応般

4 ビッグデータとのつきあい方
―ビッグデータ活用のための技術と応用基盤―

原　隆浩 ■ 大阪大学

ビッグデータ時代の到来

人やモノから多様な種類の大量のデータが日々生成されており，「ビッグデータ」時代がまさに到来している．ビッグデータを解析することで，さまざまな有益な情報を抽出・生成できる可能性があるため，産業界および学術界において非常に注目されており，活発に研究開発が行われている．

ビッグデータに関する研究開発は，その重要性から，多くの国において政府主導で戦略的に推進されている．たとえば，米国政府は2012年3月29日に“Big Data Research and Development Initiative”を発表し，研究開発を強力に支援している．さらに，欧州委員会の第7次研究枠組計画（FP7）においてビッグデータに関する多数の研究プロジェクトが推進されている．また，中国でも国家戦略として，IoT（Internet of Things）を中心とするビッグデータの研究開発に注力している．我が国でも，総務省が平成25年版の情報通信白書において，「『スマートICT』の進展による新たな価値の創造」を課題に挙げ，ビッグデータ活用の重要性について記載している．さらに，日本政府は，ビッグデータの活用のために，個人情報保護法を中心とした法整備にとりかかっている．

★ ビッグデータの応用例

2012年の米国大統領選挙でObama大統領がビッグデータを活用して，勝利を収めたことはよく知られている．大量のデータを収集し，選挙の情勢や各地域の住民の嗜好・特性を正確に把握し，有効な選挙戦略を立てたことが勝因の1つといわれている．このように，大量のデータを解析することで，これまでは不可能であった，対象となる事象・物の詳細なモデル化，現状の詳細な把握・追跡，さらには将来の予測が可能となる．そのため，ビッグデータの解析を用いたさまざまな新たな応用が研究開発されている．

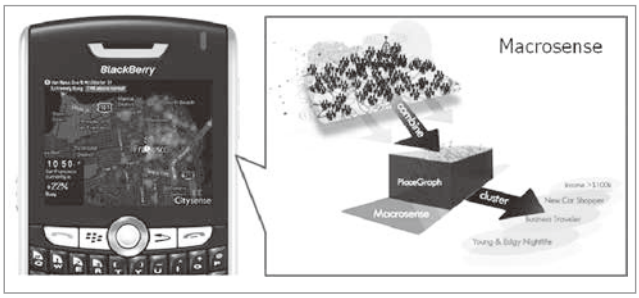

図-1　Citysense

最も典型的な例として，Webサービスにおけるユーザの利用履歴（購買履歴など）やクリックスルー（操作・閲覧履歴），レビューの大規模な解析によるマーケティング戦略，サービスの改善などが一般的に行われている．また，スマートフォンなどに搭載されているセンサから生成されるデータを大規模に収集・共有する「参加型センシング」が近年注目されており，その解析，サービス応用もビッグデータ応用例の1つである．Citysense[☆1]は，Sensor Networks社（2014年1月にYP社が買収）が2008年に開始した実サービスであり，参加型センシングによるビッグデータ解析を最初期に商用利用したものの1つである．CitySenseではスマートフォンなどを持つユーザを対象とし，San Franciscoのユーザ分布をユーザの端末に表示する（図-1）．たとえば，「今，どこに人が集まっていて，どこに行こうとしているか?」などの問合せが可能となる．このために，同社のMacroSense（位置データを用いたユーザ動作解析ツール）を利用し，現在の数万単位のユーザの位置情報と，過去の数10

☆1 http://www.sensenetworks.com/

億のデータを利用して，現状をリアルタイムに予測している．

★ビッグデータ処理のための技術

それでは，上記のような応用を実現するためには，どのような技術が必要となるだろうか？

まず，M2M（Machine to Machine）などでは各種センサや監視機器などから連続的に発生するデータ（データストリーム）を，アプリケーションなどの要求に応じて取捨選択し，加工するなどしてデータベースに格納する．さらに，異常検出などのモニタリングアプリケーションでは，データの到着時にリアルタイムで，特定の値やパターンに合致するデータを発見する必要がある．そのために，データストリームを効率的に処理する技術が必要となる．

さらに，ユーザや車両が絶え間なく生成するセンサデータの解析や，Web サービス等におけるユーザの利用履歴，操作情報の解析は，いずれも大規模になるため，単一のサーバで処理を行うのは現実的ではない．そのため，効率的な大規模分散処理の仕組みが必要となる．

生成される大規模データに対する処理だけではなく，データを効率的に蓄積・管理するデータベース技術も重要となる．これまで，大規模データの管理には，関係データベースと SQL を用いることが一般的であったが，データやその発生源，用途が多様化した現在では，十分な機能・性能を得られない場合がある．

上記の 3 つはビッグデータ処理のための基盤技術であるが，最近では，これまで各機関・企業・個人が個別に収集・管理していたビッグデータを公開し共有することの重要性も謳われている．そのため，ビッグデータの共有を可能とする応用基盤の構築が急務である．

次章以降では，上記の 4 つの技術について，代表的なものを紹介する．

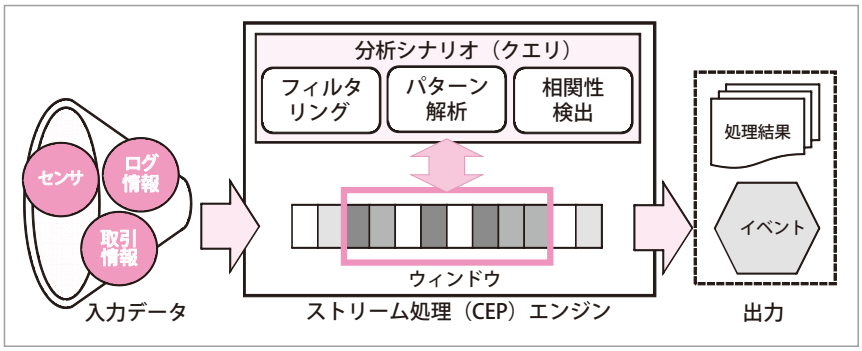

図 -2　ストリーム処理（CEP）エンジンの構成

ストリーム処理技術

ストリーム処理に関しては，2000 年前後からデータベース分野を中心に，単一もしくは複数のデータストリームに対して指定した条件に合致したデータを効率的に発見するための問合せ処理や，特定のパターンに類似した時系列データをストリームから発見するための類似検索，類似性に基づいて時系列データをグループ化するクラスタリングなど，さまざまな研究開発が行われている．

ビッグデータへの注目に伴い，ストリーム処理の重要性も高まっており，複合イベント処理（CEP：Complex Event Processing）と称して，各種ベンダがさまざまな製品を開発している．たとえば，オラクルの Oracle CEP，サイベースの SAP Sybase Event Stream Processor，日立製作所の uConsminexus Stream Data Platform などがある．

図 -2 は，ストリーム処理（CEP）エンジンの一般的な構成を表している．入力データとしては，ユーザの位置情報や各種センサデータ，クリックスルーやクエリログ，システムログ，さらに金融情報の場合は入出金・取引データ，株価などさまざまなものが，短い時間間隔で定期的もしくは不定期にシステムに投入される．

データストリームは通常，絶えず（半永久的に）到着するため，すべてのデータを蓄積するのではなく，到着するたびに処理が行われる．この際，最近の一定数もしくは一定時間に到着したデータ（ウィンドウ）を単位として処理を行うウィンドウ演算の機能を持つものが多い．さらに，ユーザは，到着したデータに対して，どのように処理・分析を行うか（分析シナリオ）

を，問合せ（クエリ）としてシステムに指定する．これは，従来のデータベースにおける単発的な問合せとは異なり，データストリームに対して連続的に（絶えず）実行されるものであるため，「連続問合せ」と呼ばれる．一般的には，データストリームのすべてではなく，問合せで指定した条件に合致するもののみがデータベースに蓄積される場合が多い．

上記の連続問合せには，到着したデータの取捨選択（フィルタリング）や，特徴パターン解析，時系列データの類似性判定，相関性抽出など，さまざまなものが含まれる．問合せの記述言語としては，Stanford 大学で開発された CQL（Continuous Query Language）[1] が有名である．CQL は，SQL に似た宣言的な記法で，問合せを記述できる．

大規模分散処理技術：Hadoop

ビッグデータを効率的に分散処理するフレームワークとしては，Apache ソフトウェア財団がオープンソースとして公開・開発している Hadoop[☆2] が有名である．Yahoo! や Facebook，楽天など多くの IT サービスプロバイダが，Hadoop を用いてログの集計・解析などさまざまな業務上のタスクを実行している．Hadoop は主に，Hadoop 分散ファイルシステム（HDFS）と MapReduce エンジンから構成される．MapReduce は，Google によって 2004 年に提案された分散処理のためのプログラミングモデルである．

Hadoop の MapReduce は，HDFS に格納された大量のデータに対して，複数のサーバを用いて一括処理（バッチ処理）を行い，その結果をファイルとして HDFS に書き込む．HDFS はこのような処理に対して，データ転送のスループットを高めるように設計されている．

★ HDFS

HDFS のシステム構成の概要を図 -3 に示す．HDFS は基本的に，ネームノードと呼ばれる管理サーバと，複数のデータノードと呼ばれるデータサーバから構成

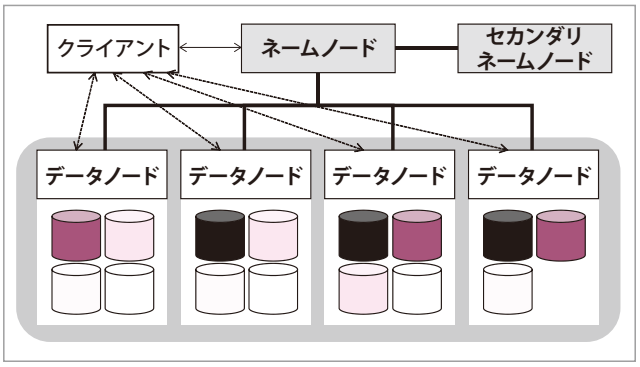

図 -3　HDFS の構成

される．データは，指定されたサイズ（デフォルトは 64MB）のデータブロックに分割され，障害時に備えて指定数（デフォルトは 3）に複製されてデータノードに配置される．この配置場所はネームノードが決定する．データノードの追加・離脱の際も，ネームノードがデータの再配置を決定する．ネームノードはクライアント（たとえば MapReduce プロセス）からの処理依頼に応じて，データの読み書きをどのデータノードに行うかを決定し，クライアントに伝える．実際のデータのやりとりは，クライアントと該当するデータノード間で直接行われる．

HDFS では，データをブロック単位でハードディスクの連続領域に書き込み，一度書き込んだデータの上書きを許可しない（追加は可能）ことで，データのまとまった読出しを高速化している．

HDFS では，ネームノードが単一障害点となるため，障害時に備えて，バックアップのためにセカンダリネームノードが配置される．

★ MapReduce

MapReduce では，HDFS に格納されているデータに対する並列処理を，Map 処理と Reduce 処理の 2 段階で実行する．Map 処理では，データを小さな単位に分割し，大規模数のサーバにそのデータに対するタスクを割り当てる．この際，無駄なデータ転送を削減するために，各データを所持するサーバ（データノード）に該当するタスクを割り当てることが一般的である．各サーバは担当のタスクを実行し，中間結果を出力する．この中間結果は，Reduce 処理の各タスクを実行するサーバに転送される．Reduce 処理では，

☆2 http://hadoop.apache.org/

Map 処理の結果（中間結果）を集約し，最終的な結果を得る．最終結果は，HDFS に書き込まれる．

MapReduce の処理の例を図 -4 に示す．この例では，大量の Twitter やアンケート情報の中から地名に関する単語を抽出し，各地名の出現回数をカウントしている．まず，Map 処理において，各サーバ（A, B, ,…）は割り当てられた全データから地名を抽出する．これが Map 処理の出力結果であり，中間結果となる．そして，抽出した地名のリスト内の各項目（地名と出現回数の組）を，Reduce 処理のタスクを実行するサーバ（Q, R, …）へ送信する．この際，地名をキーとして，五十音順で一定の範囲のキーが，各サーバに割り当てられている．Reduce 処理では，各サーバは自身に割り当てられた各キーの出現回数を集約し，最終結果としている．

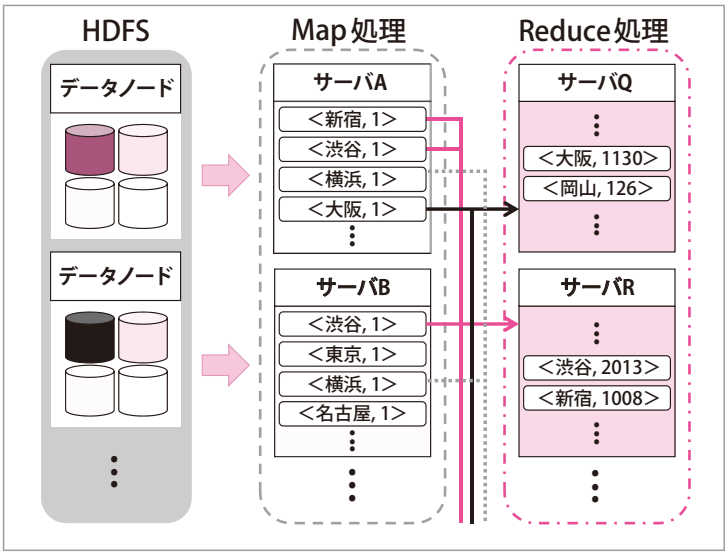

図 -4　MapReduce 処理の実行例

NoSQL

HDFS は，MapReduce によるバッチ（分散）処理を効率的に行うためのファイルシステムであり，基本的にその処理に一度だけ発生するデータ読出し・転送の性能を最大化することを目的としている．一方，蓄積されたビッグデータに対して，（分散処理だけではない）さまざまなデータ処理を効率的に実行することが求められている．この際，従来の関係データベースだけでは十分な性能が出ない場合が多いことから，“Not Only SQL（NoSQL）”と称して，多種多様なデータベースが開発されている．つまり，NoSQL はある特定のデータベースのことを指すものではない．NoSQL では，蓄積されたデータベースに対する頻繁なデータ読出し・書込みや，その整合性などが考慮されている．なお，HDFS や MapReduce 自体は NoSQL とは呼べないが，HDFS 上に NoSQL を構築することは可能である．実際に，HBase がその代表例である．また，MapReduce は，HDFS 上に構築された NoSQL に対して効率的な分散処理の機能を提供するための，一種のイネーブラとして捉えることも可能である．以下では，NoSQL が登場した背景と特徴について紹介する．

★ NoSQL が登場した背景

関係データベースは長年にわたり商用システムの根幹を支えてきたが，その長所である「スキーマ定義がしっかりしている」「正規化と結合処理によって無駄や情報欠損がなく表形式で対象を表現できる」「処理の整合性や複製間の一貫性が保障される」といった特徴が，ビッグデータの処理では弊害となる場合がある．その例を以下に示す．

問題①：さまざまな情報源からデータが生成される環境では，あらかじめスキーマを定義するのが困難な場合が多く，柔軟性・拡張性の面で問題がある．

問題②：正規化によって，対象を複数の表に分けて表現し，結合処理によって再現することは，ビッグデータでは負荷が非常に大きい．さらに，ビッグデータの分散処理のために，データを水平分割（タップル・レコード単位で分割）して大規模数のサーバに分散配置することが多いが，まず正規化によって分割された表を結合する必要があり，このための負荷が膨大となる．

問題③：大規模な種類のデータに対してその複製を大規模数のサーバに分散配置することが多いため，それらの一貫性（バージョン）を完全に保障すること自体の負荷が非常に大きく，システム全体の性能を低下させる場合がある．

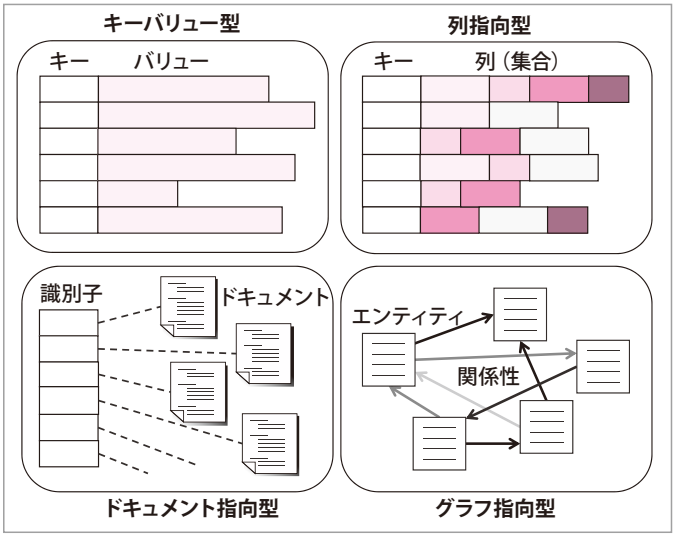

図 -5　NoSQL の分類と構造のイメージ

以上のような問題点を解決するために，さまざまな NoSQL が開発されている．

★ NoSQL の分類と特徴

NoSQL の分類にはいくつか異なる方法があるが，キーバリュー型，列指向型，ドキュメント指向型，グラフ指向型に分類するものが多い[2]．各型のデータ構造のイメージを図 -5 に，代表的な商用およびオープンソースの NoSQL を表 -1 に示す．以下では，各型の特徴について，上記の問題と対応付けて議論する．なお，問題③については，各 NoSQL によって対応がさまざまであり，想定するアプリケーションに応じて，一貫性を軽減したりしている．

キーバリュー型

キーバリュー型は，データをキーとバリュー（値）の組という単純な構造で表す．バリューがデータ本体で，キーがその識別子のようなイメージである．バリューで扱うデータは文字列などさまざまであるが，構造を持つデータもバリューとしてひとまとめになるため，構造を意識した処理には対応できない．そのため，キーを用いてデータを参照し処理を行う，キーに従ってデータをサーバに配置するなどといった単純な処理を対象とする場合に用いられる．構造を意識しないという特徴から，上記の問題①と②を解消している．

列指向型

列指向型は，キーバリュー型を拡張して，バリューに複数の列（カラム）を持てるようにしたものである．ただし，関係データベースとは異なり，データごとに列の数や属性（列名）を動的に決定できる．キーバリュー型の特徴をおおむね継承しており（そのため問題①と②を解消），さらに，カラムの属性を指定したデータの読み書きや検索，処理が可能となる．

分類	代表例
キーバリュー型	Amazon Dynamo, Redis, ROMA
列指向型	HBase, Cassandra, Hypertable
ドキュメント指向型	MongoDB, CouchDB, Terrastore
グラフ指向型	Neo4j, AllegroGraph, Sones

表 -1　NoSQL の分類と代表例

ドキュメント指向型

ドキュメント指向型は，XML や JSON などの記述言語によって記述されたドキュメントをデータとして扱うことが可能であり，各データには一意の識別子が付与される．前述の 2 つの型と同様に，上記の問題①および②を基本的に解消している．該当する記述言語の仕様に従い，各データへの読み書き・処理等を行うことが可能である．

グラフ指向型

グラフ指向型は，対象を複数のノード（エンティティ）とノード間の枝（関係性）としてグラフ構造で表現する．エンティティと関係性は属性を持ち，それぞれ具体的にどのようなものかを表している．グラフ指向型の NoSQL はたとえば，ソーシャルネットワークサービス（SNS）におけるユーザ同士の関係（グラフ構造）の解析などに用いられる．関係データベースでも，データ間の関係を複数の表の共通属性などで表現できるが，結合処理等によって再現する必要がある．グラフ指向型は，このような関係を直接的に表現できるため，上記の問題②を解消できる（問題①を解消できるかはノードの表現方式に依存する）．

情報銀行

ここまでで，ビッグデータを扱うための基盤技術について紹介したが，ビッグデータの恩恵を最大限に活

用するためには，さまざまな企業・機関および個人が個別に収集・管理（独占）している大規模なデータを共有可能な応用基盤の確立や法整備が急務と考えられる．法整備については，各国において，プライバシーを保護しつつ，匿名化した個人情報であれば第三者に提供可能にするといった法改正等が検討されている．

応用基盤については，情報通信研究機構の K-L Grid などセンサ，科学，言語，音声，文書データを共有するためのプラットフォームが多数構築されている．しかし，近年のスマートフォンユーザが生成する位置情報やセンサデータなどを大規模に収集・管理し，プライバシーを考慮した共有を可能とする応用基盤はほとんど存在しない．

ここで，「情報銀行」と呼ばれる興味深いコンセプトが東京大学・柴崎亮介教授らによって提唱されている[3]．これは，個人が自身の管理下でさまざまな活動情報（行動履歴，購買履歴，予定，既往歴など）を統合し，承認の上で「情報銀行」に預け，運用を委託するというものである（図 -6）．これにより，個人の管理・制御の下でプライバシーを保護しながら，大規模な個人情報を社会・産業のために高次利用することが可能となり，価値創造のサイクルを実現できる．つまり，ビッグデータの恩恵を最大限に活用することが可能となる．具体的には，以下のような利点が生まれる．

- 個人が，自身の情報がどのように使われているのか，意図に反して使われていないかを監視できる．さらに，使われ方を制御できる．
- 事業者としては，これまで自身の事業に関する断片的な情報のみしか利用できなかったが，複数サービスに関する情報をユーザの管理下で横断的に利用できるようになる．
- 個人としては，サービス品質の向上が期待できる．社会・産業的には，有益な情報・新たなサービスの創生が期待できる．

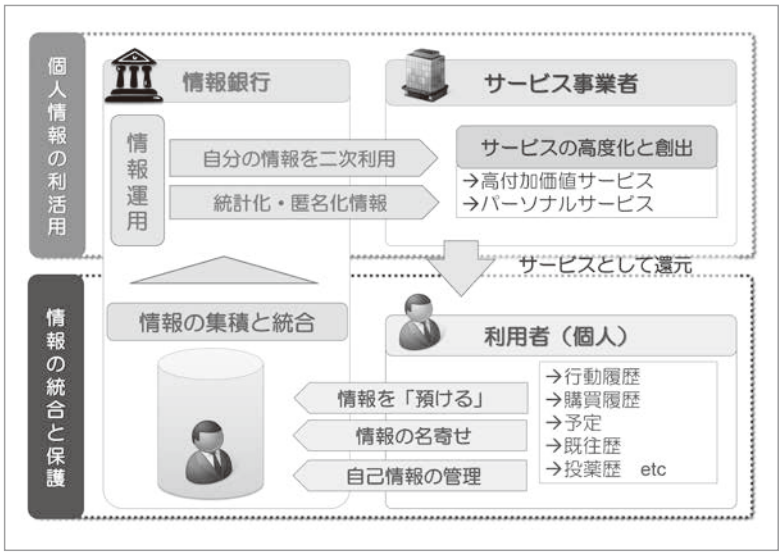

図 -6　情報銀行における価値創造のサイクル（http://shiba.iis.u-tokyo.ac.jp/research/ibank/pdf/2011.iBank-Poster.pdf より引用）

今後の展望

本稿では，ビッグデータを活用するための，ストリーム処理，大規模分散処理，データベースについて，それぞれ代表的な技術を紹介した．後者の 2 つについては，たとえば Hadoop を簡単に利用可能な機能を有する NoSQL データベースもいくつかあり，すでにある程度の連携が行われている．しかし，ストリーム処理エンジンと後者 2 つの連携や，さらにそれらを多段に連携させて，大規模データストリームをリアルタイムに分散処理する基盤や，その結果等をクラウド上で新たなサービスとして共有可能とする基盤については，その有効性が認識されているものの，実用的なフレームワークはまだないのが現状である．このようなフレームワークの構築が今後の重要課題の 1 つと考えられる．

参考文献

1) Arasu, A., Babu, S. and Widom, J. : The CQL Continuous Query Language : Semantic Foundations and Query Execution, The VLDB Journal, Vol.15, No.2, pp.121-142 (2006).
2) 太田　洋 監修，本橋信也，河野達也，鶴見利章 著：NoSQL の基礎知識―ビッグデータを活かすデータベース技術，リックテレコム（2012）．
3) 情報銀行コンソーシアム（仮称），http://www.information-bank.net/

（2014 年 6 月 18 日受付）

原　隆浩（正会員）hara@ist.osaka-u.ac.jp

1997 年大阪大学大学院前期課程修了．2004 年より同大学院情報科学研究科准教授．工学博士．2008 年，2009 年本会論文賞受賞．データベースシステム，モバイルコンピューティングに関する研究に従事．IEEE，ACM，電子情報通信学会，日本データベース学会各会員．