〈導入〉

1 クラウドの成立過程とその技術的特徴について

丸山不二夫　早稲田大学

クラウドへの関心が高まっている．本稿では，ネットワークを通じてサービスを提供する典型的な形態としてのクラウドの成立過程を振り返り，その中で形成されたクラウドの技術的な特徴を見ていきたいと思う．

本稿の構成について

クラウドというコインには，サービス提供とサービス利用という 2 つの面がある．

クラウドのサービス利用については，「所有から利用へ」という言葉が代表するように，クラウドの経済的なメリットに大きな期待が集まっている．現実には，さまざまなクラウドへの志向を動かしているのは，こうした関心なのであるが，残念ながら，本稿では，それらに十分に触れることはできなかった．本稿は，主にクラウドのサービス提供の面にフォーカスしている．

最初に，本稿の構成の概略を述べておきたい．

第 1 に，本稿は，サービス提供者としてのクラウドは，次のような段階を経て発展してきたと考えている．

- 2004 年の Google の上場を画期とする，クラウドという概念が成立する以前の，インターネットの普及の中で，事実として「インターネット・クラウド」が成立した時期．
- 2006 年の Amazon EC2/S3 のサービス開始を画期とする，インターネット・クラウドの中から「エンタープライズ・クラウド」が派生し，クラウド概念が登場した時期．
- 2008 年の Microsoft Azure の登場を画期とする，クラウド概念の成熟とともに，派生形ではない，自立型のエンタープライズ・クラウドが登場した時期．

第 2 に，本稿では，インターネット・クラウドの特質として，「Web スケール」という特質に注目している．インターネットに膨大な数の個人・企業が登場し，ネット上の情報が爆発的に増大し，同時に，そこに今まで存在しなかった新しいビジネスが誕生すること．それが，Web スケールという抽象を可能にした．Web スケールの個人のニーズに支えられた，Web スケールのビジネスは，Web スケールのシステムを必要とする．それがクラウドである．

第 3 に，本稿では，クラウドの技術的特徴について次のように考えている．Web スケールのシステムは，技術的には，Scalability，Availability を特徴とする Scale-out アーキテクチャに適合的である．そこでは，リレーショナル・データベースに代わって，Key/Value 型のデータ・ストアが重用される（Key/Value 型のデータ・ストアの技術的特徴については，本特集の，首藤解説を参照されたい）．本稿の後半で見るように，古典的な整合性概念は Eventually Consistency 概念に，伝統的な ACID（Atomicity, Consistency, Isolation, Durability）トランザクションの概念は BASE（Basically Availability, Soft state, Eventually consistency）トランザクションに置き換わろうとしている．

クラウドの前史（1）

■インターネットの普及とエンタープライズ・システムの変化

クラウドに歴史的に先行したのは，言うまでもなくインターネットの爆発的な普及である．前世紀の最後半 90 年代半ばに起きたこの動きは，またたくまに，世界を覆い尽くした．日本でのインターネット利用の状況を次の図 -1 にまとめた．21 世紀に入ってすぐに，日本でのインターネット普及率は全世帯の 5 割を超え，その後も拡大を続け，現在では，8 割に近づいている．インターネット接続におけるブロードバンド接続の割合も急速

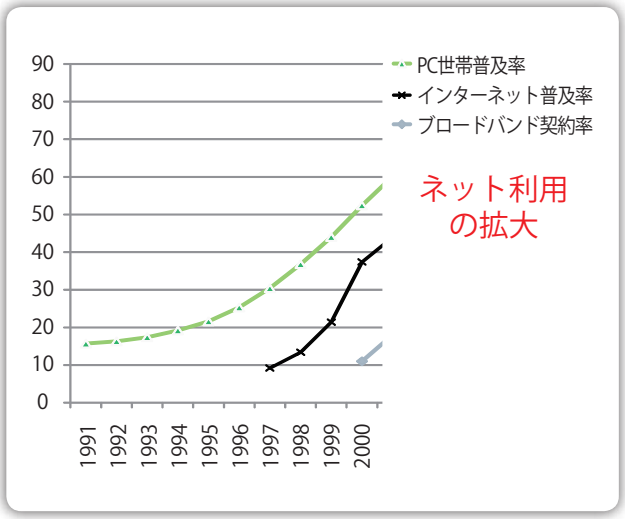


図-1　日本におけるインターネット利用の拡大

に増加し，今日では，ほとんどすべてのインターネット接続がブロードバンド化するに至っている．グローバルには，インターネット利用者は，15 億人に達している．

家庭でのインターネット利用の前提には，家庭での PC の普及があった．日本での PC 普及は，1980 年代後半から始まったが，21 世紀直前には全世帯の 5 割を超え，今では 9 割近い家庭に普及している．この背景には，コンピュータの劇的な価格低下がある．こうしたコンピュータのコモディティ化は，クラウドにおけるスケールアウト技術を支える重要な一因となっている．

インターネットのインパクトは，IT 技術のフラグシップであるエンタープライズ・システムにも，深い影響を及ぼす．90 年代，企業内のネットワーク化が進んだが，そこでのネットワーク利用の主要な形態は，企業内に閉じた「サーバ・クライアント・モデル」であった．20 世紀末から 21 世紀の初めにかけて，インターネットの爆発的な普及の影響を強く受けて，エンタープライズのシステムでは 2 つの大きな変化が進行する．1 つは，「Web アプリケーション」の登場である．Web サーバを前面に立て，その後ろにビジネス・ロジックを担うコンポーネントを置き，最背面にデータベースを置く構成から，「3-Tier Model」と呼ばれることもある．この構成は，さまざまの変化を遂げながらも，現在のクラウド技術の骨格を形成している．もう 1 つは，それに少し遅れて登場した「Web サービス」である．のちにクラウドの主要なプレーヤとなる Google，Amazon は，SOAP（Simple Object Access Protocol）中心の Web サービスに対して，REST（Representational State Transfer）の一貫した推進者であった．現在のクラウド技術では，Microsoft を含めて，RESTful Web サービスが主流である．

クラウドの前史（2）

■ネット・ビジネスの勃興と企業でのネット利用の拡大

エンタープライズ・システムの Web アプリへの変化は，新しいビジネスの創出を可能とした．企業がネットを通じて，たくさんの顧客と直接コンタクトすることが可能となったのである．こうした新しい市場を狙って，e-コマースを中心に，新しいネット上のビジネスが続々と誕生し，ビジネス的にも大きな成功を収めた．Amazon の株式公開は 1997 年，eBay の株式公開は 1998 年，日本の楽天の株式上場は 2000 年のことである．技術的には，これらのシステムは，Web アプリを提供する 3-Tier モデルの大規模化として実現されてきた．

同時に，企業のイントラネットで主要に利用されるエンタープライズの基幹システムでの，Web アプリ技術，Web サービス技術の利用が，J2EE を中心に急速に進んだことも重要である．それらの基幹システムは，インターネットへのアクセス能力をも拡大しつつ，企業システムのネットワーク化を進行させた．

「Web スケール」という概念

企業の内外を問わず，グローバルな規模でネットワークとネットユーザの遍在は急速に進んだ．重要なことは，この時期に，「Web スケール」という規模感が形成され，多くの人に共有され始めたことである．「Web スケール」は，インターネットのグローバルな展開が初めて可能にした概念で，量的には，それまでの基準と比べると桁違いに多数で，かつ，質的には，その数が絶え間なく増大し得るものである．典型的には，世界中のネット上の情報の総体を考えてみればいい．

システムが対象とする顧客が Web スケールであるとき，システムの「規模」の問題は，システムに大きなインパクトを与える．第 1 に，ビジネスの拡大につれて，システムの拡大というシステムの変更が日常化する．第 2 に，3-Tier モデルは，Web 層とビジネス・ロジック層を多重化することで比較的容易に規模拡大できる．しかし，それもある程度までである．システム全体を通じてユニークなデータを保持する多重化を許さないデータベース層が，システム全体のボトルネックになる．第 3 に，単なる Web スケールに収まらない処理があり得る．Web スケールの N 個のデータと Web スケールの M 個のデータをつき合わせる必要がある処理が生まれてくる．たとえば，e-コマース・サイトでのリコメンデーション（ユーザごとのおすすめ商品紹介）．あるユーザ A の過去の

購買履歴から，それぞれの購入品について，購入者のリストを作り，彼らが購入したすべての商品のリストを作り，購入頻度の高いものの中から，ユーザ A がまだ購入していないもののリストを作る．その計算量は膨大である．

Web スケールの顧客と Web スケールのデータは，Web スケールに対応できるシステムを必要とする．何よりも，Web スケールでのビジネスが，Web スケールのシステムを必要としたのだ．

ネットワークへの個人の登場

■サーバ・セントリック P2P 型システムの発展

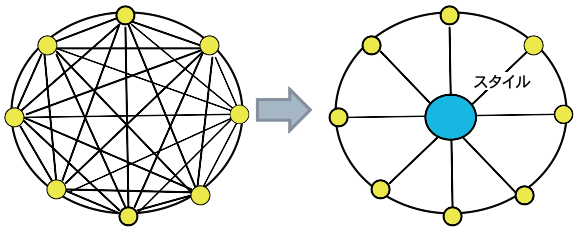
ネットワークに登場したのは企業だけではない．PC のコモディティ化とネットワークの低価格化が進む中，無数の個人がインターネットの世界に主体的な情報発信者として登場するようになる．Blog や SNS などの新しいネットワーク・メディアが生まれ，ネットワークのブロードバンド化と相まって，ネット上のサービスと情報は，爆発的に増大を始めた．

Youtube，Flickr，Facebook，Twitter などの新しいソーシャル・メディアは，基本的には，個人と個人が情報を共有し，あるいは，個人と個人がコミュニケーションをするという，個人と個人を結ぶ Person to Person のメディアである．技術的には，こうした機能を，中心にあるサーバが担っている．こうしたシステムを，サーバ・セントリック P2P 型システムという．こうした，従来のエンタープライズ・システムには存在しなかった新しい構成のシステムが発展し，ビジネス的にも大きな成功を収めている．日本でも，mixi や GREE が，こうしたタイプの企業である．

O'Reily の Web2.0 というコンセプトが提起されたのは，2004 年になってからのことだが，こうした流れは，20 世紀末から今日に至るまで，絶え間なく進んでいる．そして，携帯等のモバイル・デバイスの普及とそれらのブロードバンド・インターネット接続によって，いままた，第 2 段階の情報爆発が起ころうとしている．

多くの人に次のような確信が生まれてくる．

「我々は，いつでもインターネットにつながることができる．我々は，インターネットを通じて，望んだものを，望んだ時間に，望んだデバイス上で得ることができる．サービスや情報は，ネットを通じて，空の上から降ってくる」



コミュニケーションと情報の共有
論理的にはPerson to Person
実装は，サーバ中心

図 -2　Server Centric P2P

インターネット・クラウドの成立

■すべては，Google から始まった

本稿では，「インターネット・クラウド」という言葉を，Web スケールの多数の消費者を対象として，彼らにインターネットを通じてサービスを提供しようとする大規模なサーバ群・データセンタのことを指して使っている．こうした定義に従えば，大規模な e- コマース・サイトや大規模な SNS サイトも，「インターネット・クラウド」に含まれることになるのだが，それは構わないと筆者は考えている．

今日から振り返ってみれば，インターネット・クラウドの成立に，技術的には 1 つの画期があることは明確である．Google の登場である．Google は，自らは，クラウドという名前を使うことはなかったが，誰よりも先に，Web スケールのデータの巨大さをはっきりと認識していた．クラウドに関していえば，すべては Google から始まったと言っていいと筆者は考えている．Google の上場は，2004 年のことである．

彼らは，ネットワーク上の情報爆発に注目し，検索手段の提供によって，その情報を「組織しアクセス可能にする」ことが，時代の大きな課題だということを認識していた．今では，当たり前の発想に思えるが，これは，卓見である．そればかりではない．彼らは，この検索サービスに広告を連動させるというビジネス・モデルを作り出す．Google は，インターネット・クラウドの最大の成功者になった．

インターネット・クラウドの技術的特徴

■Google の大規模分散システム

Google が作り上げたシステムは，技術的にも独創的なものであり，インターネット・クラウド技術の画期をなすものであった．Google が切り開いた技術的特徴は，

それ以降のクラウド技術にも決定的な影響を与え続けている．それを以下に見ておこう．

第 1 に，Google は，Web スケールの処理を行うのに，コモディティ化した安価なマシンをたくさんネットワーク上に並べて協調動作させるという，Scale-out の戦略を採用した．世界中のページをクロールしてそのデータを格納するには，巨大な容量のファイルシステムが必要となるのだが，彼らはそれを自分たちで，Scale-out する分散ファイルシステム GFS（Google File System）を作り上げたのである．

第 2 に，安価な数千数万ものマシンから構成されているシステムでは，ハードウェアの故障は，例外としてではなく，いつでも起こり得る．だから，システムの Availability を確保するためには，恒常的なモニタリング，エラーの検出，自動的な回復機能が，システムに統合されている必要がある．Google は，こうした問題に，ハードではなくソフトの力で対応するのである．

第 3 に，Google は，こうして，リアルタイムに Web スケールの膨大なリクエストに応え得る Scalable で Available な巨大な分散システムを構築しただけではなく，そうした処理が難しい，Web スケールの複数の巨大なデータ同士を突き合わせる処理についても，まったく新しい処理のスタイルを確立した．MapReduce というアルゴリズムとその処理系である．MapReduce は，Google の大量データ処理の心臓部分として，分散処理のアルゴリズムに，大きな影響を与えた．

第 4 に，3-Tier モデルのボトルネックであったデータベースについても，巨大なテーブルを，ネットワーク上のノードごとにキーの範囲で分割した小テーブル（Tablet）を分散配置し，それらへの並列なアクセスを可能とした．Scale-out する，超大容量の高速データ・ストアである BigTable の Key/Value 型のデザインは，その後のクラウド・システムのデータベース設計に，決定的な影響を与えた．

第 5 に，Google は，彼らの巨大な分散処理システムを，データセンタとして運用するに際して，そのエネルギー・コストの問題の重要性にいち早く気付き，徹底して無駄を省き，経済的であると同時に環境にも配慮した（二重の eco）データセンタ構築技術を作り上げた．

エンタープライズ・クラウドの登場

■Amazon EC2/S3 の商用サービスの開始

Google の確立したインターネット・クラウド技術は卓越したものであったが，クラウド技術は，そこで完成したわけではない．クラウド技術の第 2 段階への進化は，e- コマースの雄 Amazon によってなされた．Google のインターネット・クラウド技術が，主には，一般のサービス消費者を対象にしていたのに対して，Amazon は，エンタープライズを対象とするクラウド・サービス EC2/S3 を開始したのである．本稿では，「エンタープライズ・クラウド」という言葉を，自身のサービス提供を目指すユーザを対象として，彼らにそのサービス提供の環境を提供する新しいサービスを支える大規模なサーバ群・データセンタを指して用いている．

Amazon は，大規模な e- コマース・サイトの複雑で困難な運用の経験の中から，リソースを合理的に管理する方法を編み出した．

Web 上の高度に分散したソフトウェアの開発作業では，開発者は，プログラムのサイズにかかわらず，仕事の 70% を，保守作業やビジネス上の差別化に結び付かない仕事に費やしている．

このシステムが革命的なのは，それが，開発者に，本来集中すべき仕事に集中させ，その仕事を片付けるやり方を変えるということである．

EC2/S3 がサービスを開始した 2006 年の Web2.0 Summit での Amazon 会長の Jeffery Bezos の言葉である．Amazon のシステムは，単に，Amazon の開発者をシステム管理の膨大な雑務から解放しただけではなかった．Amazon は，そこで生まれた開発者のリソースと，柔軟に Scale-out する自らのインターネット・クラウドの余力を，エンタープライズ・クラウド・サービスとして，外部に提供することに成功したのである．Amazon のクラウド商用サービスの開始は，クラウドのもう 1 つの画期であった．

EC2/S3 のようなサービス提供ができるためには，ネットワーク上に分散して存在している物理的なディスクや物理的なサーバを，仮想化して論理的に管理して，稼働していないものはリソース・プールに登録して，変動する要求に応じて動的にそこからリソースを取り出して仕事を割り当てて，スケーラブルなサービス提供を保証しなければならない．Amazon の EC2/S3 のサービス開始は，クラウドを支えるこうした課題の重要性に，皆の眼を向けさせた．

クラウド概念の成立

筆者は，クラウドという概念が成立したのは，この 2006 年頃だったと考えている．

誰によってと問うことは，あまり意味がない．そもそも，インターネット・クラウドという言葉は，パブリックなネットワークを意味する雲型のアイコンとともに，ネットワークの世界では，Google 以前から広く用いられていた．インターネット・クラウドは，Web スケー

ルの数の個人をターゲットとしたビジネスによってドライブされ，ビジネスの発展とともにそのシステムの規模を不断に拡大していくという傾向を持っている．こうしたインターネット・クラウド環境の中で，Scale-out アーキテクチャ，Availability の確保，Key/Value データ・ストア，BASE トランザクションといったクラウドの技術的な特徴は，日々，新たに再発見・再発明され得るのである．そうした意味では，Web スケールでのインターネット・クラウドの発展こそが，クラウドの産みの母なのだ．

ネットワークを通じて企業にサービスを提供するという点では，インターネット・クラウド起源のエンタープライズ・クラウドは，それ以前から存在していた ASP や SaaS の潮流と合流を果たしたともいえる．Salesforce が，その代表である．彼らのシステムは Scalability を欠いており，Scale-out アーキテクチャを取っているわけではない．このように，クラウド・サービスの形態としては同じに見えるが，起源が異なっていることには留意が必要である．ただ，これらのシステムも，サービスの拡大と価格競争の中で，遅かれ早かれ，Scale-out 型に変化していくだろうと筆者は考えている．

エンタープライズ・クラウドの新段階

■Microsoft Azure の登場

クラウド概念が成立すると，エンドユーザにではなくサービス提供者にサービスを提供するというエンタープライズ・クラウドのビジネスの可能性に注目して，この市場に新しく参入しようという動きが生まれてくる．まず，Google は，自らのインターネット・クラウドのリソースを利用して，サービス提供者向けのサービス Google App Engine を開始する．

エンタープライズ・クラウドの新しい画期をなすのは，Microsoft のクラウド Windows Azure の登場である．2008 年のことである．Microsoft は，多くのインターネット・クラウド・サービスを展開していたのだが，彼らのクラウドは，これらから派生したものではない．彼らは，先行した Google，Amazon らのシステムを研究し，スクラッチから，巨大なクラウド・データセンタの建設に着手したのである．

Azure は，Google がアプリケーションの背後に隠し，Amazon がユーザには意識させようとしなかった，クラウド上のリソース管理を，クラウド OS としてユーザに公開した．これまでの OS がファイルシステムをその最重要の構成要素としたように，クラウド OS としての Azure は，Key/Value データ・ストアと Queue と Blob の 3 者を，そのストレージとしたのである．

Key/Value データ・ストアは，Scalability と Availability の両方の要求に応えるように，P2P ネットワークにおける分散索引技術である DHT（Distributed Hash Table）を応用したもので，ユーザは，URI を利用した REST インタフェースと，簡便な操作で，データの CRUD 操作（作成（Create）・読み出し（Read）・更新（Update）・削除（Delete））が可能になる．Queue は，2 つのプロセス間のパイプに相当するものだが，簡単なメカニズムで，複雑な分散プロセス間の同期に威力を発揮する．BLOB（Binary Large Object）は，データベースにおいて巨大なバイナリデータを格納するためのデータ型のことであり，Azure でもこれを利用することができる．これらのクラウド上のリソースを Azure ユーザは，自由に利用できるのだ．

よりチャレンジングな試みは，「サービス・モデル」というシステム記述を利用すると，「Fabric コントローラ」が働いて，クラウド上のノードのトポロジやそれぞれの役割を，指定できるという機能である．こうして，ユーザは，自分自身のシステムを設計して，それをクラウド上に自由にデプロイできるようになる．

Azure のメリット

Google らのインターネット・クラウドのサービス提供のスタイルを見て，クラウドはエンタープライズ向けには使えないと考えている人も多いが，それは誤解である．Microsoft Azure の何よりの特徴は，クラウドのエンタープライズ利用に，強くフォーカスしているところにある．

第 1 に，Azure は，既存システムのクラウドへの移植・移行が容易であるように設計されている．Azure は，デフォルトでは，エンタープライズでは標準的な手法である Web アプリ・エンジンとして機能する．開発者は，新たなスキルの習得を最小限にして，現実にエンタープライズで利用されている Web アプリの開発手法をそのまま利用して，クラウドのアプリケーション開発に移行できる．

第 2 に，Azure では，こうした開発手法の同一性を利用して，Azure 本体と On Premise のシステムとの境界を移動して，クラウド重視型から既存の On Premise 重視型までのさまざまなシステムを柔軟に設計できる．クラウド利用は，ハイブリッド型が主流になると筆者は考えているが，Azure は，最初からそれに対応している．

第 3 に，こうして作られたシステムが，従来のシステムと決定的に異なるのは，要求に応じてシステムの Scale-out，Scale-in が，動的に自由にできることである．利用者は，設備を所有せず，利用に応じて料金を払うだ

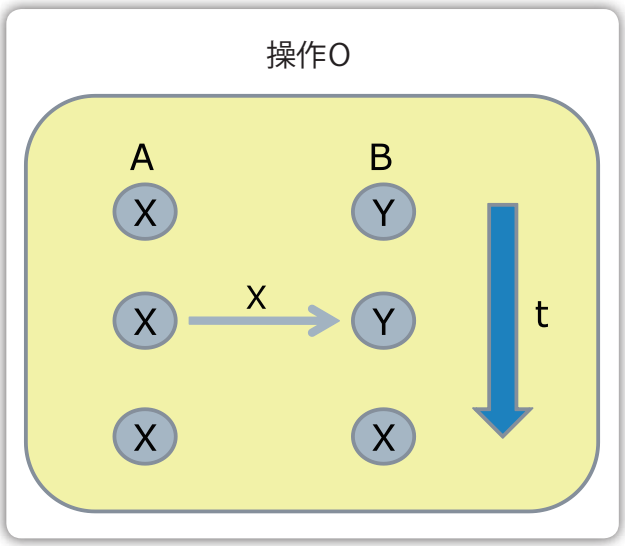


図 -3　情報伝搬に要する時間

けだから，その経済的メリットは大きい．

第 4 に，Azure の利用者が，Azure のシステムに責任を持つのは，前述の「サービス・モデル」とアプリケーションに関してのみである．それ以下の，システムとネットワークの管理に利用者は責任を持つ必要がない．このことによって，システム管理コストだけでなく，開発者をより重要な仕事に投入できるようになる．

Eventually Consistency

Azure は，大規模分散システムであるクラウドでのエンタープライズ利用の問題を通じて，情報処理の基礎にある基本的な問題に新しい照明をあてた．その 1 つが，データの整合性についての Eventually Consistency という概念である．

ネットワーク上のノード A とノード B に同じ情報を持たせるという問題を考えてみよう．ノード A は情報 X を持ち，ノード B は情報 Y を持っているとしよう，ノード A が情報 X をノード B に送って，ノード B の情報を Y から X に変え，ノード A とノード B の情報の同期を取るという基本的な操作 O を考えてみよう．問題は，この操作 O による状態遷移は瞬間的に起きるわけではないということだ．情報 X は，最良の場合でも，光のスピードでしか B に伝わらない．A から B に光が届くまでの時間を t としよう．t は，一般には小さいが 0 ではない．この t が経過するまでは，A と B の情報の同期は完了しない．この中途の期間では，操作 O が保障すべきデータの整合性は破れたままである．逆に，時間 t が経過すれば，データの整合性は実現され得る．

このように，一定の時間の経過の後に，結果的に保証される整合性を，Eventually Consistency という．通常の Consistency は，t=0 のケースだと思えばいい．クラウドのように，数千・数万の分散したノードに情報が存在するとき，データの整合性の問題を扱うには，Eventually Consistency の概念が必要となる．留意してほしいのは，Eventually Consistency は，先の思考実験を見れば分かるように，整合性を弱めたものではなく，整合性概念の原理的限界と基礎を示すものだということである．それは，物理学での特殊相対論による「同時性概念」の変革に対応するものである．

Eventually Consistency と 2-Phase Commit

この思考実験を別の角度から眺めてみよう．先の操作 O の適用範囲はどこまでかという問題を考えてみよう．基本的には，操作 O は，ノード A から情報 X が伝わってノード B に届くまでの全過程を含むという考えがあり得る．O の操作は，A が情報 X を送り出し，B が情報 X を正しく受け取ったときに終わると考えるのである．この考えと先のアプリオリな同時性の放棄の考えを結び付けると，最低限，時間 t の間，操作 O はノード A とノード B を監視する必要があることになる．Eventually Consistency の考えは，時間ゼロで可能な自明なデータ同期の問題を，分散トランザクションの問題に変えてしまうのだ．

こうした場合，従来の分散トランザクションの手法に従えば，操作 O を代表する監視ノード O を，トランザクション・マネージャとしてネットワーク上に追加配置するのが自然である．O は，状態遷移の進行が始まると，まず，他からの干渉をシャットアウトして，このトランザクションをブロックする．ついで，A から情報 X を受け取り，ノード Y から情報 X が届いたという通知を待つ．通知が届いたら，B の状態を Y に変えるように B に指示して，トランザクションのブロックを解く．もっとも，これは処理を単純化した流れであって，実際には，さまざまなケースを考えなければいけなくなる．分散トランザクションの 2-Phase Commit（2PC）のプロトコルは，そもそも，複雑である上，トランザクションのあいだ中，全域的なロックを悲観的に掛け続ける必要がある．

ただ，実は，瞬時に可能な，自明なデータ同期通信は存在しないという考えを 2PC に導入すると，2PC プロトコル自体が，不可能になるのだ．なぜなら，先の例では，当初 A，B2 つのノード間のトランザクション処理のために第 3 のトランザクション・マネージャ・ノード O を導入したのだが，今度は，O と A，B との間のトランザクション処理用の通信も自明なものではなくなるからである．

BASE トランザクション

ここでは，ノード A，B の同期を取る操作 O が，A，B の全域にわたって責任を持つという考えを放棄して，操作 O は，情報を送り出すノード A だけに，ローカルに責任を持つと考えてみよう．ただ，2 つの仮定を導入する．1 つは，ノード A とノード B の間に，確実な情報伝達路 Q が存在して，それが基本的には利用できるという仮定（Basically Availability）であり，もう 1 つは，ノードの状態は，それが結び付いた情報伝達路 Q からの情報によって変化するという仮定（Soft State）である．

こうすると，操作 O は，ノード A の情報 X を，ローカルに情報伝達路 Q に Atomic に書き込んだ時点で，その処理を終えることができる．その後は，情報 X は，一定の時間をかけて，Q を通って B に到達する．B は，Q から得た情報 X で，ローカルに自らの情報を，Atomic に書き換える．こうした過程は，Eventually Consistency の要請を満たしているのは明らかである．クラウドでは，こうしたタイプのトランザクション処理を，Basically Availability と Soft State と Eventually Consistency の頭文字を取って，BASE トランザクションと呼ぶ．

Azure は，クラウドのような大規模分散処理環境での BASE トランザクションの重要性を，初めて明確に意識したシステムである．Azure では，基本的に利用可能な確実な情報伝達路に，永続性を持つ Queue を採用した．Queue は，Azure の基本的なストレージ・システムの一部として位置づけられている．Azure の Queue は，分散ノード上で並列動作する，複数の，m 個と n 個のプロセスの間の同期に，Optimistic Concurrency Control（楽観的並列性コントロール）の手段として，本質的な役割を果たしているのであるが本稿では，その説明は割愛する．

クラウドの Key/Value データ・ストアの実装に，Scalability と Availability の要請を満たすために，P2P 起源の DHT 技術が利用されているのも特徴的であるが，そうしたクラウドでの DHT 利用の技術的な特徴については，本特集中の論考を参考にされたい．

おわりに

クラウドへの流れは，インターネット以降に形成された IT 技術の本流に根差すものであり，同時に，理論的にも，情報技術の基礎についての深い洞察を求めるものであることを述べてきた．クラウド技術の前進が，IT 技術の新しい発展の画期となると，筆者は確信している．

参考文献

1）丸山，浦本，石田：Cloud Computing の世界― Google の分散処理技術―，電子情報通信学会誌 (2008).
2）丸山不二夫：クラウドの技術的特徴― Scalability と Availability，高速分散データベースを中心に，UNIX マガジン誌 (Apr. 2009).
3）丸山不二夫：クラウドとモバイル・デバイス，ASCII Technology 誌 (Oct. 2009).

（平成 21 年 9 月 12 日受付）

丸山不二夫
fujio.maruyama@gmail.com
早稲田大学大学院情報生産システム研究科客員教授．専門：情報教育．日本ソフトウェア学会会員．著書「UNIX データベース論」,「P2P for Java/JXTA」,「情報メディア論」,「Grid for Java」,「Grid/WSRF とビジネスプロセスの統合」他．