大規模データ処理アーキテクチャの動向と展望

Trends and Prospects of Large-scale Data Processing Architecture

中 山　陽 太 郎

要　約　データベース技術は，対象とするデータの特性に応じて発展を続けている．Web 上に集約される大規模データは，それまでのデータ処理技術の枠に収まらないビッグデータとして認識され，データ処理技術の多様化を引き起こした．Web 系企業においては，2000 年代初頭より大規模データ処理の課題が認識され，Google における分散データ処理基盤のクローンとして登場した Hadoop は一つの解決を示したが，さらに複雑化するビッグデータ分析の要求に対応するものとして，2010 年頃より分散インメモリ処理による Spark が登場している．また Web スケール・アプリケーションにおけるデータ処理基盤として，NoSQL の利用が広がっている．一方，従来の RDBMS においても，大規模トランザクション処理や H/W リソースの発展への最適化が求められ，新たなアーキテクチャによるリファクタリングが進んでいる．多様なデータ処理技術は，独立に発展しながらも補完し合い，ビッグデータ処理のプラットフォームとして，融合されたアーキテクチャを形成している．

Abstract　Database technology has continued to develop in accordance with the characteristics of the data of interest. Large data aggregated on the Web is recognized as big data that does not fit in the frame of data processing technology so far, it led to diversification of data processing technology. In Web companies, it is recognized challenges of large-scale data processing from the early 2000s around, although Hadoop has emerged as a clone of distributed data processing infrastructure in Google showed one of the resolution, request of big data analysis to be more complex as corresponding, from around 2010, the Spark by distributed-memory processing has appeared in. On the other hand, even in a conventional RDBMS, the optimization is required for large-scale transaction processing and/or improvement of H/W resources and it is progressing refactoring due to a new architecture. A big data processing platform while they are developed independently but complement each other.

1. は　じ　め　に

インターネットによる Web ビジネスの発展に伴い，人や社会，ビジネスから発信されるデータは膨大なものとなり，ビッグデータと呼ばれる状況を生み出した．Web スケール IT[*1] 企業の成長とともに，インターネット上に集約されるデータ量の大規模化が深刻な問題となり，既存のデータベース技術では対応しきれない状況が顕在化した．単一サーバでは限界のある大規模データ処理に対して，並列分散処理を利用することが Google の論文によって広められ，その実装として登場した Hadoop はビッグデータ処理のための有効なツールとして，Web 系企業を中心に普及が進んだ．近年は，自動車や家電，工具，医療機器など，多様なデバイスや機器がインターネットに接続することで，より高度なサービスを提供する IoT（モノのインターネット：Internet of Things）技術が急速に拡大することが予測されている．IoT の普及により，Web 上に収集され処理されるデータ量はさらに増大しビッグデータとしての利用が多様化す

ると考えられる．

また，複雑なデータ処理における Hadoop の課題が認識され，それを克服する新たな技術として Spark が登場し，次世代のデータ処理プラットフォームとして注目を集めている[*2]．さらに誕生から既に 40 年以上を経たリレーショナル・データベース管理システム（RDBMS）は，Web スケール IT の発展やハードウェア（H/W）の進化を背景として，さらなる最適化が求められるようになり，基底アーキテクチャの見直しが始まっている．分散システムにおける計算リソースの性能向上とその利用が容易となった状況において，ビッグデータとして共有された大規模データに対する問題意識が，データベース技術の大きな変革を引き起こす要因になっている．

本稿では，ビッグデータが一般にも認識されるようになって以降，これまでにどのような技術革新が起こっているのか，データ利用の多様化や技術環境の変化の文脈に照らし合わせ，データベース技術動向を俯瞰するとともに，将来の技術発展の展望について考察する．2 章では，データベース技術の多様化の背景と課題，現状の動向について説明し，3 章では，Web スケール IT の発展を背景として登場した Hadoop から Spark への展開について解説する．4 章で，NoSQL の分散システムにおける制約やトランザクションの特性について述べ，5 章では，従来の RDBMS のアーキテクチャの限界と，新たな RDBMS である NewSQL について説明する．6 章では，将来のデータベース技術の展望について，ハードウェアの進化への対応及び，データ統合やデータストア連携の観点から解説し，7 章でまとめを述べる．

2. データベース技術多様化の背景

Web 上のデータの大規模化に伴うデータ処理効率の課題は，2000 年代初めより認識され，大きく二つの革新的な取り組みが始まったと見ることができる．一つは，RDBMS の初期より研究開発に取り組んできた Michael Stonebraker[*3] を中心とする研究プロジェクトであり，もう一つは，Google や Facebook，Amazon などの Web スケール IT 企業における取り組みである．

Stonebraker は，従来の RDBMS では扱うことができないストリームデータや大量トランザクションのトラフィックなどに対応することを目的として，既存の RDBMS のアーキテクチャを書き直すことを提唱した．一方，Web スケール IT 企業による取り組みは，大規模データ処理基盤として Google が開発した分散ファイルシステム GFS（Google File System）と分散並列フレームワーク MapReduce の論文公開を基点に，そのクローンとして開発された Hadoop が Apache Foundation のもと OSS として公開されたことで広まった．Hadoop の登場以後，大規模分散データ処理は，ビッグデータを扱うデータ処理技術として大きな進展を遂げることになった．

2.1 ビッグデータにおける課題

ビッグデータ処理の課題として，大量なデータの蓄積，移動，データベースへの格納における処理時間の増大があげられる．Hadoop 上に蓄積された大規模データソースを RDBMS で利用する場合，大量のデータ転送とデータロードに多大なネットワーク，メモリ，ストレージのリソースが必要となる．また，複数データソースの統合や結合のため，処理の複雑化と共に処理時間も増大する．これらのことから，データソースとデータ処理とを密接に連携させ，可能な限りデータの移動（Data Moving）を排除するための仕組みが重要であると認識されるようになった．大規模，複雑化するデータと最適化された処理を直接連携させることにより，データの不要な分散や複製を排除し，データ処理を効率化することが可能となる．また，今後のサー

バアーキテクチャの進化により，ハードウェアとソフトウェアをより密接に連携させることが，大規模データにおける効率的な処理のために重要であると考えられる．

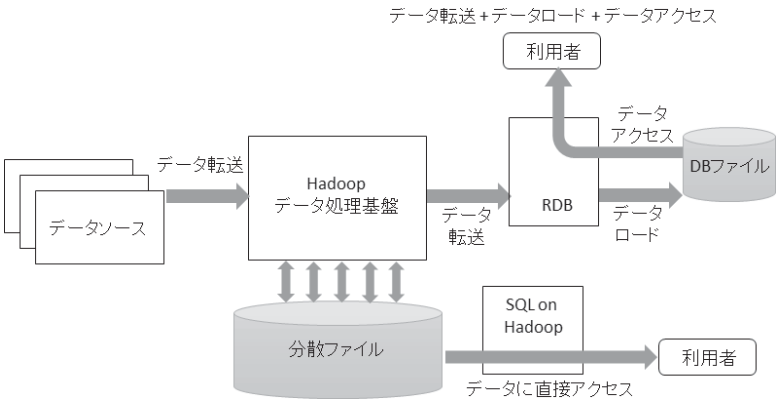

図 1　データソース，Hadoop，RDBMS の連携

例として，図 1 に大規模データ処理のイメージを示す．Hadoop 上に蓄積された大規模データソースを RDBMS 側で処理するためには，データを転送しデータロードする必要がある．それに掛かる処理負荷はディスクへの大規模なアクセス，ネットワーク上のデータ転送，RDBMS へのデータローディングの総計であり，目的とするデータ分析自体より大きな比率を占める．

データの移動負荷をなくし，蓄積された分散ファイルに対して直接処理を行う仕組みを実現するものとして，Hadoop 上で動作する SQL on Hadoop と呼ばれる SQL ミドルウェアが登場している．SQL on Hadoop は，事前のスキーマ定義が不要であり，蓄積されたデータの形式やクエリに応じてスキーマを定義することが可能である．このため，データロードが不要となり，データベースの格納処理の課題に対する一つの解決と考えられる．SQL on Hadoop については，3. 3 節で説明する．日本ユニシスの取り組みとして，RDBMS から Hadoop 上のデータを直接 SQL によって処理するための連携について 6. 2 節で述べる．

2. 2　RDBMS の課題とリファクタリング

インターネット技術の普及は，ビジネスにおける Web アプリケーションの多様な形態の出現をもたらした．e コマースサイトにおける商品の推奨や広告，商品の評価，ソーシャルデータとの連携など，頻度・粒度・構造の異なるデータの連携が求められ，これまでのビジネスにおけるデータ処理とは異なる特性への対応が必要となっている．これらの新たなデータ処理の例として以下がある．

- Web スケール・アプリケーションにおける大量トランザクション
- 時系列ストリームデータ，センサー・データ，株価変動データ，クリックストリーム
- 大規模データに対する高速な集約，集計，分析処理，機械学習

既存の RDBMS の基本的なアーキテクチャは，1970 年に IBM の Edgar F. Codd によるリレーショナル・データベースの論文[2]を基に開発された System R*4 をルーツとしており，ビジネスデータを対象とするものであった．

Stonebraker は，多様化するデータの特性に対して，従来の RDBMS の単一的なアーキテク

チャだけによって対応することは適さないとし，データ処理の特性に応じてRDBMSエンジンをリファクタリングする考えを提唱した[1]．その実践として，OLTP，データウェアハウス（DWH），ストリームデータ処理のそれぞれの用途に特化したデータベース・エンジンの研究開発を推進した．表1に用途と研究名称を挙げる．

H-Storeは，NewSQLと呼ばれ，Webアプリケーションの高速大量トランザクション処理に対するOLTP用途のRDBMS研究である．C-Storeは，カラム指向のデータ形式をサポートしたデータウェアハウスである．Auroraは，連続するストリームデータ処理のためのデータストリーム管理システム（DSMS）である．これらはいずれも研究開発されたソフトウェアを基にベンチャー企業が興され商用化されている．それぞれの特徴については，5章で説明する．

表1　データベース研究プロジェクト

用途	研究名称（製品名・企業）
並列分散OLTPデータベース・エンジン	H-Store（VoltDB・VoltDB）
カラム指向データウェアハウス	C-Store（Vertica・HP）
データストリーム管理エンジン	Aurora（StreamBase・TIBCO）

3.　大規模分散データ処理

3.1　Hadoopにおける並列データ処理

Hadoopは，Googleの分散ファイルシステムGoogle File System（GFS）[3]及び並列処理フレームワークMapReduce[4]に関する論文を基に，Doug Cuttingらにより開発され，OSSとして公開された．その後Hadoopは，Web系企業を中心として急速に普及し，現在も技術的な発展を続けている．図2にHadoopの処理イメージを示す．

Hadoopは分散ファイルシステムHDFS（Hadoop分散ファイルシステム）と並列処理フレームワークMapReduceの二つの核となるシステムから構成され，データノードを追加することで，数千台までのクラスタを構成し，テラバイトからペタバイト級のデータを並列に処理することが可能である．2013年10月より，Hadoop2.0と呼ばれるYarn（Yet Another Resource Negotiator）による汎用化が進められ，MapReduceに限定されていた分散処理は，任意の分散処理フレームワークに対応が可能なアーキテクチャとして汎用化された．

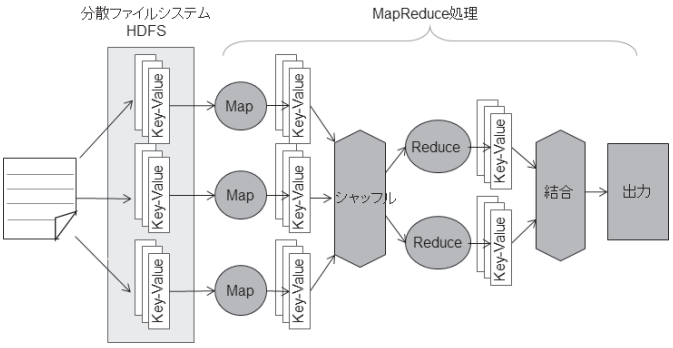

図2　Hadoopによる分散処理のイメージ

3. 2　複雑な大規模データ処理技術としての Spark

Hadoop は大規模データ処理において革新的な解決をもたらし，ビッグデータの利用の流れは，さらに複雑なデータ分析や機械学習への利用に対する要求を引き起こした．Hadoop は本来，バッチ処理として利用されることを想定したものである．一回の MapReduce によるジョブで，入出力のディスク I/O が発生するため，機械学習やデータ分析が複雑化することに伴い，ジョブを複雑に組み合わせた多段階処理が必要となり，ディスク I/O の増大が引き起こされた．その結果，データの大規模化と処理の複雑化に伴う処理時間の増大が，課題として認識されるようになった．

この課題を背景として，UC Berkeley の AMPLab[*5] で研究開発された分散処理基盤が Spark である[5]．Spark は，Hadoop と同様に拡張性と可用性を実現すると同時に，複雑化するデータ分析において，効率的なデータ処理を実現した．回復性を備えた分散データセット（Resilient Distributed Dataset：RDD）を特徴とし，RDD をクラスタノード上に分散配置することで，並列分散処理と可用性を実現している．Spark におけるデータ処理では，MapReduce と異なり，ジョブごとにディスクへの入出力を伴わず，複数の処理間でデータをパイプライン的に処理することを可能とする．これにより，MapReduce における多段階処理の効率上の課題を解決している．

Spark の全体構成は，データ処理の核となる Spark Core Engine を基盤として，複数のコンポーネントから構成されている（図 3）．例えば，MLlib（Machine Learning Library）は，機械学習用のライブラリ群であり，SaprkSQL は，Spark に対して SQL によるアクセスを可能とするミドルウェアである．

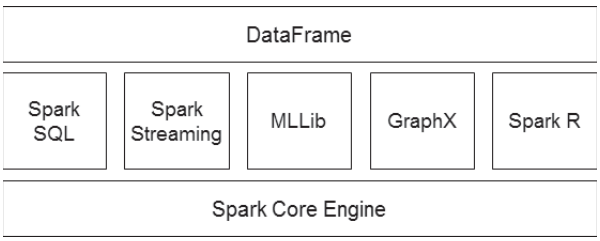

図 3　Spark 関連ソフトウェア

3. 3　SQL on Hadoop

Hadoop の有効性が認識され，企業のシステムにおいても選択技の一つとして検討されるようになると，MapReduce フレームワークによる開発スキルが課題として認識されるようになった．また，Hadoop に蓄積されたデータをインターラクティブに処理したいという要求が高まった．MapReduce フレームワークを隠蔽するものとしては，Hadoop の初期より，SQL ライクなスクリプト言語である Hive[6][7] や Pig[8] が提供されており，MapReduce を意識せずに Hadoop を利用できることで，その開発の容易性が評価されている．しかし，内部的に MapReduce に変換されて動作するため，処理の複雑さに比例して性能が遅延するという課題がある．このため，SQL on Hadoop と呼ばれる Hadoop 上で動作するミドルウェアが多く登場してきている．これらは，MapReduce ではない独自の並列分散処理を採用することで，SQL によるインターラクティブな操作を可能とした．代表的なものに，Impala[*6][9]，Presto[*7][10]，

Drill[*8][11]などがある．最近では，Hive も MapReduce に代わる分散処理 Tez[*9] を採用して高速化を図っている．

SQL on Hadoop の特徴として，独自の並列処理による高速化に加え，分散ファイルに蓄積された CSV 形式のデータソースに対して，事前にスキーマを定義することなくアクセスできることがあげられる．RDBMS のように定義されたスキーマの下でのデータロードが不要であり，配置されたデータソースに対して直接 SQL を実行することが可能である．また，SQL on Hadoop の技術は，従来の RDBMS におけるデータロードを排除することで，ビッグデータの移動や加工時の処理時間の課題を解決した．

4. NoSQL

4. 1 NoSQL 概要

NoSQL[*10] は Not Only SQL の略とされ，従来の RDBMS ではないデータベース・システムの総称とされる．NoSQL のアーキテクチャは，分散システムの実現方式とデータモデルの違いによって分類される．分散システムの実現方式として，マスタ・スレーブ型，P2P 型（マルチマスタ型）に分類される．また単一サーバ型を含める場合もある[*11]．データモデルの違いによる代表的な分類を表 2 に示す．

表 2　NoSQL のデータモデルによる分類

データモデル分類	説明	主要なソフトウェア
ビッグテーブル型	キー・バリューをカラムファミリーというテーブル形式で表現	Cassandra，HBase
キー・バリュー型	単純なデータ構造であり，KVS（キー・バリュー・ストア）と呼ばれる．分散ハッシュメモリ型のものも含まれる．	Dynamo，Riak，Couchbase
ドキュメント指向型	JSON 形式をサポートし，データ構造の柔軟性が高い．	MongoDB，Riak
グラフ型	グラフ型による頂点と辺によるデータ表現	GraphX，Titan，Neo4j

近年の動向として，キー・バリュー型（KVS）のデータ構造が拡張され，ドキュメントとして JSON 型をサポートするものや，ビッグテーブル型や KVS 上で稼働するグラフ型 NoSQL が登場している．また，問い合わせ言語として，Cassandra のように SQL のサブセットをサポートするものが出てきている．米国では，Web スケール・アプリケーションにおけるデータ処理基盤として MongoDB や Cassandra の適用が広がっている[*12]．

4. 2 NoSQL の例：Cassandra

ここでは，NoSQL の特性について，ビッグテーブル型 NoSQL である Cassandra[12][13] を例として説明する．Cassandra は，Amazon の KVS である Dynamo[14] の分散基盤と Google の Bigtable[15] のデータモデルを融合したものとして Facebook によって開発された．拡張性と堅牢な可用性を特徴とし，数台から千台規模のクラスタの構築が可能であり，大規模 Web 系企業のデータ処理基盤として利用が広がっている．すべてのノードがマスターとなる P2P 型のアー

キテクチャにより単一障害点が排除され，ノード間のデータのレプリケーションにより，対障害性と可用性が保障されることが特徴である．

データ操作言語として，CQL（Cassandra Query Language）と呼ばれるSQLのサブセットが提供され，テーブル定義，データの格納，検索，更新などをSQL文によって操作することが可能である．但し，RDBMSと異なり，結合は使用できず，索引についても制約がある．データ構造は，ビッグテーブル型（図4左）であり，行を構成するカラム（Column）とバリュー（Value）のペアと行を識別する主キー（Primary key）からなる．データの分散方式は，コンシステントハッシュ法に基づいており（図4右），ノードの追加によるデータの再配置の負荷はノード数に応じて局所化される．これによってノードの追加を柔軟に行うことができ，拡張性と負荷分散が保障される．

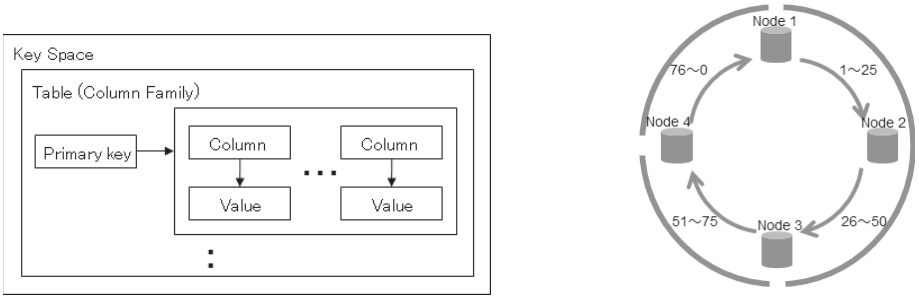

図4　Cassandraのデータモデル（左）とコンシステントハッシュ法の例（右）

Cassandraは，Webスケール規模のeコマースやコンテンツ配信の大規模データ処理基盤として利用が広がってきた．加えて，粒度の細かい大量データの集積性能に優れているため，近年，IoTやM2Mにおけるデータ処理基盤としての利用が増えている．既存システムへの適用として，拡張性や可用性が求められるシステムにおいて，RDBMSの置き換えではなく，機能の代替や棲み分けによりRDBMSと連携し，それぞれの特性を考慮した効率的なデータ処理基盤の構築が期待される．

4.3　NoSQLにおけるトランザクション

分散システムに基づくNoSQLは，CAP定理による一貫性と可用性のトレードオフによって分類される．CAP定理は，分散システムを特徴付ける概念として2000年にEric Brewerの講演[16]で用いられた．一般に分散システムは，一貫性（Consistency），可用性（Availability），ネットワーク分断耐性（Tolerance to network Partitioning）[*13]の三つのうち，同時に二つしか満たすことができないとされる．分散システムにおいて，ネットワーク分断耐性（P）は必須のため，一貫性（C）と可用性（A）のトレードオフとなり，CP型かAP型に分類される[*14]．また，分散システムにおけるトランザクションは，データの一貫性を保障するRDBMSにおけるACID特性[*15]と異なる性質をもつ．分散システムにおいて，可用性を優先する場合，複製されたデータに対する更新は，非同期処理となり，一時的にノード間で一貫性が保障されない状態となる．このようなトランザクション特性は，結果整合性としてBASE[*16]と呼ばれる．

NoSQL の Cassandra は，P2P の分散アーキテクチャであり，CAP 定理として AP 型に分類され，トランザクションの特性は BASE である．しかしながら，QUORUM（議決の定足数を意味する）による一貫性レベルにより，データを読み込むノード数と書き込みノード数を調整することで，一貫性を保障でき，一貫性（C）と可用性（A）の両立が可能である．またトランザクションは，RDBMS のような ACID 特性を完全に満たすことは実現されていないものの，単一の更新処理における楽観的同時実行制御が可能である．アトミック・バッチ（Atomic Batch）と軽量トランザクション（Light Weight Transaction）を組み合わせることで，制約された範囲において ACID 特性を保障したトランザクション処理が実行可能である．

5. SQL から NewSQL へ

5. 1 RDBMS の新たな基底アーキテクチャ

Stonebraker は，従来の汎用的な用途の RDBMS のアーキテクチャの書き直しを提唱し[18]，新たな RDBMS の研究開発を行った．H-Store[*17] は，e コマースやオンライン・トレーディングのような従来のトランザクション量を超える大量トランザクション処理を指向した OLTP 用途の RDBMS である．H-Store では，効率上のボトルネックとなる従来の RDBMS の機能を見直し，新たなアーキテクチャの上で不要となる機構を排除することで，特定のデータ処理に最適化された RDBMS を実現している．RDBMS における性能上の障壁として以下がある．

- 同時実行制御（Concurrent Control）
- 先行ログ書き込み（WAL：Write Ahead Log）
- ロック機構（ラッチ，ブロックレベル，行レベル）
- バッファプール管理

これらの機構による性能上の障壁を回避するため，H-Store では，トランザクション処理をスレッド単位でシーケンシャルに実行する方式を採用している．データを水平分散によりパーティション化し，パーティション内においては，同時に 1 スレッドあたり 1 トランザクションとするため，排他制御が不要となりロックの機構を排除することを実現している（図 5）[*18]．

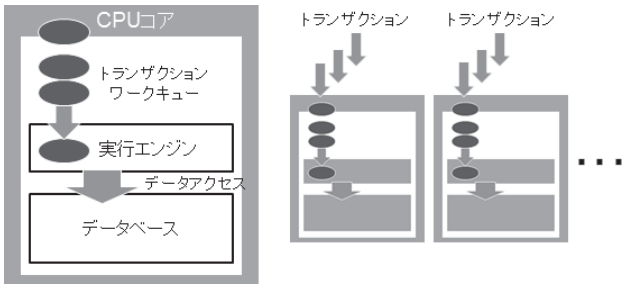

図 5　H-Store におけるトランザクション実行のイメージ

同期レプリケーションによって一貫性を保障し，同時に可用性を保障することで，障害リカバリが簡略化され，Redo ログの機構が不要となる．従来の RDBMS では，Redo ログはトランザクションのコミット時に同期書き込みされるため，性能上の制約となっていた．H-Store ではこれを排除し，可用性と性能向上を同時に実現している．このように，RDBMS における基底アーキテクチャを見直すことにより，従来の性能の課題の解決を図っている．

5.2　カラム指向データウェアハウス

カラム指向の RDBMS は，Stonebraker らによって C-Store[19] として研究された．ストレージ上のデータファイルの形式を従来の行ベースではなくカラム単位とすることで，読み込みの多い OLAP やデータウェアハウスを高速化することができる．C-Store の研究以降，カラム指向アーキテクチャは，既存の DWH 製品に大きな影響を与え，大手ベンダーによる DWH 製品はほぼすべてカラム指向をサポートするようになってきている．図 6 に従来の行指向型 RDBMS のイメージとカラム指向型 RDBMS のデータの読み込み処理のイメージを示す．

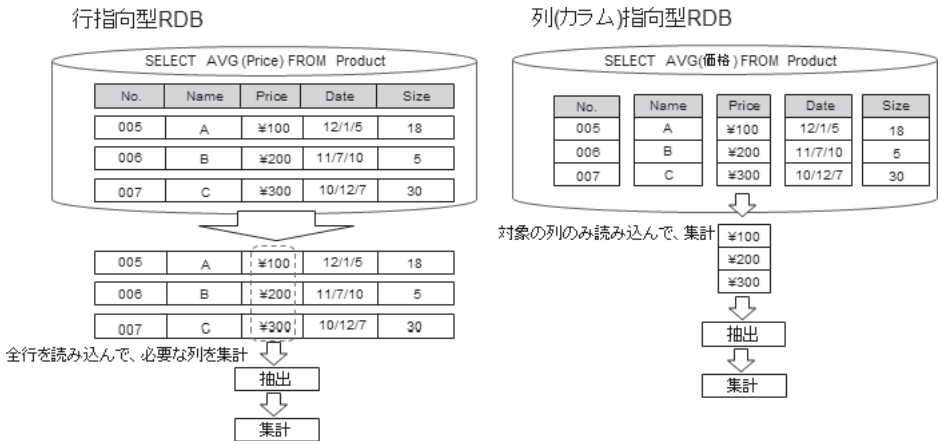

図 6　カラム指向型 RDBMS における読み込み処理のイメージ

カラム指向は，射影演算において必要なカラムのデータだけを読み込むため，不要なカラムも同時に読み込む行型のストレージに比べ，読み込み時のデータサイズを小さくでき，高速処理が可能となる．また，同じデータ型だけを集約したデータ管理により，文字列や数値など，データ型で異なる圧縮技術のそれぞれに適切に適用することが可能となり，行型による管理に比べ，より粒度の細かい最適化が可能である．

5.3　ストリームデータ処理

ストリームデータ処理システム（Data Stream Management System：DSMS）は，センサー機器や金融の株価など，継続的に発信されるデータを，連続的に処理するためのデータ処理基盤である．2000 年代初期より活発に研究が行われ，2010 年頃より商用製品が登場した．DSMS では，入力されたデータをディスクに格納せず，そのままクエリエンジンによって処理する．問い合わせ言語として，連続データ処理のために SQL を拡張した CQL（Continuous Query Language：連続クエリ言語）が用いられる．Stonebraker らは，リアルタイム・ストリーム処理における八つの要件を提案しており[20]，データをストレージに格納することなく処理することや，ストリーム処理に対応した SQL などを挙げている．DSMS は，今後，M2M におけるセンサー・データや IoT 技術が普及することで，利用が進むことが予想される．

また，Web システムの要求から登場したストリームデータ処理技術として，Twitter により開発された Storm や LinkedIn による Kafka[21]がある．Kafka は，Web サービスなどから発せられる大容量のログやイベントを高スループットかつ低レイテンシに収集・配信することを

目的とする分散メッセージングシステムである（図7）.

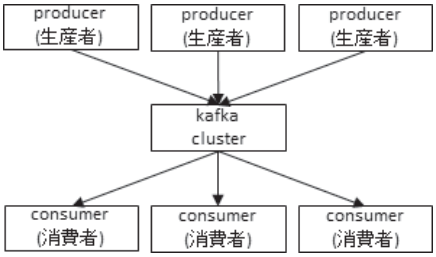

図7　Kafkaにおける通信

これらは，大量かつ高頻度で発生する低遅延によるデータ処理が必要となる処理，例えばSNSのメッセージやクリックストリームによるWebサイト解析，不正侵入検知などのストリームデータ処理のために開発されたものであり，並列分散処理によるスケーラビリティと可用性を実現するアーキテクチャとなっている.

6.　今後の動向と取り組み

6.1　ハードウェア技術の進化によるデータ処理技術の発展

2015年の第41回VLDB国際会議（41st International Conference on Very Large Data Bases）の基調講演では，データベースとハードウェアとの親和性がテーマとして取り上げられている．Oracle[22]では，今後SQL on Siliconとしてデータベースをマイクロプロセッサのレベルで統合し，チップに性能，信頼性，コスト，セキュリティの向上を図るための機能を組み込むとしている．従来，データベースのアルゴリズムはチップに組み込むには複雑であるとされたが，インメモリのカラム指向の技術は，この制約を打開するものとなった．Ailamaki[23]では，NUMAにおける不均一性の影響を排除するための考慮や，キャッシュリソースの有効な利用，異種データベースの統合的なアクセスの研究により，H/Wアーキテクチャに適合したアプローチが鍵になるとする.

H/Wにおける技術展開として，今後さらにメニーコアと大規模メモリが進むとともに，ストレージ・クラス・メモリ（SCM）と呼ばれる次世代の不揮発メモリ技術の発展が重要と予想される[24][25]．SCMは，メモリと2次記憶装置の両方の特徴を持つ不揮発メモリであり，DBMSにおいては，リカバリ機構におけるログのディスクへの書き込みの効率化や，バッファ管理の単純化によるデータの読み込み，書き込みオーバーヘッドの削減，またトランザクション処理の同時実行性能の向上など，大幅な性能向上が期待できる[*19]．また，不揮発メモリの利用による高性能化だけでなく，新たなH/Wアーキテクチャに適するDBMSの最適化が課題と考えられる．H/Wとデータベースの最適化のための連携は，H/Wにデータ処理を取り込むアプローチと，DBMSにH/Wアーキテクチャと親和性の高い処理を取り入れるアプローチとがある．今後，この両方のアプローチにより，DBMS技術とH/Wとがより密接に連携し融合していくことが重要と考えられる.

6.2　データ統合技術の発展

特性の異なるデータ処理技術を組み合わせることで，ビッグデータの多様性に対応すること

を目指したラムダ・アーキテクチャ[*20]が提案されている．ラムダ・アーキテクチャは，バッチ層，サービス層，スピード層の三つから構成される．バッチ層では，マスタデータソースに対してバッチ・ビューとして参照結果セット取得のために関数を実行する．サービス層では，バッチの結果セットを集計する．スピード層では，低レイテンシのデータを補完しサービス層に提供する．ラムダ・アーキテクチャに基づくデータ処理基盤の例を図 8 に示す．蓄積基盤としての Hadoop や NoSQL に対して，リアルタイム・データ処理としてのインメモリ型 KVS，ストリームデータ処理としての Storm や Spark Streaming を組み合わせ，ビッグデータの多様な側面に対する汎用的なデータ処理基盤を実現することが想定される．

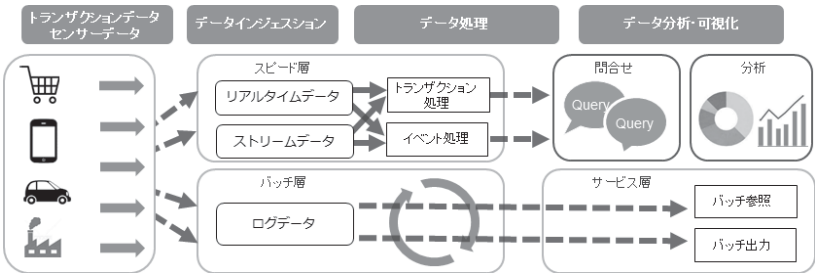

図 8　ラムダ・アーキテクチャの例

ビッグデータの特性であるデータ量や粒度，発生の頻度，速度，また形式の多様性など，さまざまな様相に対応してデータ技術を適切に連携させることが求められている．日本ユニシス総合技術研究所では，この課題に対して，PostgreSQL から透過的に Hadoop 上のデータを処理するための Hadoop クラスタ内データ処理 PG Inter-Analytics[26]を開発した．これにより，PostgreSQL から Hadoop 上のデータへの SQL による一元的なアクセスを可能とし，透過的に Hadoop 上のデータを処理できる．従来の RDBMS のアプリケーションは，Hadoop を意識せず大規模データ処理を利用することが可能となる．また Hadoop 上に蓄積された大規模データを直接処理することにより，データ転送の負荷が排除され効率的な処理を可能とする．今後，Spark に対応することで，データ処理の高速化やストリームデータへの対応を行う予定である．

7. お わ り に

本稿では，データの大規模化とデータ利用の多様化におけるデータベース技術の動向と展望について，Hadoop で展開された大規模分散処理と，RDBMS の発展による二つの技術的な視点から解説した．データベース技術は，汎用的な RDBMS の時代から，多様化するデータ形態に対応し，Hadoop をはじめ，NoSQL や NewSQL など，新たなデータ処理技術やデータベースが登場している．これらは，それぞれ異なる役割を担うものとして発展してきたが，異なるデータソースの統合が求められるビッグデータ活用の実現において，互いに連携して動作する汎用的なビッグデータ・マネージメント・システムとしてのアーキテクチャが提案されるようになっており，例えばラムダ・アーキテクチャはその例である．

インターネットの発達により，ビジネスやコミュニケーションの形態は多様化し，生活や社会において大きな変化をもたらした．人や社会，企業の複雑化した活動から発生する大規模なデータに加え，今後の IoT の発達により，生活環境における多様なモノから大量のデータが

発信される．これらコンテキストの異なる多様なデータソースを組み合わせることにより，新たな価値の創出が期待される一方で，データ処理の複雑化による新たな問題が生じる可能性がある．分散化によるデータ処理と新しい H/W 技術が，より密接に連携し，融合することで，今後のデータ処理の課題への解決となることが期待される．

また，最近注目を集めている機械学習や人工知能において，IoT がもたらすビッグデータの利用が重要になるとされる．これは大規模データ処理の新たな応用であり，そのための高度なデータベース技術の発展が予想される．データベース技術は，インテリジェンスと自律性を指向する未来の IT 技術の基幹を支えるものとして，今後も技術革新が進んでいくと考えられる．

＊ 1　大手 Web 系企業で採用されたスケールアウト型分散システム構築の手法であり，ビジネス要求に応じた柔軟で迅速な拡張を可能とする技術である．今後，一般企業においても普及が予想される．

＊ 2　Hadoop，Spark とも Apache Software Foundation のトップレベルプロジェクトである．

＊ 3　M. Stonebraker は，現代のデータベース・システムにおける基礎的な貢献により，2014 年の ACM チューリング賞を受賞した．
http://amturing.acm.org/award_winners/stonebraker_1172121.cfm

＊ 4　1970 年代に IBM で開発された最初の RDBMS であり，SQL を実装しトランザクション処理を実現した．

＊ 5　AMPLab は，カリフォルニア大学バークレー校（UC Berkeley）において，ビッグデータに関連する諸研究を行うために立ち上げられた研究機関．（https://amplab.cs.berkeley.edu/）

＊ 6　Impala は，Cloudera により開発され 2012 年に OSS として公開されていたが，2015 年 11 月に Apache Software Foundation に寄贈された．

＊ 7　Presto は，Facebook により開発され，2013 年に OSS として公開された．

＊ 8　Drill は，MapR により開発され，2012 年に Apache Software Foundation に寄贈された．

＊ 9　Tez は，Hadoop Yarn で動作する DAG（有向非循環グラフ）型のデータ処理フローの実行フレームワーク．MapReduce に比べ，複雑な処理フローを高速に処理する．

＊10　NoSQL という言葉が一般に認知されたのは，2009 年の NoSQL カンファレンス以降と言われる．

＊11　例えば，グラフ DB の Neo4j は単一サーバ型である．

＊12　DB-Engine Ranking（http://db-engines.com/en/ranking）（2016 年 1 月参照）や，Gartner Magic Quadrant for Operational Database Management Systems 2015 のレポートを参考に推測．Gartner Magic Quadrant は以下から参照可能．
MongoDB：https://www.mongodb.com/collateral/gartner-mq-2015
DataStax：http://www.datastax.com/gartner-magic-quadrant-odbms

＊13　ネットワーク耐性とは，ネットワーク障害により，クラスタが複数のグループに分断され，メッセージの遅延や喪失なども含む状況である．

＊14　Brewer はその後 2012 年に CAP 定理のより詳細な分析を示し，CAP 定理の（三つのうち二つが成立するという）単純化した解釈が誤解を生じるとしている[17]．

＊15　ACID：Availability（可用性），Consistency（一貫性），Isolation（独立性），Durability（永続性）

＊16　BASE：Basically Available（基本的な可用性），Soft-state（厳密でない状態），Eventual Consistency（結果整合性）．

＊17　MIT の Michael Stonebraker を中心に，MIT，Brown，Yale による大規模トランザクション処理システム H-Store の共同研究プロジェクトにより行われた．

＊18　パーティションにまたがった更新の場合は，内部的に分散 2 相コミットとなるため，1 パーティションでの実行に比べその分オーバーヘッドとなる．

＊19　HPE（ヒューレット・パッカード・エンタープライズ）は，Universal memory と呼ばれる不揮発メモリを搭載するサーバ The Machine に関するロードマップを発表しており，今後の実用化が期待される．

＊20　Storm の開発者である Nathan Marz によって 2012 年に提唱されたアーキテクチャ．
http://www.databasetube.com/database/big-data-lambda-architecture/

参考文献

[1] Michael Stonebraker, Samuel Madden, Daniel J. Abadi et al., "The end of an architectural era: (it's time for a complete rewrite)", in Proceeding of VLDB, 2007.
[2] E. F. Codd, "A relational model of data for large shared data banks", Communications of the ACM, 1970.
[3] S. Ghemawat, "The Google file system", In SOSP '03 Proceedings of the 19th ACM symposium on operating systems principles, 2003.
[4] J. Dean, et al., "MapReduce: simplified data processing on large clusters", In Proceedings of OSDI '04: 6th Symposium on Operating Systems Design and Implementation, 2004.
[5] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin et al., "Spark: cluster computing with working sets", Proceedings of USENIX, 2010.
[6] Apache HIVE, http://hive.apache.org/, In Apache Software Foundation Project.
[7] A. Thusoo, et al., Hive: a warehousing solution over a map-reduce framework, In Proceedings of the VLDB Vol. 2-2, 2009.
[8] Apache Pig, http://pig.apache.org/, In Apache Software Foundation Project.
[9] Apache Impala, http://impala.io/, In Apache Software Foundation Project.
[10] Presto, https://prestodb.io/, Facebook.
[11] Apache Drill, https://drill.apache.org/, In Apache Software Foundation Project.
[12] A. Lakshman, P. Malik, "Cassandra: a decentralized structured storage system", ACM Special Interest Group on Operating Systems (SIGOPS), 2010.
[13] DataStax Cassandra, http://www.datastax.com/
[14] G. DeCandia, D. Hastorun, M. Jampani et al., "Dynamo: Amazon's Highly Available Key-value Store", ACM Symposium on Operating Systems Principles (SOSP), 2007.
[15] F. Chang, J. Dean, S. Ghemawat, W. Hsieh et al., "Bigtable: a distributed storage system for structured data", in Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2006.
[16] E. Brewer, "Towards Robust Distributed Systems", 19th ACM symposium on Principles of distributed computing (PODC), 2000.
[17] E. Brewer, "CAP Twelve Years Later: How the "Rules" Have Changed", Infoq, 2012, http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed
[18] S. Harizopoulos, D. J. Abadi, S. Madden, M. Stonebraker, "OLTP through the looking glass, and what we found there," in Proceedings of the ACM SIGMOD, 2008.
[19] Mike Stonebraker, Daniel J. Abadi, Adam Batkin et al., "C-store: a column-oriented DBMS", in Proceedings of VLDB, 2005.
[20] Michael Stonebraker, Uğur Çetintemel, Stan Zdonik, "The 8 requirements of real-time stream processing", ACM SIGMOD, 2005.
[21] Apache Kafka, http://kafka.apache.org/ , In Apache Software Foundation Project.
[22] Juan Loaiza, Oracle, "Engineering Database Hardware and Software Together", Keynote at VLDB 2015, http://www.vldb.org/2015/videos.html
[23] Anastasia Ailamaki, EPFL, "Databases and Hardware: The Beginning and Sequel of a Beautiful Friendship", Keynote at VLDB 2015, http://www.vldb.org/2015/videos.html
[24] Ismail Oukid, Wolfgang Lehner, Thomas Kissinger et al., "Instant Recovery for Main Memory Databases", CIDR 2015.
[25] C. Mohan, Suparna Bhattacharya, "Implications of Storage Class Memories (SCM) on Software Architectures", HPCA Workshop, 2010.
[26] 中山　陽太郎, "PostgreSQL による Hadoop を利用した大規模データ分析機能 PG Inter-Analytics の紹介", PostgreSQL カンファレンス, 2013, https://www.postgresql.jp/events/jpug-pgcon2013/#A3

上記巻末注および参考文献欄の URL は，2016 年 1 月 20 日時点での存在を確認.

執筆者紹介　中 山　陽 太 郎（Yotaro Nakayama）

1988年日本ユニシス(株)入社. Unisys汎用機IXシリーズのデータベース管理システム主管業務を経て，オブジェクト指向DB，ETLツール，オープンソースDBの開発保守を担当．2009年より総合技術研究所に所属．データ統合，疫学DB構築，及び大規模データ処理技術に関する調査研究に従事.