 GB のインスタンス ストレージ
- 大 (L): 4 コアの 1.6 GHz CPU、7 GB のメモリ、1,000 GB のインスタンス ストレージ
- 特大 (XL): 8 コアの 1.6 GHz CPU、14 GB のメモリ、2,040 GB のインスタンス ストレージ

極小 (XS) のインスタンスは、プロセッサ コアを他の極小インスタンスと共有します。ただし、それ以外のサイズではすべて、それぞれのインスタンスに 1 つ以上の専用コアが用意されます。そのため、アプリケーションのパフォーマンスは予測可能であり、インスタンスの実行可能時間が制限されるという不確定要素がありません。たとえば、Web ロール インスタンスで必要なだけ時間をかけてユーザーからの要求を処理し、Worker ロール インスタンスで円周率を百万桁まで計算することも可能です。

Web ロールおよび Worker ロールについては (VM ロールは除く)、ファブリック コントローラーが各インスタンスのオペレーティング システムの管理も行います。管理内容には、OS へのパッチの適用やその他のシステム ソフトウェアの更新などが含まれます。そのため、開発者はアプリケーションの作成のみに集中でき、プラットフォーム自体の管理に気を配る必要がありません。ただし、ファブリック コントローラーが常に、各ロールについて最低 2 つのインスタンスが実行されていることを前提としている点は、理解しておく必要があります。そのため、ソフトウェアを更新する際、ファブリック コントローラーはアプリケーション全体を停止させることなく、インスタンスの 1 つをシャットダウンします。これ以外の理由も考慮し、Windows Azure のすべてのロールに関して、インスタンスを 1 つしか実行しないのは一般的にお勧めできません。

コンテンツ配信ネットワーク

BLOB の一般的な用途の 1 つに、さまざまな場所からアクセスできる情報の格納が挙げられます。世界中の Flash、Silverlight、または HTML 5 のクライアントにビデオを提供するアプリケーションがその例です。こうしたシナリオでのパフォーマンス向上を目的として、Windows Azure はコンテンツ配信ネットワーク (CDN) を提供しています。CDN では、BLOB のデータを使用するクライアントの近くのサービス拠点に、BLOB のコピーが保持されます。図 6 に、この概念図を示します。

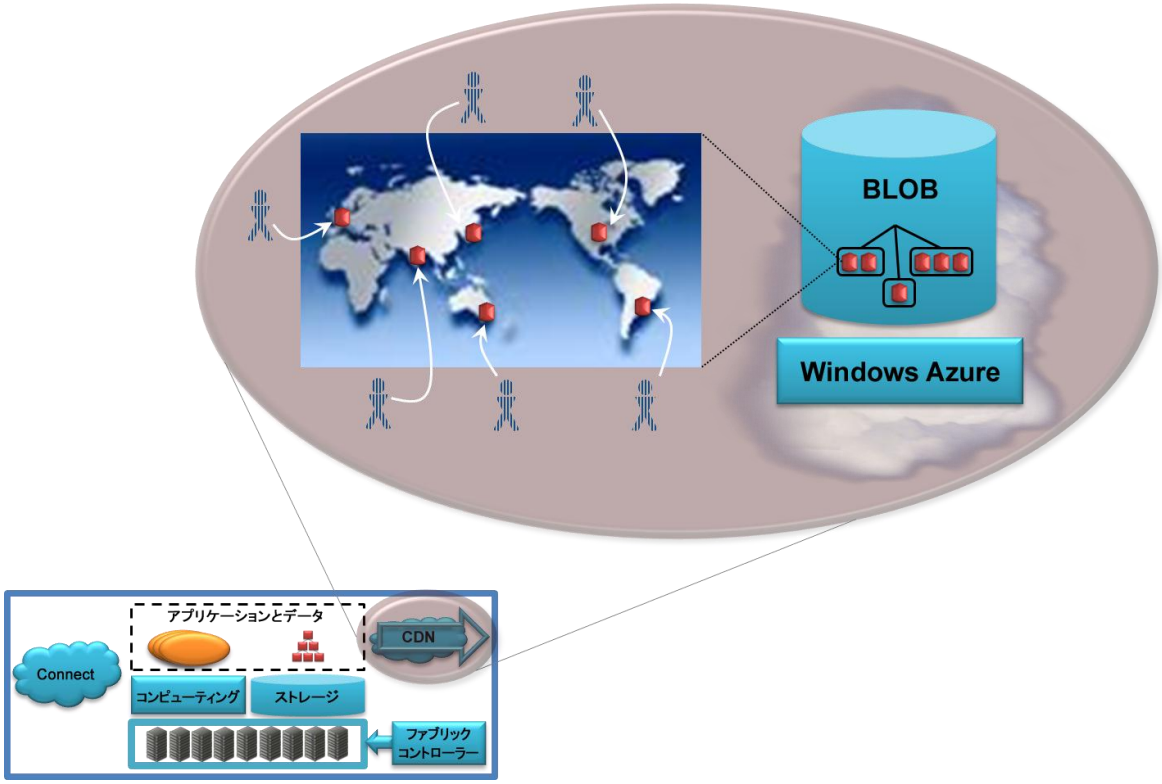


図 6: Windows Azure CDN は世界中の BLOB のコピーをキャッシュすることにより、対象の情報へのより迅速なアクセスを実現

概念としてこの図は正確ですが、実際の CDN が図とまったく同じだとは考えないでください。Windows Azure CDN は、図で示しているよりもずっと多くのグローバルなキャッシング サイトを備えています。ユーザーが初めて特定の BLOB にアクセスすると、CDN はそのユーザーと地理的に近い場所に、対象の BLOB のコピーを格納します。この BLOB に次回アクセスする際は、より遠くにある元のサイトからではなく、キャッシュからコンテンツが配信されます。

たとえば、Windows Azure を使用して、その日のスポーツ イベントのビデオを遠隔地の視聴者に配信するとします。特定のビデオにアクセスする最初のユーザーは、まだ目的の BLOB が近くのサービス拠点にキャッシュされていないため、CDN の恩恵を得ることはできません。ただし、同じ地域にいる他のユーザーはすべて、キャッシュ コピーによってビデオの読み込み時間が短縮されるため、より快適にビデオを視聴できます。

CONNECT

マイクロソフトのクラウド内でアプリケーションを実行するのは便利です。ただし、組織内で使用しているアプリケーションやデータがすぐになくなることもないでしょう。ここで必要となるのが、オンプレミス環境と Windows Azure との効果的な接続機能です。

Windows Azure Connect は、この要件を満たすために設計されており、Windows Azure アプリケーションと、マイクロソフト クラウドの外部で実行されるマシンとの間で IP レベルの接続を提供することにより、こうした組み合わせを簡単に利用できるようにします。図 7 に、この概念図を示します。

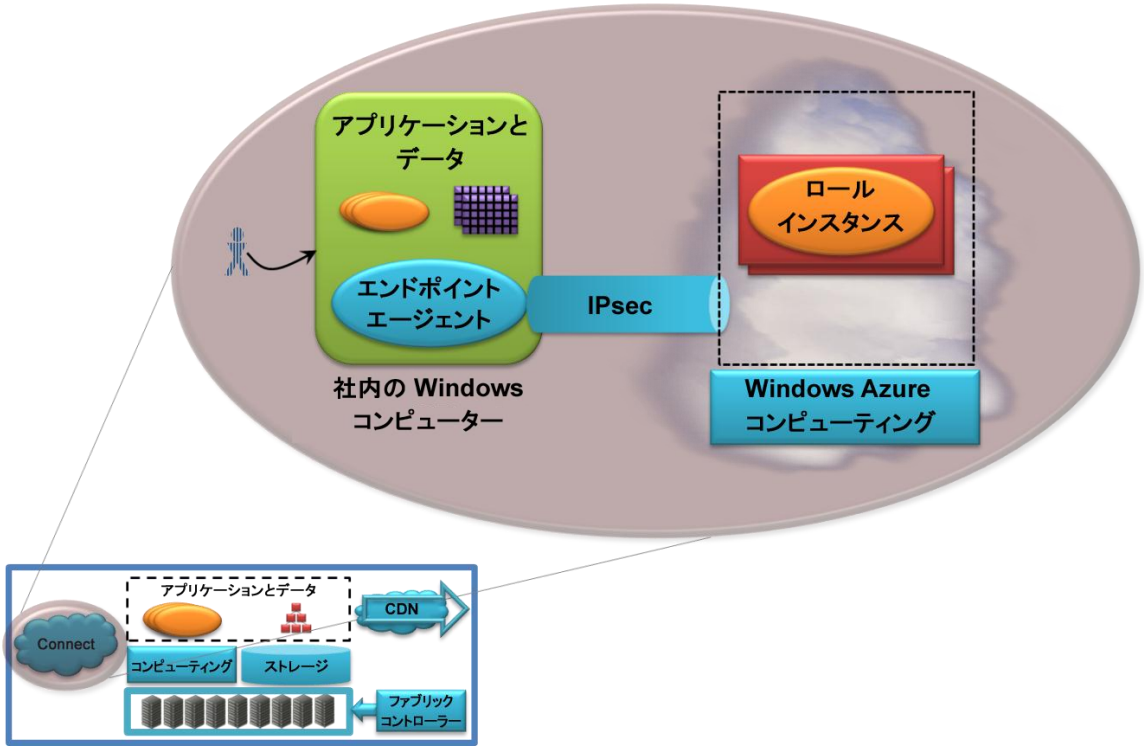


図 7: Windows Azure Connect により、オンプレミスの Windows マシンと Windows Azure アプリケーションとの間で IP レベルの接続を実現

図で示すように、Windows Azure Connect を使用するには、Windows Azure アプリケーションに接続する社内コンピューターのそれぞれに "エンドポイント エージェント" をインストールする必要があります (このテクノロジは IPv6 に依存しているため、現在利用可能なエンドポイント エージェントは、Windows Server 2008、Windows Server 2008 R2、Windows Vista、および Windows 7 のみに対応しています)。また、Windows Azure アプリケーションが Windows Azure Connect と連動するように設定する必要もあります。この作業を終えると、エージェントが IPsec を使用して、対象のアプリケーション内の特定のロールと通信できるようになります。

これが本格的な仮想プライベート ネットワーク (VPN) ではない点に注意してください。マイクロソフトは、いずれ VPN 機能を提供する計画があることを発表していますが、Windows Azure Connect は、それよりもシンプルなソリューションです。たとえば、Windows Azure Connect のセットアップに、組織のネットワーク管理者への連絡は必要ありません。必要となるのは、ローカル コンピューターへのエンドポイント エージェントのインストールのみです。このアプローチでは、IPsec の構成に関する潜在的な複雑性が意識されることもありません。構成作業は Windows Azure Connect によって実行されます。

このテクノロジのセットアップが完了すると、Windows Azure アプリケーション内のロールを、同一 IP ネットワーク内の社内マシンと同様に扱えるようになります。これにより、次のことが可能になります。

- Windows Azure アプリケーションから、オンプレミスのデータベースに直接アクセスできます。たとえば、ある組織で、ASP.NET を使用して構築された既存の Windows Server アプリケーションを Windows Azure Web ロールに移行するとします。このアプリケーションが使用するデータベースを引き続きオンプレミスに置く必要がある場合、Windows Azure Connect の接続機能を使用すれば、Windows Azure で実行されるようになったアプリケーションから、オンプレミスのデータベースに従来どおりアクセスできます。この場合、接続文字列の変更も必要ありません。

- Windows Azure アプリケーションを、オンプレミス環境のドメインに参加させることができます。これにより、オンプレミスのユーザーによるクラウド アプリケーションへのシングル サインオンが可能になります。また、このアプリケーションでは、既存の Active Directory のアカウントとグループを使用してアクセス コントロールを実施できます。

重要なのは、クラウドを現在のオンプレミス環境と十分に適応させることです。Windows Azure Connect によって IP レベルの直接接続を可能にすることで、Windows Azure アプリケーションではこの要件をより簡単に達成できます。

WINDOWS AZURE の活用: シナリオ

Windows Azure のコンポーネントを理解することは重要ですが、それだけでは不十分です。このプラットフォームの機能を把握するには、実際の使用例を見ていくことが最善の方法です。そのため、このセクションでは、スケーラブルな Web アプリケーションの作成、並列処理アプリケーションの作成、バックグラウンド処理を実行する Web アプリケーションの作成、リレーショナル データを使用する Web アプリケーションの作成、リレーショナル データを使用するオンプレミスの Web アプリケーションの移行、およびオンプレミス/ホスティング アプリケーションからのクラウド ストレージの利用といった各種の Windows Azure シナリオについて検討します。

スケーラブルな WEB アプリケーションの作成

ある企業が、インターネット経由でアクセスできる Web アプリケーションを作成しようとしています。現在、こうしたアプリケーションはオンプレミスのデータ センターやホスティング サービスを利用して実行するのが一般的ですが、Windows Azure などのクラウド プラットフォームは、多くの面でより優れた選択肢といえます。たとえば、アプリケーションで多数の同時ユーザーに対応する必要がある場合には、その点を明確に考慮して設計されたプラットフォーム上にアプリケーションを構築する方が賢明です。Windows Azure では、本質的にスケーラブルなアプリケーションとデータがサポートされるため、従来の Web テクノロジよりもはるかに大きな負荷を処理できます。

また、長期間にわたって低い利用率が継続し、ごくたまに利用率が急増するような、アプリケーションの負荷が大幅に変動する場合を考えてみましょう。オンライン チケット サイトや、不定期に重大ニュースが掲載されるニュース ビデオ サイトのほか、一日のある一定の時間帯だけ利用率が高まるアプリケーションなどがその例です。この種のアプリケーションを従来のデータ センターで実行するには、たとえシステムの大半がほとんどの時間使用されないとしても、ピーク時に対応できる数のマシンが常に必要となります。一方、Windows Azure 上にアプリケーションを構築した場合、アプリケーションを運用する企業は必要なときだけインスタンス数を増やし、それ以外のときには減らすことができます。Windows Azure の利用料金は従量制 (各インスタンスの使用時間単位で課金) のため、ほとんど使用しないマシンをいくつも保持するよりもコストを低く抑えられるでしょう。

Windows Azure 上でスケーラビリティの高い Web アプリケーションを作成するには、Web ロールとテーブルを使用します。図 8 は、そのしくみを簡単に図解したものです。

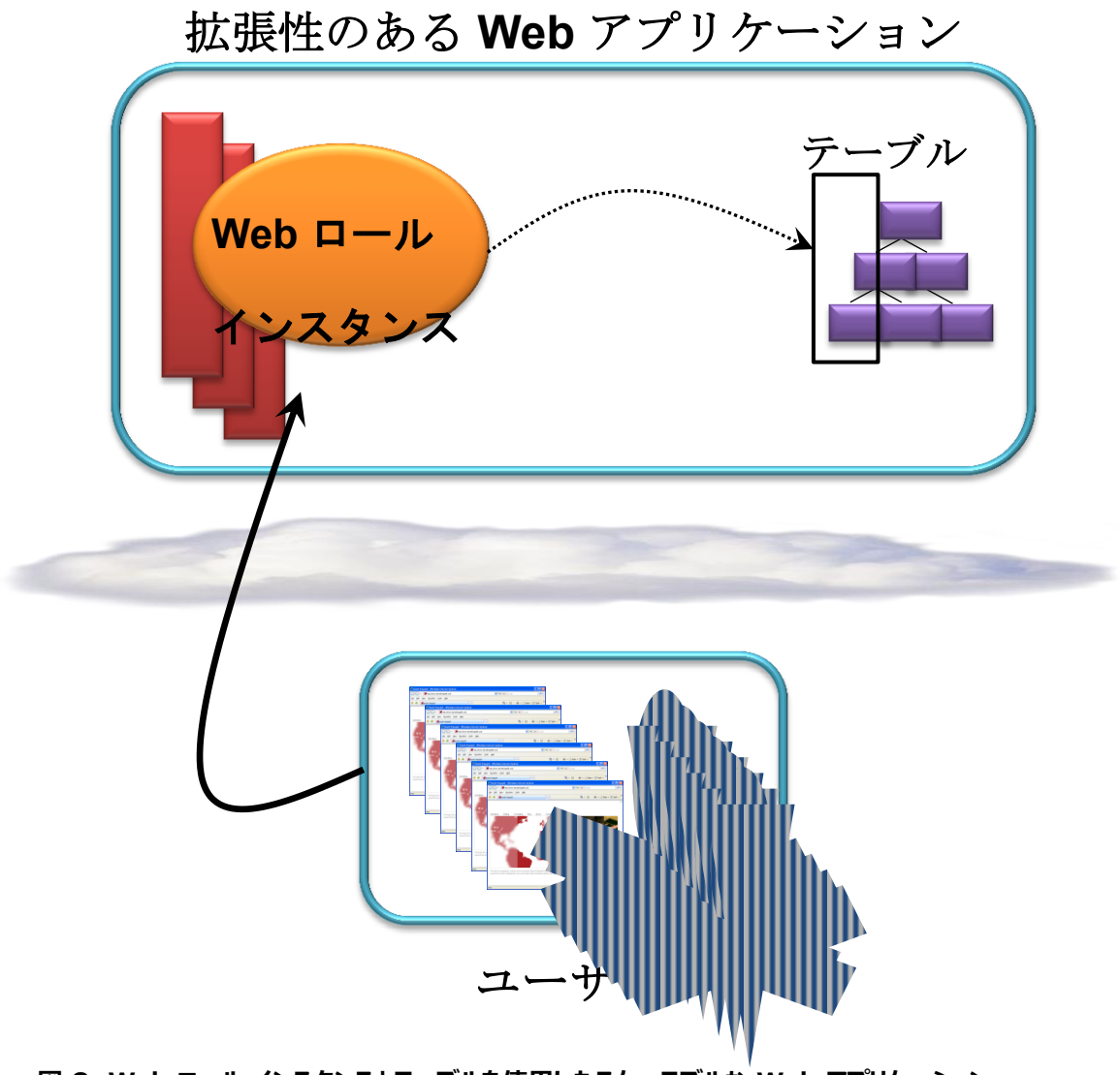


図 8: Web ロール インスタンスとテーブルを使用したスケーラブルな Web アプリケーション

この例では、クライアントがブラウザーなので、アプリケーション ロジックの実装には ASP.NET などの Web テクノロジが使用されます。WCF を使用して、REST ベースまたは SOAP ベースの Web サービスを提供するスケーラブルな Web アプリケーションを作成し、対象のサービスを Silverlight クライアントなどから呼び出すことも可能です。いずれの場合も、実行する Web ロール インスタンスの数を開発者が指定すると、その数の VM が Windows Azure ファブリック コントローラーによって作成されます。前述のように、ファブリック コントローラーはインスタンスの監視も行うため、必要な数のインスタンスが常に提供されます。このアプリケーションでは、データ ストレージに Windows Azure ストレージのテーブルを使用することにより、大量のデータを扱うことができるスケールアウト ストレージが実現されます。

並列処理アプリケーションの作成

スケーラブルな Web アプリケーションは便利ですが、Windows Azure の能力が発揮されるシナリオはこれだけではありません。ここでは、一時的に高いコンピューティング能力を必要とする企業の、並列処理アプリケーションについて考えます。銀行での財務モデリング、映画の特殊効果スタジオでのレンダリング処理、製薬会社での新薬の開発など、その事例は数多く挙げられます。こうした一時的なニーズに対応するため、コンピューターの大規模なクラスターを保持することもできますが、これには多くのコストもかかります。一方、

Windows Azure は必要なリソースを必要なときに提供するため、オンデマンドでコンピューティング クラスターを調達できます。

この種のアプリケーションは Worker ロールを使用することで作成できます。選択肢はそれだけではありませんが、一般に並列処理アプリケーションは大規模なデータセットを使用するため、Windows Azure BLOB ならそのデータセットを格納できます。図 9 に、こうしたアプリケーションの構造を示します。

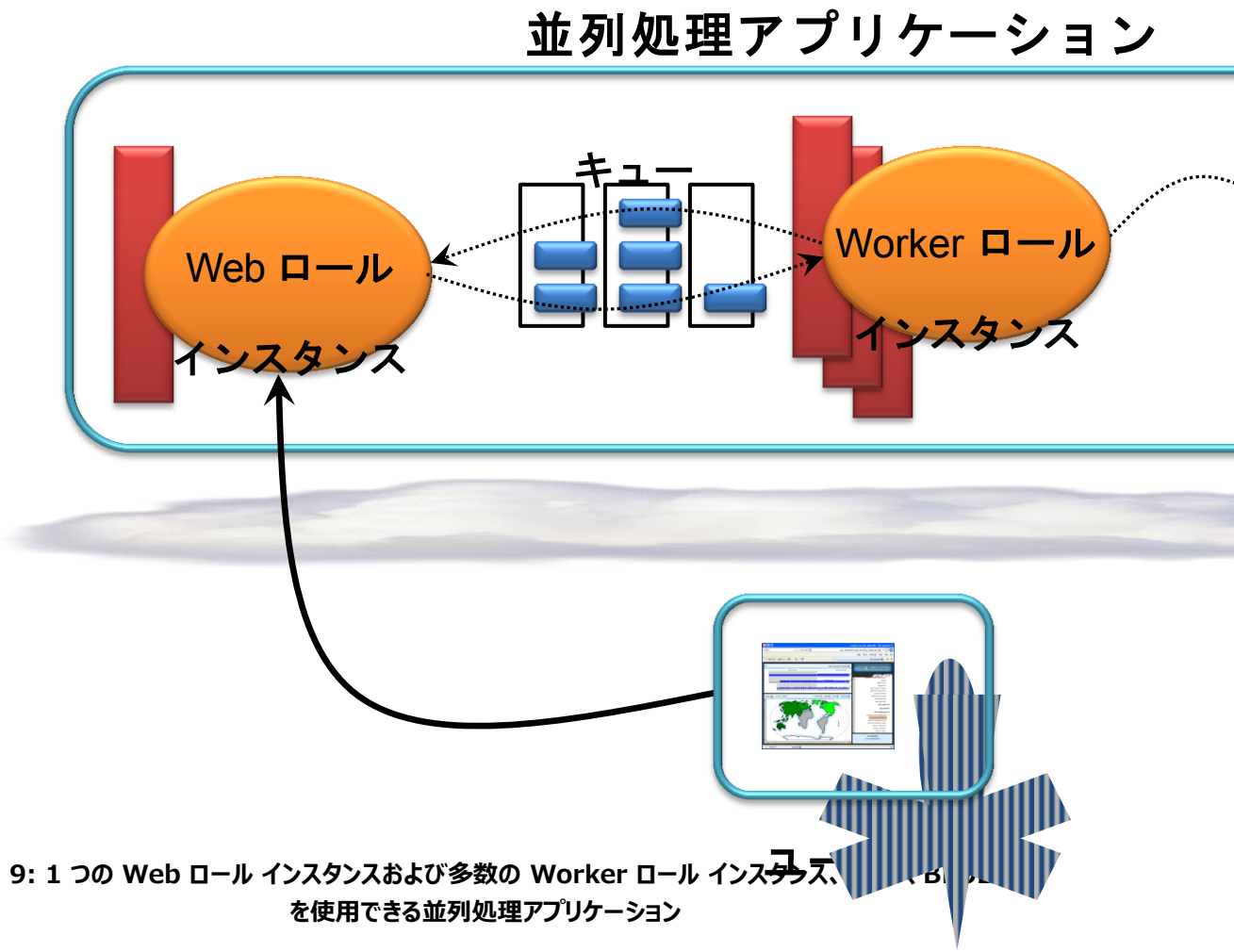


図 9: 1 つの Web ロール インスタンスおよび多数の Worker ロール インスタンス、
を使用できる並列処理アプリケーション

このシナリオでは、複数の Worker ロール インスタンスを同時に実行し、それぞれが BLOB データを使用することにより、並列処理を実現しています。Windows Azure ではインスタンスを実行できる時間に制限がないため、各インスタンスが任意の作業量を実行できます。アプリケーションとユーザーとのやり取りには、単一の Web ロール インスタンスを利用します。ユーザーはこのインターフェイスを通じて、実行する Worker インスタンス数の決定、インスタンスの起動と停止、結果へのアクセスなどを行えます。Web ロール インスタンスと Worker ロール インスタンスとの間の通信には、Windows Azure ストレージのキューを利用します。

クラウドで利用できる膨大な量の処理能力を考えれば、この新しいアプローチはハイパフォーマンス コンピューティングを一変させる可能性があります。たとえば、Microsoft Windows HPC Server では、Windows Azure の Worker ロール インスタンスをオンプレミスの物理サーバーと併用し、または物理サーバーの代わりに使用して、コンピューティング クラスターを作成できるようになりました。ただ、この新しいコンピューティング能力の提供手段を利用することはできますが、メリットが得られるのはごく一部の状況に限られます。

バックグラウンド処理を実行するスケーラブルな WEB アプリケーションの作成

現在構築されるアプリケーションの大多数は、ブラウザー インターフェイスを使用しているといっても差し支えないでしょう。ブラウザーの要求を受理し、応答するだけのアプリケーションは便利な反面、限界もあります。Web 経由でアクセス可能なソフトウェアでは、アプリケーションの要求/応答とは独立した、バックグラウンド処理が必要になる状況が多くあります。

たとえば、ビデオ共有のための Web アプリケーションについて考えてみます。このアプリケーションは、おそらくは多数の同時ユーザーによる複数のブラウザー要求を受理する必要があります。こうした要求の中には新しいビデオをアップロードし、後からアクセスできるように、それぞれに対して処理と保存が必要となる場合があります。この処理が完了するまでユーザーを待たせるのは非生産的です。それを回避するには、ブラウザー要求を受理するアプリケーション側で、この処理を実行するバックグラウンド タスクを実行できる必要があります。

このシナリオには、Web ロールと Worker ロールを同時に使用することで対応できます。図 10 は、この種のアプリケーションのしくみを示しています。

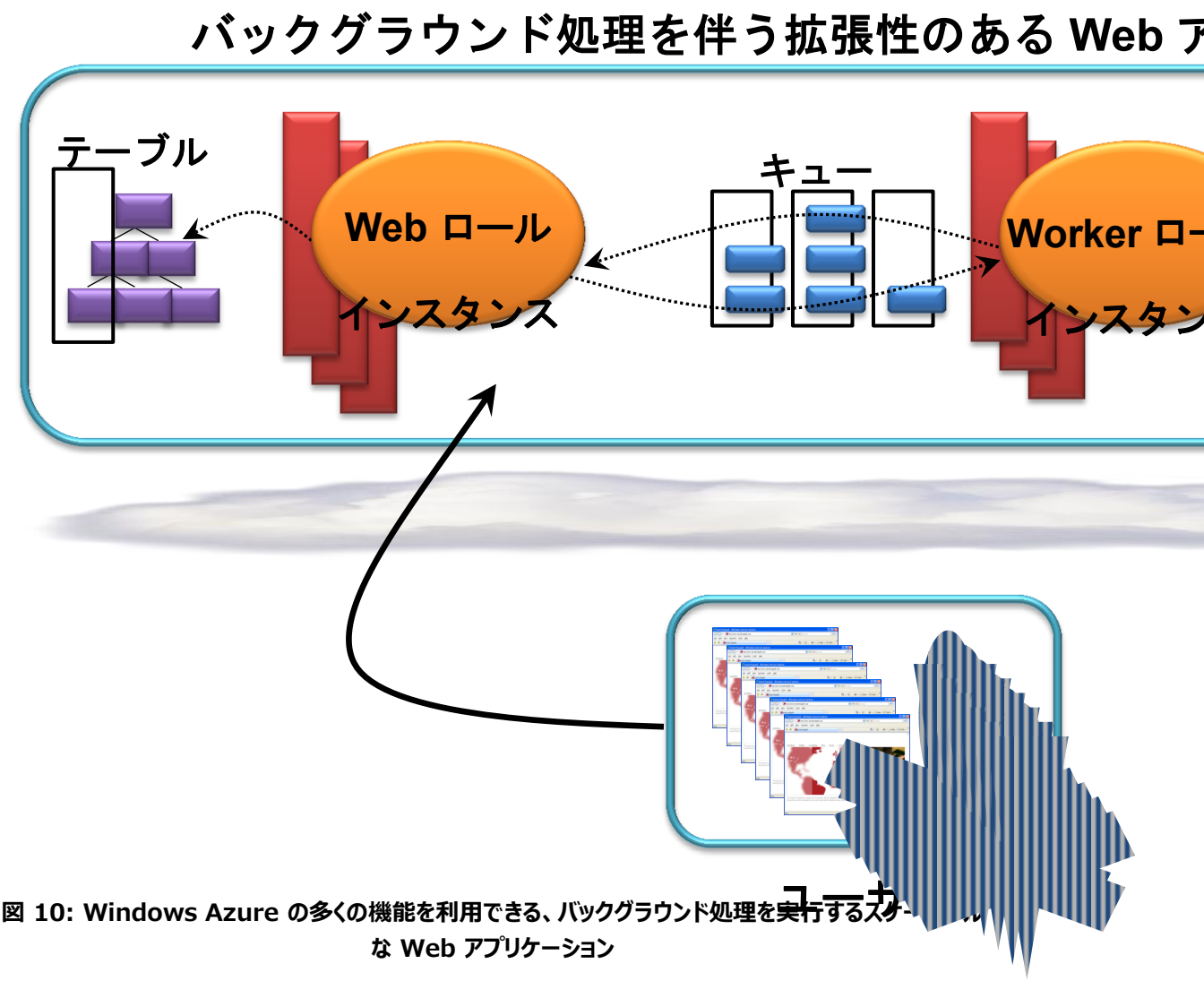


図 10: Windows Azure の多くの機能を利用できる、バックグラウンド処理を実行するスケーラブルな Web アプリケーション

前述のスケーラブルな Web アプリケーションと同様、このアプリケーションでも複数の Web ロール インスタンスを使用してユーザーの要求を処理します。多数の同時ユーザーに対応するため、このアプリケーションではテーブルを使用してプロファイル情報を保存しています。バックグラウンド処理は Worker ロール インスタンスで

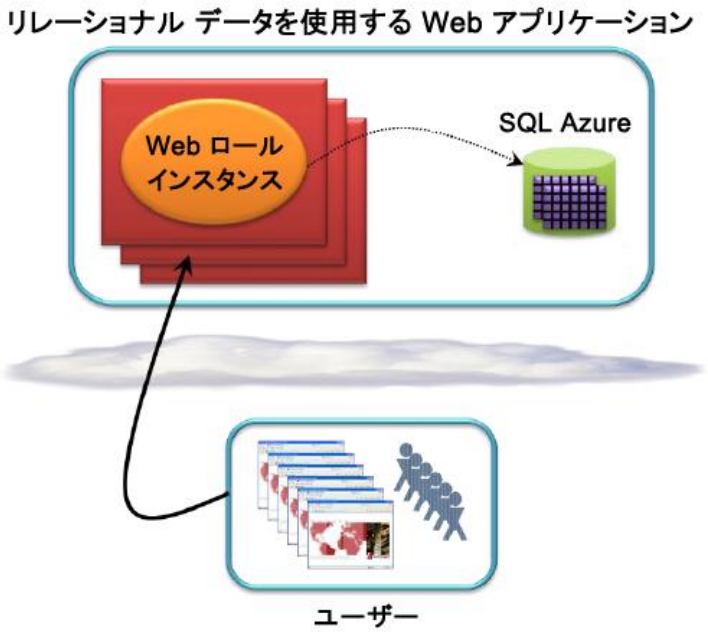
実行され、キュー経由でタスクを渡しています。この例の Worker ロール インスタンスは BLOB データを使用して処理を行っていますが、他のアプローチの利用も可能です。

この例では、Web ロール インスタンス、Worker ロール インスタンス、BLOB、テーブル、およびキューといった、Windows Azure が提供する基本機能の多くを組み合わせてアプリケーションを作成する方法を示しています。この図には示されていませんが、ビデオ共有アプリケーションでは、Windows Azure CDN を使用することでアクセス時間を短縮できます。アプリケーションによって、これらの機能がすべて必要になるとは限りませんが、この例のような複雑なシナリオに対応するには、利用可能な全機能を活用する必要があります。

リレーショナル データを使用する WEB アプリケーションの作成

Windows Azure の BLOB とテーブルは一部の状況には適していますが、リレーショナル データの方が適している場合も少なくありません。たとえば、ある企業が Windows Azure 上で、社員向けのアプリケーションを構築し、実行しようとしているとします。このアプリケーションは、使用期間が短期間である、または予測できないために、企業のデータ センター内のサーバーを割り当てるだけの価値がないことが考えられます。または、できるだけ早くアプリケーションの運用を開始する必要があるために、社内 IT 部門によってサーバーがプロビジョニングされるまで待てない可能性もあります。さらに、この企業では、Windows Azure でアプリケーションを運用する方が低コストでシンプルだと考えているのかもしれません。

理由は何であれ、このアプリケーションはおそらく、Windows Azure テーブルによって可能となるような大規模なものにする必要はありません。作成者はむしろ、既にノウハウのあるリレーショナル アプローチを使用し、



慣れ親しんだレポート作成ツールを完備したいと考えるでしょう。こうしたケースでは、図 11 に示すように、アプリケーションで Windows Azure と SQL Azure を併用できます。

図 11: SQL Azure を使用してリレーショナル データを操作できる Windows Azure アプリケーション

SQL Azure は管理されたクラウド サービスとして、SQL Server の機能の大半を提供します。各アプリケーションではデータベースの作成や SQL クエリの実行などが可能ですが、データベース システムやデータベー

スを実行するハードウェアの管理は、マイクロソフトが行うため必要ありません。SQL Azure には、オンプレミスの SQL Server と同様に、表形式データ ストリーム (TDS) プロトコルを使用してアクセスできます。そのため、Windows Azure アプリケーションは、Entity Framework および ADO.NET のような馴染みのあるメカニズムを使用してリレーショナル データにアクセスできます。また、SQL Azure はクラウド サービスであるため、使用量ベースで課金されます。

Windows Azure と SQL Azure は、それぞれが対応するオンプレミス型の製品のクラウド版にあたるため、この種のアプリケーションのコードやデータを両者間で非常に簡単に移行できます。たとえば、Windows Azure アプリケーションでは複数のインスタンスを実行できなければならない点など、両者はまったく同一なわけではありませんが、クラウド環境とオンプレミス環境はきわめて似ています。オンプレミスまたはクラウドのいずれにもコードとデータを配置できるアプリケーションを作成する場合には、この移植性の高さが活かされます。

リレーショナル データを使用するオンプレミスの WEB アプリケーションの移行

企業が Windows Azure 用に新たな Web アプリケーションを作成するのではなく、既存の Windows Server アプリケーションをこのクラウド プラットフォームに移行することを希望しているとしましょう。これを実現するアプローチの 1 つが、Windows Azure VM ロールの利用です。図 12 に示すように、その構造は前のケースとよく似ています。

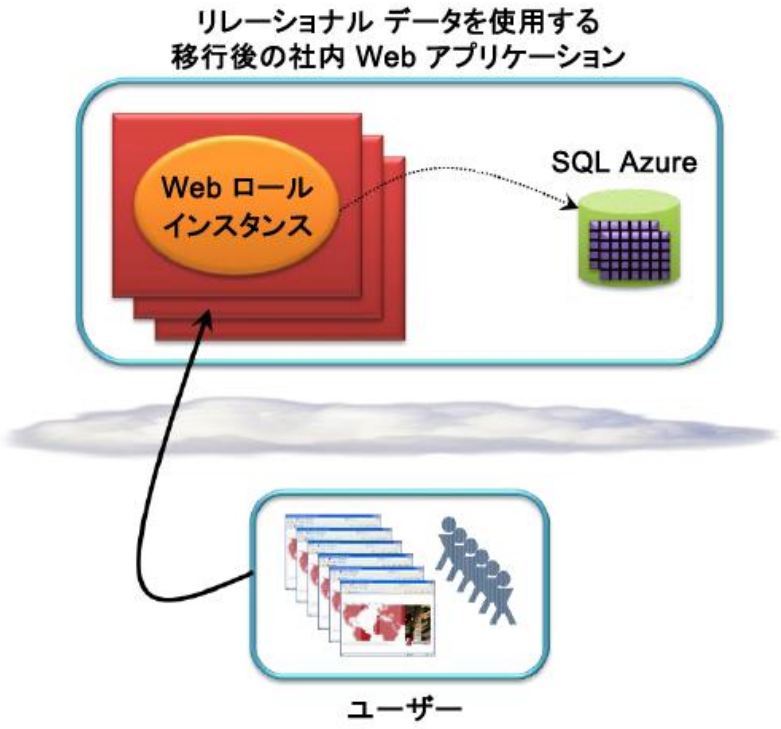


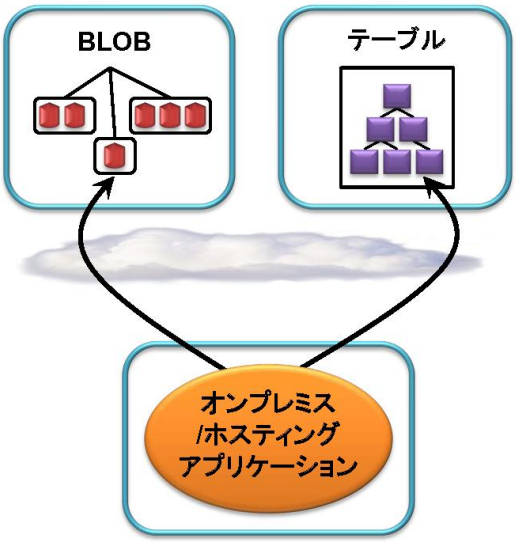
図 12: VM ロールと SQL Azure の使用によって実現される、一部のオンプレミスのアプリケーションの Windows Azure への移行

VM ロールを使用するには、Windows Server 2008 R2 を実行する社内のコンピューターから仮想ハード ディスク (VHD) を作成します。その後、このイメージを Windows Azure にアップロードすれば、VM ロール内で実行できます。図に示すように、アプリケーションからは SQL Azure 内のリレーショナル データにアクセスできます。別の選択肢として、データをオンプレミスに残し、先ほど説明した Windows Azure Connect 経由で直接そのデータにアクセスする方法があります。

VM ロールは便利です。ただし、Windows Server アプリケーションを Windows Azure に移行するには、単にアプリケーションを VHD にパッケージ化し、それを VM ロール内で実行すればよいわけではない点を理解しておいてください。まず、Windows Azure ファブリック コントローラーは、各ロールについて最低 2 つのインスタンスが常に実行されていることを前提としている点を覚えておいてください (実際、Windows Azure サービス レベル アグリーメント (SLA) の達成には、この条件が満たされている必要があります)。Windows Azure では、全ユーザーの要求による負荷がロールのインスタンス全体で分散される点も忘れてはなりません。移行するアプリケーションがこうした動作を前提に構築されている場合 (たとえば、負荷分散された Web ファーム内で既に運用されている場合) は、大きな変更を加えなくても Windows Azure 上でうまく機能します。ただし、単一のインスタンスでのみ実行されることを想定したアプリケーションの場合は、Windows Azure 上で正常に動作させるために、一定の再設計が必要となる可能性があります。

オンプレミス/ホスティング アプリケーションからのクラウド ストレージの使用

Windows Azure はさまざまな機能を備えていますが、アプリケーションではその中の 1 つの機能しか必要としない場合があります。その一例として、膨大な量のデータを保存する必要があるオンプレミス/ホスティング アプリケーションが挙げられます。たとえば、古い電子メールを保管する場合、企業はストレージのコストを抑えつつ、メールへのアクセス性も維持したいと考えます。ホスティング サービスを通じて運用しているニュース Web サイトで、大量のテキストやグラフィックス、ビデオ、およびユーザーのプロファイル情報を保存するための、



グローバルにアクセスできるスケーラブルな場所が必要な場合もあります。また、写真共有サイトで、信頼できるサード パーティを利用して、情報の保存に関する負担軽減を図る場合もあります。図 13 に示すように、Windows Azure ストレージはこれらのすべての状況に対応できます。

図 13: Windows Azure の BLOB またはテーブルを利用し、クラウド内にデータを保存するオンプレミス/ホスティング アプリケーション

前述のように、オンプレミス/ホスティング アプリケーションは Windows Azure ストレージに直接アクセスできます。このストレージは一般に、アクセス速度ではローカル ストレージよりも劣りますが、コストの低さ、スケーラビリティ、および信頼性ではより優れています。アプリケーションによっては、Windows Azure ストレージを優先する価値は十分にあります。また、図には示されていませんが、アプリケーションは SQL Azure を同じ方法で使用することができます。

WINDOWS AZURE の理解: 詳細

Windows Azure を理解するために、このプラットフォームの基本要素について確認し、そうした基本要素の一般的なシナリオでの適用例を見てきましたが、それだけではこのテクノロジを十分理解したとはいえません。このセクションでは、Windows Azure のより興味深い側面のいくつかを詳しく説明します。

WINDOWS AZURE アプリケーションの作成

開発者にとって、Windows Azure アプリケーションは、従来の Windows アプリケーションとほぼ同様に構築できます。このプラットフォームでは .NET アプリケーションと、アンマネージ コードを使用して構築されたアプリケーションの両方がサポートされるため、開発者は作成するプログラムに最適な方法を使用できます。開発を容易にするため、Visual Studio には、Windows Azure アプリケーションを作成するための各種プロジェクト テンプレートが用意されています。Visual Studio から Windows Azure に直接アプリケーションをアップロードすることも可能です。

クラウド環境とオンプレミス環境の明らかな違いは、Windows Azure アプリケーションがローカルで実行されない点にあります。この違いが、開発をより困難にする場合もあります。マイクロソフトはこうした問題を軽減するため、開発者のマシン上で Windows Azure 環境を実行できるよう、"開発ファブリック" を提供しています。

開発ファブリックは単一のデスクトップ マシンやサーバー マシン上で実行され、Web ロール、Worker ロール、VM ロール、および Windows Azure ストレージ オプションの 3 つすべてを完備した、クラウド上の Windows Azure 機能をエミュレートします。開発者は Windows Azure アプリケーションを構築して、開発ファブリックに展開し、実際のサービスとほぼ同じ方法で実行できます。たとえば、各ロールで実行する必要のあるインスタンス数を決定し、キューを使ってそれらのインスタンス間で通信を行うなど、実際の Windows Azure で可能な作業をほぼすべて実行できます 開発者はローカルでアプリケーションの開発とテストを行った後、コードとその構成情報をアップロードして実行できます。

作成方法を問わず、Windows Azure アプリケーションをクラウドで利用できるようにするには、通常 2 つの手順からなるプロセスを実行します。最初に、開発者はプラットフォームのステージング領域にアプリケーションをアップロードします。アプリケーションを稼働させる準備が整ったら、Windows Azure ポータルを使用して、実稼働の開始を要求します。このステージングと実稼働の切り替えにダウンタイムは発生しないため、ユーザーの業務に支障を与えることなく、実行中のアプリケーションを新バージョンにアップグレードできます。

ステージングされたアプリケーションには、<*GUID*>.cloudapp.net という形式の DNS 名が割り当てられます。<*GUID*> は、Windows Azure によって割り当てられた一意のグローバル識別子です。実稼働環境では、開発者が同じドメイン内の DNS 名 (myazureservice.cloudapp.net など) を選択します。マイクロソフトの cloudapp.net ドメインではなく、カスタム ドメインを使用する場合、アプリケーションの所有者は、標準の CNAME を使用して DNS エイリアスを作成できます。

外部からアプリケーションにアクセスできるようになると、いずれかの方法でユーザーの認証を行う必要が出てきます。そのため Windows Azure では、開発者が任意の HTTP ベースの認証メカニズムを使用できるようにしています。たとえば、ASP.NET アプリケーションでは、メンバーシップ プロバイダーを使用してアプリケーション固有のユーザー ID とパスワードを保存できるほか、Microsoft Windows Live ID サービスなどの他の方法を使用することもできます。Windows Azure アプリケーションでは、Windows Identity Foundation (WIF) を使用してクレームベース ID を実装してもかまいません。どの方法を選択するかはアプリケーションの作成者が決定できます。

コンピューティング サービスについての考察

ほとんどのテクノロジと同様に、Windows Azure コンピューティングは、最初にリリースされて以来進化しています。たとえば、Web ロールおよび Worker ロールのコードは当初、ユーザー モードでしか実行できませんでした。しかし現在は、どちらのロールでも特権を昇格させるためのオプションが用意されており、アプリケーションを管理モードで実行できます。これは COM コンポーネントをインストールする必要のあるアプリケーションなどで役に立つ機能であり、Windows Azure の最初のリリースでは多少問題のあった部分です。

現在、Web ロールまたは Worker ロールのインスタンスの実行は、必ずクリーンな状態で開始され、インスタンスを実行する VM 内の基盤となるオペレーティング システムには、Windows Azure で定義された標準のイメージが使用されます。つまり、ロールでソフトウェアをインストールする場合は必ず、毎回新しいインスタンスを作成する必要があります。これは、単一の COM コンポーネントを追加するような、シンプルなインストールでは問題になりません。一方、目的を達成するために、インスタンスがさまざまなソフトウェアをインストールする必要があるとします。新しいロール インスタンスを作成するたびにこれを行うと、非常に時間がかかるでしょう。

こうした遅延を回避するのが VM ロールの主な目的です。インスタンスを作成するたびにインストールを実行する必要はなく、必要なソフトウェアをすべて VHD にパッケージ化し、その VHD を使用して VM ロール インスタンスを作成できます。この方法により、昇格された特権で Web ロールや Worker ロールを使用するよりも、はるかに時間が短縮されます。また、VM ロールの使用は、インストール プロセス中に、Windows Azure でサポートされないいずれかの手動作業が必要となる場合にも適したソリューションです。

当初のバージョンの Windows Azure からもう 1 つ変更されたのは、このプラットフォームでリモート デスクトップ プロトコル (RDP) を介したアクセスがサポートされるようになった点です。これはデバッグなどに役立つ機能であり、開発者が特定のインスタンスに直接アクセスすることが可能になります。ただし、この機能は仮想デスクトップ インフラストラクチャ (VDI) には使用できません。これは Windows Azure が (少なくとも現在は)、こうしたシナリオをサポートするように設計されていないためです。

Windows Azure コンピューティングのその他の重要な機能は、このテクノロジが最初にリリースされたときから変わっていません。たとえば、Windows Azure では、開発者がアプリケーションを実行するデータ センターとデータを保存する場所を指定できます。また、特定のグループのアプリケーションやデータ (SQL Azure 内のデータを含む) を、すべて同じデータ センター内で運用するように指定することも可能です。マイクロソフトではまず、Windows Azure データ センターを米国、欧州、アジアに展開し、その後、他の地域にも展開する予定です。

ストレージ サービスについての考察

Windows Azure ストレージを使用するには、最初に "ストレージ アカウント" を作成する必要があります。アカウントの作成者には、このアカウント内の情報へのアクセスを制御するために秘密キーが提供されます。アプリケーションがこのストレージ アカウント内の情報 (BLOB、テーブル、キュー) に対して行う各要求には、この秘密キーで作成された署名が付加されます。つまり、認証はアカウント レベルで行われます (ただし、BLOB には後述する別の方法もあります)。Windows Azure ストレージでは、アクセス コントロール リストなど、ストレージ内のデータへのアクセス許可をきめ細かく制御する手段は提供されません。

BLOB

バイナリ ラージ オブジェクト (BLOB) は多くの場合、アプリケーションのすべてのニーズを満たします。ビデオ、オーディオ、アーカイブされた電子メール メッセージなど、格納するデータの種類を問わず、BLOB はアプリケー

ションでデータの保存/アクセスを実行する際の一般的な方法です。BLOB を使用するには、まず所定のストレージ アカウント内に 1 つ以上のコンテナーを作成します。作成したコンテナーには、それぞれ 1 つ以上の BLOB を収容できます。

特定の BLOB を識別するため、アプリケーションは次の形式の URI を割り当てることができます。

http://<*StorageAccount*>.blob.core.windows.net/<*Container*>/<*BlobName*>

<*StorageAccount*> は新しいストレージ アカウントが作成されたときに割り当てられる一意の識別子で、<*Container*> および <*BlobName*> は、特定のコンテナーの名前と、そのコンテナー内の BLOB の名前です。

BLOB には、次の 2 つの形式があります。

- ブロック BLOB: 各ブロック BLOB には、最大 200 GB のデータを格納できます。効率的に転送できるよう、ブロック BLOB は複数のブロックに分割されています。障害が発生した場合は直近のブロックから転送を再開すれば済み、BLOB 全体を再送する必要はありません。BLOB の構成ブロックがすべてアップロードされた後、BLOB 全体をまとめてコミットできます。

- ページ BLOB: 各ページ BLOB の最大容量は 1 TB です。ページ BLOB は 512 バイトのページに分割され、BLOB 内の各ページに対し、アプリケーションはランダムに読み書きを実行できます。

格納する BLOB の種類にかかわらず、コンテナーはプライベートまたはパブリックとして設定できます。プライベート コンテナー内の BLOB の場合、読み取り要求と書き込み要求はいずれも、BLOB のストレージ アカウントに対応するキーを使用して署名されている必要があります。パブリック コンテナー内の BLOB の場合は、書き込み要求のみ署名が必要となり、BLOB の読み取りは全アプリケーションに許可されます。このコンテナーは、ビデオや写真などの非構造化データを、インターネット上で一般公開する場合に便利です。現状で、Windows Azure CDN が対応するのは、パブリック BLOB コンテナーに格納されたデータのみです。

BLOB のもう 1 つの重要な機能として、Windows Azure ドライブをサポートするための役割があります。この役割を理解するにはまず、各種のロール インスタンスが自身のローカル ファイル システムに自由にアクセスできることを認識しておく必要があります。ただし既定では、これは永続的ストレージではなく、インスタンスがシャットダウンされると、VM およびそのローカル ストレージが削除されます。ところが、このインスタンスに Windows Azure ドライブをマウントすると、ページ BLOB をローカル ドライブと同様に使用し、NTFS ファイル システムを完備できます。ドライブへの書き込み処理は、基盤となる BLOB に即座に書き込み可能です。インスタンスの実行中以外は、このデータがページ BLOB に永続的に保存され、再度マウントすることができます。ドライブには次のような用途があります。

- 開発者は NTFS ファイル システムを含む VHD をアップロードし、この VHD を Windows Azure ドライブとしてマウントできます。これにより、Windows Azure とオンプレミスの Windows Server システムとの間で、ファイル システムのデータを簡単に移行できます。

- Windows Azure の開発者は、Windows Azure ドライブを基盤ストレージとして使用し、Windows Azure のロール インスタンス内で MySQL データベース システムをインストールして実行できます。

テーブル

単なるバイトの集合体である BLOB の理解は容易ですが、テーブルは多少複雑です。図 14 に、テーブルの各種構成要素の関係を示します。

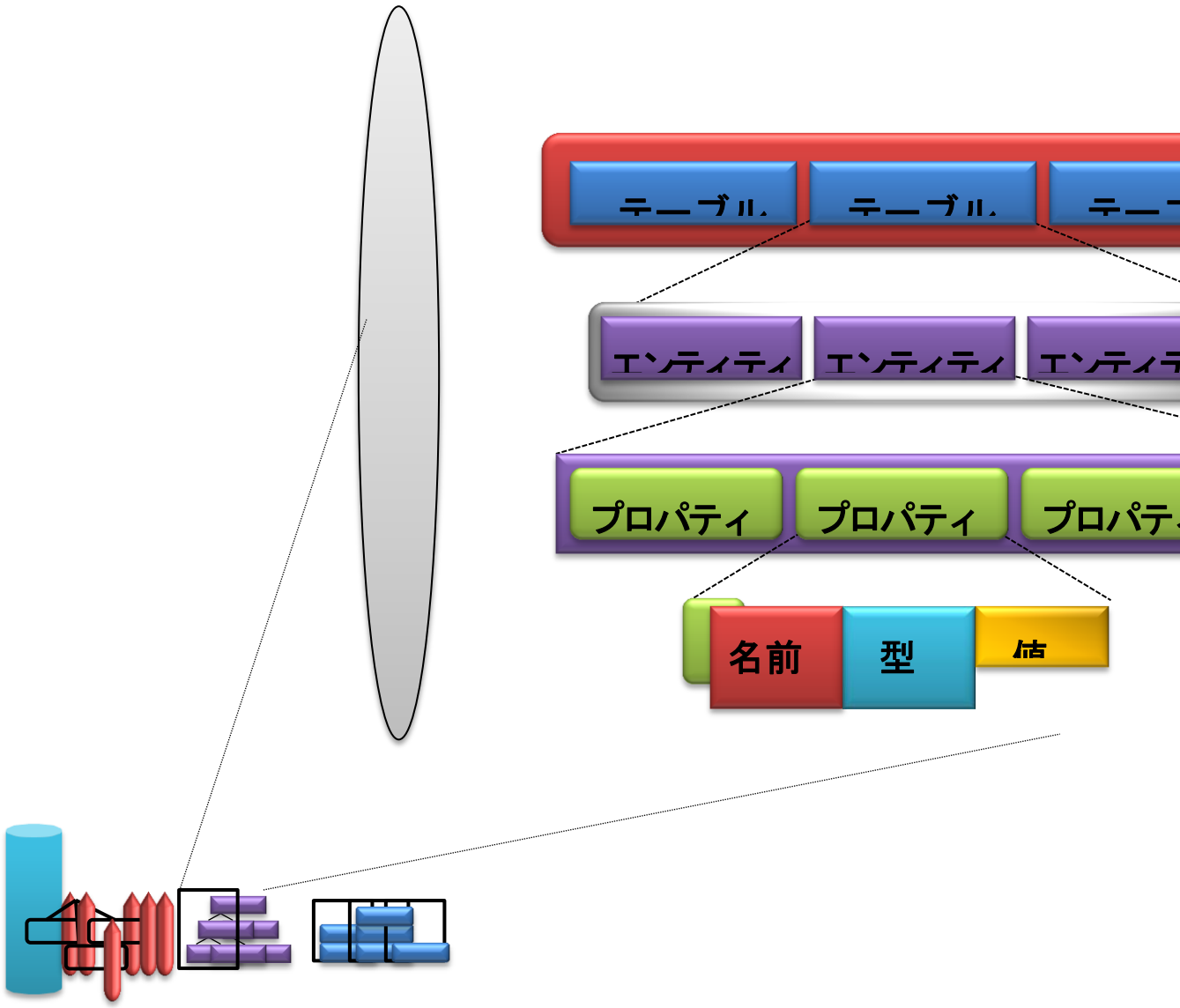


図 14: エンティティベースのストレージを提供するテーブル

図に示すように、各テーブルは複数のエンティティで構成されます。エンティティには、名前、型、および値からなる 0 個以上のプロパティが含まれます。Binary、Bool、DateTime、Double、GUID、Int、Int64、String などのさまざまな型がサポートされており、プロパティには、格納されている値に応じて、さまざまなタイミングで異なる型を使用できます。また、エンティティ内のすべてのプロパティが同じ型である必要はなく、開発者がアプリケーションに最も適したものを自由に選択できます。

含まれるプロパティにかかわらず、エンティティのサイズは最大 1 MB で、常に 1 つの単位としてアクセスされます。エンティティを読み込むとすべてのプロパティが返され、エンティティを書き込むと、すべてのプロパティを置き換えることができます。1 つのテーブルに含まれる一連のエンティティをひとまとまりとして更新し、全エンティティの更新結果が成功または失敗となるようにすることも可能です。

Windows Azure ストレージのテーブルは、多くの点でリレーショナル テーブルとは異なります。最も明らかな違いは、一般的な意味でのテーブルではないことです。また、通常の ADO.NET を使用してアクセスできず、SQL クエリもサポートしません。Windows Azure ストレージのテーブルではスキーマが強制されないため、同一エンティティ内の複数のプロパティが異なる型を持つことができ、さらにその型を後で変更できます。当然、

次のような疑問が生まれます。なぜ単純に、標準の SQL クエリを使用する通常のリレーショナル テーブルをサポートしないのでしょうか。

それは、Windows Azure が、大規模なスケーラビリティを備えたアプリケーションのサポートを第一の目標としているためです。従来のリレーショナル データベースは、スケールアップ、つまり DBMS を実行するコンピューターを拡張し続けることによって、増加するユーザーに対応します。しかし、本当に膨大な数の同時ユーザーをサポートするには、ストレージをスケールアップするのではなく、スケールアウトする必要があります。それには、ストレージ メカニズムをシンプルにする必要がありますが、標準の SQL による従来のリレーショナル テーブルでは、その目的をうまく達成できません。そこで必要となるのが、Windows Azure のテーブルが提供するような構造です。

使い慣れたリレーショナルのコンセプトをそのまま適用できないため、テーブルの使用には開発者側での見直しが必要となりますが、それでもなお、きわめてスケーラブルなアプリケーションを作成できる点で、このアプローチは効果的です。このアプローチでは開発者が規模について考慮する必要がなく、新しいテーブルを作成して新しいエンティティを追加しさえすれば、後の処理は Windows Azure が実行します。また、DBMS の管理に要する多くの作業も、Windows Azure が行うため不要になります。その結果、開発者は大量のデータの保存や管理の方法に煩わされることなく、アプリケーションの開発に専念できます。

Windows Azure ストレージの他の機能と同様に、テーブルには REST ベースでアクセスします。.NET アプリケーションの場合、こうしたアクセスに WCF Data Services を使用すれば、基となる HTTP 要求は意識されません。また、.NET を含むすべてのアプリケーションでは、このような要求を直接実行することもできます。たとえば、特定のテーブルへのクエリは、URI に対する HTTP GET として表現され、次のように記述されます。

http://<*StorageAccount*>.table.core.windows.net/<*TableName*>?$filter=<*Query*>

ここで、<*TableName*> には照会先のテーブルを指定し、<*Query*> には、このテーブルに対して実行されるクエリを入力します。クエリによって多数の結果が返される場合は、継続トークンを取得して、次のクエリに渡すことができます。これを反復することで、ひとまとまりの完全な結果セットを取得できます。

Windows Azure テーブルはすべてのストレージ シナリオに適しているわけではなく、また、使用にあたっては開発者が多少新しい知識を習得する必要がありますが、スケーラビリティを必要とするアプリケーションには、テーブルの利用が最適といえます。

キュー

テーブルと BLOB は主に、データの保存とアクセスを目的としていますが、キューの最大の目的は、Windows Azure アプリケーションの異なる要素どうしの通信を可能にすることです。Windows Azure ストレージの他の機能と同様に、キューには REST ベースでアクセスします。Windows Azure アプリケーションと外部のアプリケーションは共に、次のような URI 形式を使用してキューを参照します。

http://<StorageAccount>.queue.core.windows.net/<QueueName>

既に述べたように、クエリは一般的に、Web ロール インスタンスと Worker ロール インスタンスとの間で通信を行うために使用されます。図 15 に、このしくみを示します。

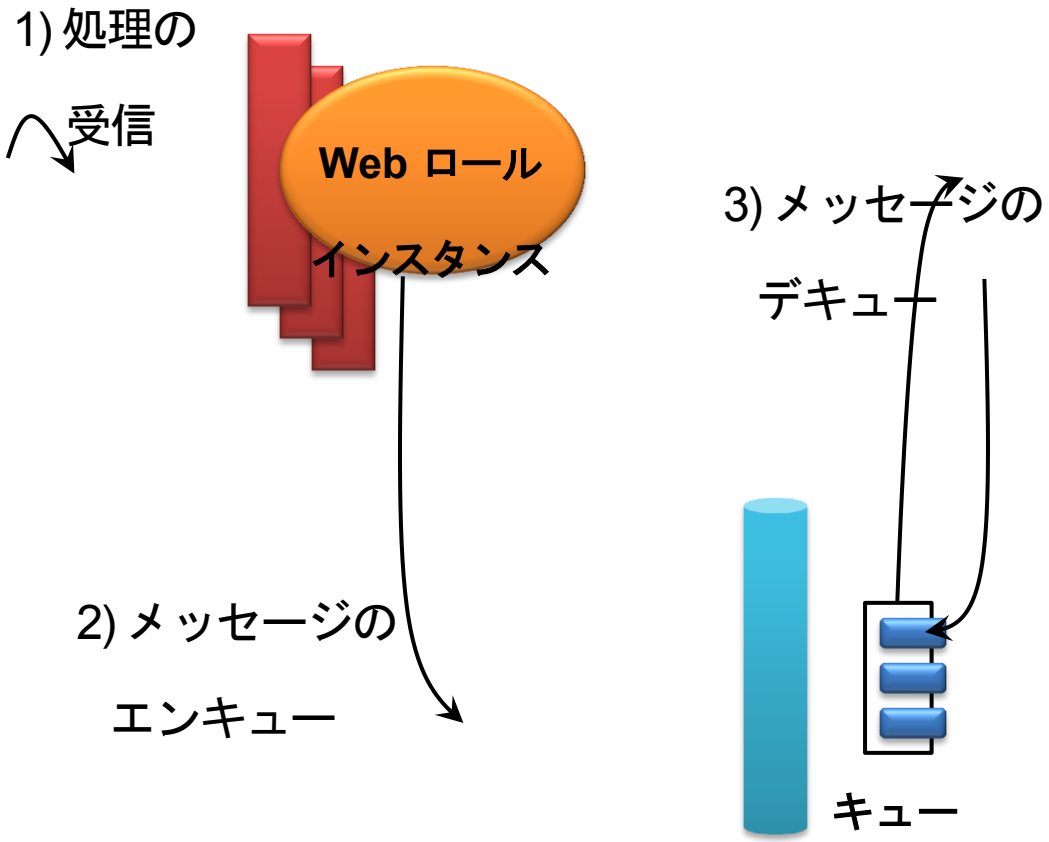


図 15: エンキュー、デキュー、処理の後で、キューから明示的に削除されるメッセージ

通常のシナリオでは、複数の Web ロール インスタンスが実行され、それぞれがユーザーからの処理要求を受信します (ステップ 1)。その処理要求を Worker ロール インスタンスに渡すため、Web ロール インスタンスがキューにメッセージを書き込みます (ステップ 2)。このメッセージのサイズは最大 8 KB で、アプリケーションによって異なりますが、BLOB やテーブル内のエンティティなどを参照する URI が含まれている場合があります。Worker ロール インスタンスはこのキューからメッセージを読み取り (ステップ 3)、メッセージが要求する処理を実行します (ステップ 4)。ただし重要なのは、キューからメッセージを読み込んでも、実際にはメッセージが削除されない点です。代わりに、そのメッセージは一定期間 (既定では 30 秒)、他のインスタンスから見えなくなります。Worker ロール インスタンスがこのメッセージの要求する処理を完了すると、対象のメッセージがキューから明示的に削除されます (ステップ 5)。

Web ロール インスタンスと Worker ロール インスタンスの分離は理にかなっています。長時間かかる処理が終わるまでユーザーが待機する必要がないうえ、いずれかのインスタンスを追加するだけで簡単にアプリケーションを拡張できるためです。では、インスタンスによってメッセージを明示的に削除する理由は何でしょうか。それは、障害に対応するためです。メッセージを読み込んだ Worker ロール インスタンスの処理が成功した場合は、そのメッセージはまだ非表示の状態 (30 秒以内) でも削除されます。ただし、Worker ロール インスタンスがメッセージをデキューし、メッセージで指定された処理を完了する前にクラッシュした場合には、そのメッセージはキューから削除されません。非表示期間がタイムアウトになると、メッセージはキューに再表示され、別の Worker ロール インスタンスによって読み込まれます。これによって、各メッセージが最低 1 回は確実に処理されることになります。

この説明からわかるように、Windows Azure ストレージのキューは、Microsoft Message Queuing (MSMQ) などの馴染みのあるテクノロジのキューとは異なるセマンティクスを持っています。たとえば、従来のキ

ュー システムでは、先入れ先出し式のセマンティクスによって各メッセージが 1 回限りしか配信されない場合があります。Windows Azure ストレージのキューでは、こうした約束事はありません。前述のとおり、メッセージは複数回配信することが可能であり、決まった順序でメッセージが配信されるとは限りません。このように、クラウドでは従来と異なる概念が使用されるため、開発者はその違いに順応していく必要があります。

ファブリック コントローラーについての考察

アプリケーション開発者にとって、Windows Azure の最も重要な部分はコンピューティングとストレージです。ただし、そのどちらも、ファブリック コントローラーなしには機能しません。ファブリック コントローラーは、データ センターのすべてのマシンを連携させて一体化することによって、その他すべての要素の基盤を提供します。

既に述べたように、ファブリック コントローラーは特定の Windows Azure データ センター内のすべてのリソースを所有します。また、物理マシンにアプリケーションを割り当てる役割も持っています。こうした割り当てはインテリジェントに実行することが重要です。たとえば、開発者がアプリケーションで使用する 5 つの Web ロール インスタンスと 4 つの Worker ロール インスタンスを要求したとします。単純な方法だと、1 つのネットワーク スイッチがサービスを提供している同じラック内のマシンに、これらのインスタンスがすべて割り当てられることがあります。この場合、ラックまたはスイッチのどちらかに障害が発生すると、アプリケーション全体が利用できなくなってしまいます。Windows Azure は高可用性を目標としているため、アプリケーションがこのような単一障害点に依存するのは適切ではありません。

この状態を回避するため、ファブリック コントローラーは所有するマシンをグループ化し、複数の "停止ドメイン" を作成します。各停止ドメインはデータ センターの一部であり、1 か所に障害が発生すると、そのドメイン内のすべてに対するアクセスをシャットダウンできます。図 16 に、この概念図を示します。

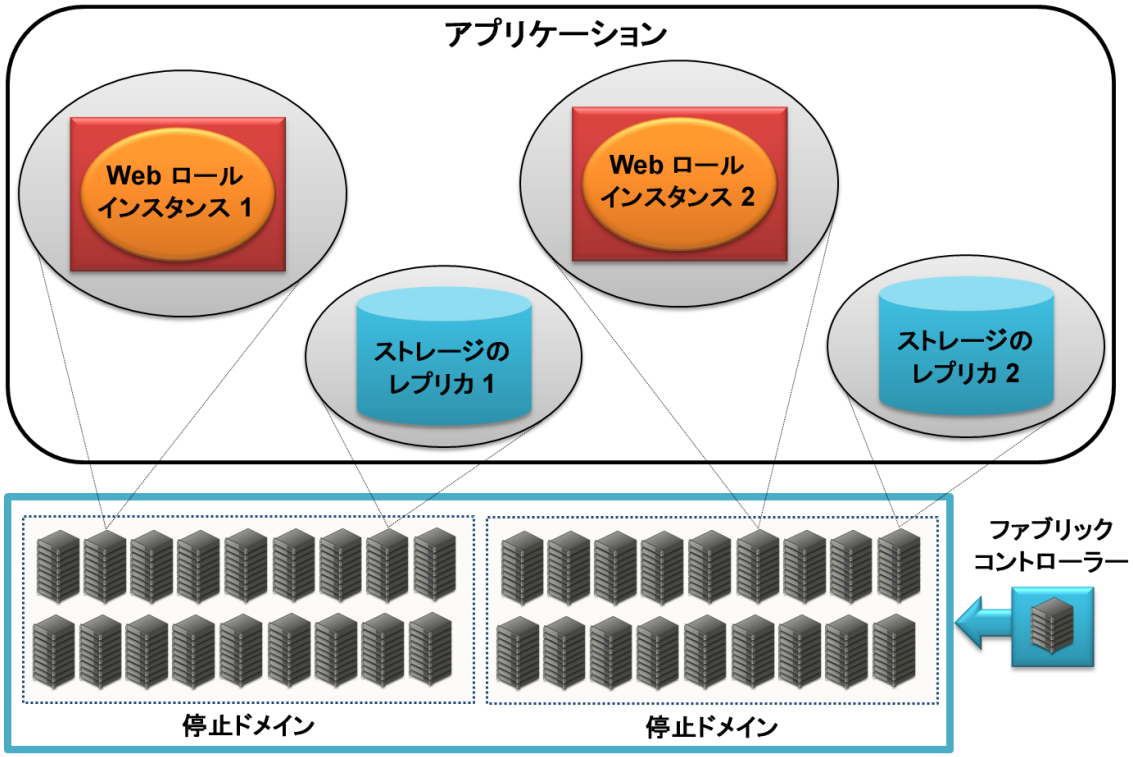


図 16: ファブリック コントローラーによって異なる停止ドメインに分散された、アプリケーションの各インスタンス

ここでは、アプリケーションが 2 つの Web ロール インスタンスのみを実行し、データ センターは 2 つの停止ドメインに分割されている、シンプルな例を示しています。ファブリック コントローラーはこのアプリケーションを展開する際に、それぞれの停止ドメインに 1 つずつ Web ロール インスタンスを割り当てます。このように割り当てることで、データ センター内の 1 つのハードウェア障害がアプリケーション全体の停止につながることを回避しています。また、ファブリック コントローラーは Windows Azure ストレージを別のアプリケーションと見なすため、データのレプリケーションは行いません。代わりにストレージ アプリケーション自体がこの処理を行い、このアプリケーションで使用する BLOB、テーブル、およびキューのレプリカが、確実に別の停止ドメインに作成されます。

ハードウェア障害が発生しても、アプリケーションの実行を継続できるようにしておくことは効果的ですが、それだけでは不十分です。Windows Azure がサポート目標とするような、本当に信頼性の高いアプリケーションでは、シャットダウンを伴わずに更新処理を実行できるようにする必要があります。これを実現するための選択肢の 1 つが、既に説明したアプローチを使用して、アプリケーションを既存のバージョンから新しいバージョンへと切り替える方法です。もう 1 つの選択肢として、Windows Azure の "更新ドメイン" を利用することもできます。このアプローチでは、ファブリック コントローラーによって、アプリケーションのロールの各インスタンスが、異なる更新ドメインへと割り当てられます。新バージョンのアプリケーションを展開する場合、ファブリック コントローラーは一度に 1 つの更新ドメインに新しいコードを展開します。ファブリック コントローラーは対象のドメインに属するロール インスタンスを停止させ、そのロールのコードを更新した後、新しいインスタンスを起動します。これは、アプリケーションの更新中も、アプリケーションの実行を継続することを目的とした方法です。更新中、一部のインスタンスがシャットダウンしている間や、さまざまなユーザーがさまざまなバージョンのアプリケーションにアクセスしている間などは、アプリケーションの応答時間が長くなる傾向があるため、ユーザーが更新に気付くこともあります。それでもユーザーから見ると、全体としてはアプリケーションを継続的に利用できます。

更新ドメイン (アプリケーションのプロパティ) を、停止ドメイン (データ センターのプロパティ) と混同しないよう注意してください。ただし、いずれのドメインも、ファブリック コントローラーが常時 Windows Azure アプリケーションを実行できるように支援するという、同一の重要な目的を持っています。

将来的な展望

マイクロソフトは、次のような機能を 2011 年に Windows Azure に追加する計画を発表しています。

- Windows Azure Platform Appliance: ホスティング サービス会社および企業がオンプレミスのデータ センター内で Windows Azure を運用できるようにする、ハードウェアとソフトウェアのセット。
- **CDN ダイナミック コンテンツ キャッシュ**: 現在、Windows Azure CDN で扱うことができるのは BLOB データのみです。新たに追加されるこの機能を使用すると、Windows Azure アプリケーションによって動的に作成されたコンテンツも CDN でキャッシュできるようになります。
- **VM ロール スナップショット: Windows Azure VM ロールの最初のリリースでは、実行中に行われた OS ボリュームへの変更を一切保存できません。スナップショットによってこれが改善され、OS ボリュームの状態を永続的ストレージへと定期的に保存できるようになります。**
- **Java のサポートの向上:** Windows Azure では現在も Java アプリケーションを実行できますが、マイクロソフトはさらなるサポート機能の追加を計画しています。予定されている改良点には、Java のパフォーマンス向上、Eclipse ベースのツールに対するサポート強化、および Windows Azure 用 Java ライブラリの拡充などがあります。

これまで同様、これらの機能を追加することで、マイクロソフトはこのクラウド プラットフォームを、より幅広い状況で利用できるようにすることを目指しています。

まとめ

クラウド内でのアプリケーションの実行やデータの保存は、多くの状況で最適な選択肢となります。Windows Azure では、さまざまな要素を連携させることでこうした選択を可能にします。Windows Azure 開発環境、SQL Azure、およびそれ以外の Windows Azure Platform の機能を組み合わせることで、Windows 開発者に、この新しい環境に移行するための足がかりを提供します。

現在はまだ、クラウド プラットフォームはほとんどの企業にとって、やや特別な選択肢であるといえます。ただし、Windows Azure を始めとするクラウド プラットフォームが実績を重ねていくことにより、この新しいアプローチへの違和感は薄れていくでしょう。将来的には、クラウドベースのアプリケーションや、その実行基盤であるクラウド プラットフォームが、ソフトウェアの世界においてますます重要な役割を果たしていくに違いありません。

参考情報

- Windows Azure Platform ホーム ページ

  http://www.microsoft.com/japan/windowsazure

- Windows Azure Platform の概要 (David Chappell)

  http://go.microsoft.com/fwlink/?LinkId=158011

- Windows Azure BLOB: BLOB ストレージのプログラミング (英語)

  http://go.microsoft.com/fwlink/?LinkId=153400

- Windows Azure テーブル: テーブル ストレージのプログラミング (英語)

  http://go.microsoft.com/fwlink/?LinkId=153401

- Windows Azure キュー : キュー ストレージのプログラミング (英語)

  http://go.microsoft.com/fwlink/?LinkId=153402

執筆者について

David Chappell は、カリフォルニア州サンフランシスコに拠点を置く Chappell & Associates (www.davidchappell.com) の代表を務めており、講演、執筆、コンサルティングを通じて、世界中の人々が最新テクノロジを理解して使用し、適切な意思決定を行えるよう支援しています。