SQL on Hadoopの可能性

Possibility of SQL on Hadoop

清酒文人，小原未生

要　約　2010年代に入ってから，ビッグデータが多くの企業から注目されている．その理由は，ビッグデータを活用することで，同業他社との差別化を図り，売上拡大や業務効率化により企業価値を向上させることができるからである．ビッグデータを蓄積/処理する基盤としてDWH向けRDBMSやHadoopがあり，成り立ちやアーキテクチャは異なるが，データを格納して，SQLをインタフェースとしてデータにアクセスすることができる点など，利用者から見ると違いが分かり難くなってきている．

上記課題に対して，DWH向けRDBMSとHadoopを用いて，データロード，データ検索について性能面での比較検証を行い，SQL on Hadoop（Hadoop内のデータをSQLで処理する機能）の適用範囲について考察を行った．その結果，データレイクにおける対話型の検索ツールとしては，SQL on Hadoopの適用は有用であるという結論に至った．本稿では検証結果と考察を報告する．

Abstract　Since the beginning of the 2010s, big data has been attracting attention from a number of companies. Because taking advantage of big data enables your company to improve the corporate value by differentiating corporate-itself from competitors, and by expanding sales and improving operational efficiency. There are RDBMS specialized in DWH and Hadoop as infrastructures for accumulating and processing big data, though their origins and architectures are different, it is difficult for user to understand the differences because both of them can access the data by the use of SQL as an interface after the storage of the data.

In relation to the problems above, we carried out a comparative validation on performance of data loading and searching by using the RDBMS specialized in DWH and Hadoop, and discuss the scope of the SQL on Hadoop (function to process the data in the Hadoop with SQL). Consequently we concluded that the application of SQL on Hadoop was very useful as the interactive search tool in the Data Lake. We report the results of verification and discussion in this paper.

1. はじめに

2011年に米国の調査会社McKinsey & Companyの研究部門であるMcKinsey Global Instituteが発表したレポート「Big data: The next frontier for innovation, competition, and productivity」[1]によりビッグデータが注目を集めており，それ以降，本格的にビッグデータの活用に取り組む企業が増えてきている．その理由は，ビッグデータを活用することで，同業他社との差別化を図り，売上拡大や業務効率化により企業価値を向上させることができるからである．スマートフォンやSNS（Social Networking Service）などの普及により，膨大かつ多様なデータが短時間に発生しており，2020年には約40ZB（ゼタバイト）のデータが発生すると言われている．これらのデータを蓄積する基盤としては，データウェアハウス（以下，DWH）専用

のリレーショナルデータベース管理システム（以下，RDBMS）と Hadoop がある．両方とも成り立ちやアーキテクチャは異なるが，データを格納して，SQL をインタフェースとしてデータにアクセスすることができる点など，利用者から見ると違いが分かり難くなってきている．

本稿では，SQL on Hadoop と DWH 向け RDBMS を性能比較し，SQL on Hadoop の適用範囲について考察する．2 章ではビッグデータの誕生背景と課題について，3 章では DWH と Hadoop の技術動向について整理する．4 章では SQL on Hadoop の適用範囲に関する仮説を挙げ，それを立証するための検証環境について説明する．5 章では各機能の性能評価に関する検証結果をまとめ，6 章では SQL on Hadoop の適用範囲に関して考察する．7 章では日本ユニシスの今後の取り組みについて述べる．

2. ビッグデータが注目されている背景と課題

2.1 ビッグデータが注目されている背景

エンタープライズシステムにおけるデータ活用は，基幹系システムのデータを分析対象としていた．しかし，2011 年の McKinsey Global Institute のレポート発表以降，スマートフォンや SNS など基幹系システム以外のデータも分析対象とすることで，他社との差別化を図り，企業価値を向上させることができると考えられるようになった．基幹系システムのデータも含めた企業内外に存在するあらゆるデータの総称がビッグデータであり，ビッグデータという言葉が注目を集めることとなった．

他にもビッグデータが注目されるようになった理由は二つある．一つは，総務省の情報通信白書で先進企業におけるビッグデータの利用実態が公開され，ビッグデータ活用の有効性が各企業に認識されたことである．もう一つは，CPU コア数増加による性能向上やネットワーク帯域拡大など，ビッグデータ活用に必要な技術が革新的に進歩したことである．

2.2 ビッグデータの課題

前節でビッグデータが注目されている背景について述べたが，2015 年時点でもビッグデータの活用が進んでいるとは言えない．その理由は，「投資対効果」，「プライバシー，機密情報の取り扱い」，「データの精度」，「技術者不足」の四つの課題があるためである．ビッグデータ活用のためには，多額の IT 投資が必要になるが，投資対効果の算出が難しく，経営層の承認を得るのが難しい．また，ビッグデータ活用を始めたとしても，プライバシー，機密情報への抵触を避けるために，データの活用が進まないことや，収集したデータの精度が悪く，意図していた活用ができないこともある．さらに，NoSQL やインメモリデータグリッドなど新しい技術が次々と出てきても，対応できる技術者は，育成が進まず慢性的に不足している．

技術者の不足については，Hadoop などでの SQL インタフェースへの対応のように，多くの人が利用できるような機能拡張が行われている．しかし，利用者視点では DWH 向け RDBMS と Hadoop のどちらを採用すべきか分かり難く，適用しても期待した性能要件を満たせない可能性がある．そうならないためには，各々のアーキテクチャや特性を理解した上で，適用を検討する必要がある．

3. DWH 向け RDBMS と Hadoop の技術動向と問題提起

3.1 DWH 向け RDBMS の技術動向

DWH とは 1990 年代初頭にビル・インモン（William H. Inmon）によって提唱された，企業ビジネスの意思決定を支援するデータの格納庫である．DWH は基幹システムのデータを加工して取り込み，目的別にデータマートを作成していた．また，DWH の実装には Oracle，SQL Server，DB2 などの汎用 RDBMS が採用されていた．しかし，汎用 RDBMS では，大量データを扱うには大量のコンピュータリソースを搭載した高価なハードウェアが必要となるため，限られた企業のみが大量データを活用していた．2000 年代に入ってから専用のハードウェアにより大幅な性能向上を実現させた DWH 専用のアプライアンスが登場したが，SNS などで大量に発生するログデータを処理するために，さらなる性能向上のニーズが高まった．これに対応するため，マスターノードの役割を固定しない超並列処理と，処理に必要な列だけを扱う列指向のアーキテクチャを実装した新型 DWH 向け RDBMS が登場した．超並列処理はデータ量に応じたスケールアウトを可能とし，列指向技術はデータ圧縮技術と組み合わせることで，I/O 処理の削減とメモリの有効活用により性能向上を実現している．

これらのニーズに応えるために，日本ユニシスでも 2013 年から，超並列処理と列指向技術を採用した Hewlett Packard Enterprise 社の HP Vertica Analytics Platform（以下，Vertica）を新型 DWH 向け RDBMS 製品として販売開始した．

3.2 Hadoop の技術動向

Hadoop は，並列処理を行うためのオープンソースミドルウェアである．並列処理は，ひとつの計算処理を分割し，複数のサーバーで並列に処理することにより処理時間を短縮する技術である．2003 年と 2004 年に Google が発表した並列処理に関する二つの論文[2][3]を参考に，Doug Cutting を中心とする少数のメンバーにより開発が始まり，Apache コミュニティのプロジェクトとして，現在も多数の開発者により日々開発が進められている．

Hadoop は「HDFS（Hadoop Distributed File System）」と「MapReduce」の二つの主要技術から構成される．HDFS は，大容量のファイルを複数のサーバーに分割して格納する分散ファイルシステムである．MapReduce は，並列処理を行う Java のフレームワーク/実行エンジンである．これらの技術により，大容量ファイル処理の時間短縮を実現している．

並列処理の技術は，スーパーコンピュータで採用されている MPI（Message Passing Interface）など Hadoop 以外にも存在する．それらと Hadoop が大きく異なる点は耐障害性がフレームワークとして備わっている点である．Hadoop はコモディティサーバーで動かすことを想定して開発されており，HW 故障が発生することを前提としたアーキテクチャになっている．障害時にデータを欠損しないように，データの複製を別のサーバーに格納する．並列処理の途中でサーバーの障害が起きた場合には，行っていた処理を別のサーバーで自動的にやり直す仕組みが備わっている．そのため，利用者は障害発生時の考慮を意識することなく，ビジネスロジックの実装に専念できることから，ビッグデータの蓄積/処理基盤として注目を集めている．

Hadoop の利用が進むにつれて，より多くの人が利用できるように周辺での機能拡張が行われている．それら機能拡張のコンポーネントは Hadoop エコシステムと呼ばれる．例えば，Java で実装する必要があった MapReduce を，SQL に似た言語で扱えるようにした Hive がその代表である．さらに，標準 SQL への準拠度を上げ，対話型の処理を行うために，SQL on

Hadoop も登場しており，その一つに Drill がある．Drill は，MapReduce を用いずにメモリで処理を行うことにより，処理速度を向上させている．また，技術力を持った先進的な企業でなくとも安心して使えるように商用ディストリビューションが提供されている．代表的なベンダは Cloudera，Hortonworks，MapR Technologies である．日本ユニシスでは 2014 年から，MapR Technologies の Hadoop ディストリビューション（以下，MapR）を販売開始した．MapR では，I/O 効率を向上させ，NFS（Network File System）でのアクセスを可能にした HDFS 互換の独自ファイルシステム（以下，MapR-FS）を採用している．

4. SQL on Hadoop の適用範囲についての仮説と検証環境

4. 1 SQL on Hadoop の適用範囲についての仮説

SQL on Hadoop の適用範囲について三つの性能の観点で仮説を検討する．一つ目は検索するデータを取り込む時間，二つ目はデータを検索する時間，三つ目は同時アクセスでの検索時間である．データ取り込みについては，Hadoop は DWH 向け RDBMS に比べ処理時間は短い．これは DWH 向け RDBMS で，検索処理を速くするようなデータの並び替えやフォーマット変換などを取り込み時に行っているためである．検索処理については，データ取り込み時に独自フォーマットで格納している DWH 向け RDBMS の処理時間が一番短く，メモリで処理を行っている SQL on Hadoop は，MapReduce で処理を行う Hive に比べて処理時間が短い．同時アクセスについては，DWH 向け RDBMS はリソースを一元的に管理することによって効率的にデータ共有を行っているため，Hadoop に比べて同時実行性能は高い．

以上のことから，以下の仮説が考えられる．これらの仮説は 5 章で検証している．

1） Hadoop は，分析対象となる全てのデータの格納に向いている．
2） DWH 向け RDBMS は，利用頻度が高いデータに絞り込んで格納する必要がある．
3） SQL on Hadoop は，対話型による試行錯誤的なデータ検索に向いている．
4） Hive は，メモリサイズ以上の大量データを扱うバッチ処理に向いている．
5） DWH 向け RDBMS は，多くの利用者が同時にアクセスするデータ検索に向いている．

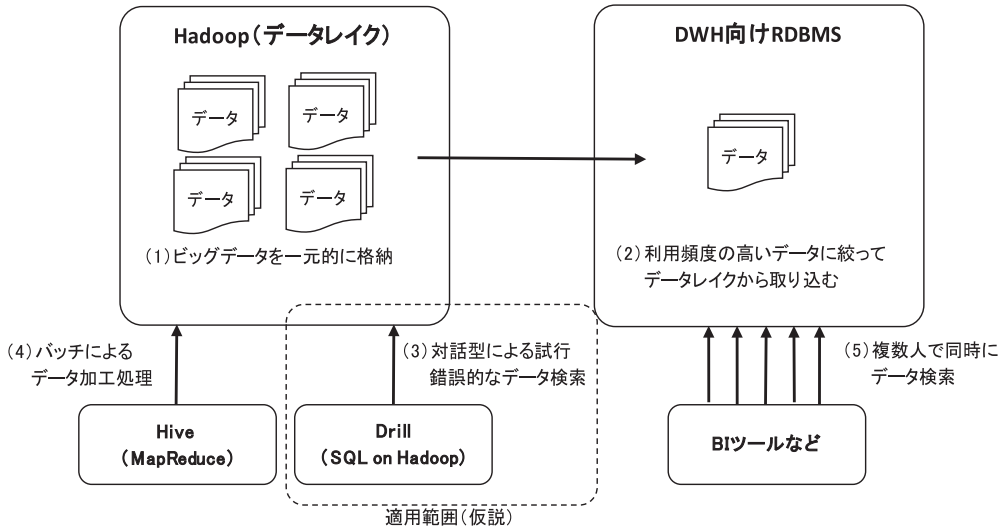

図 1 SQL on Hadoop の適用範囲についての仮説

したがって，SQL on Hadoop の適用範囲は図 1 のように，Hadoop（データレイク）内のデータに対する対話型の検索ツールとして有用であると考える．

4. 2　検証項目

4. 1 節で挙げた仮説を検証するため表 1 のように，データロードとデータ検索における単独実行と同時実行について検証する．データ検索の実行クエリは，データ分析で基本となるグループ集計とテーブル結合のクエリとする．

表 1　検証項目

検証項目	確認内容
データロード	データを検索するまでのリードタイムを確認するため，同一データに対するロード処理時間を確認する．
データ検索（単独実行）	データを検索する処理速度を確認するため，同一データに対するグループ集計とテーブル結合のクエリ処理時間を確認する．
データ検索（同時実行）	同時アクセスによる特性を確認するため，同時実行数を増やした場合の処理時間の増加傾向を確認する．
データロード＋データ検索	データ検索が完了するまでの時間を確認するため，データロードとデータ検索の合計処理時間を確認する．

4. 3　前提条件

検証における前提条件は，下記の通りである．

1）　DWH 向け RDBMS には，日本ユニシスで採用している Vertica を使用する．Vertica の前提条件は，下記の通りとする．
　ⅰ）スーパープロジェクション（列配置で最適化することにより高いデータ圧縮率を実現したデータセット）を作成する．
　ⅱ）テーブルには，パーティションを設定する．
　ⅲ）同時実行数は Vertica のデフォルト値（1 サーバーの core 数と同じ）を使用する．

2）　Hadoop には，日本ユニシスで採用している MapR を使用する．MapR の前提条件は，下記の通りとする．
　ⅰ）Hadoop エコシステムの中で，SQL でデータを扱える Hive と Drill を使用する．
　ⅱ）Hive テーブルには，パーティションを設定する．
　ⅲ）Hive テーブルは，以下二つの形式を作成する．
　　ア）デフォルト形式（テキストファイル形式）．
　　イ）列単位にデータ保存を行う Parquet 形式．
　ⅳ）上記の Hive テーブルの形式に適用する圧縮アルゴリズムは，下記の通りとする．
　　ア）デフォルト形式の場合は，MapR-FS 自動圧縮機能により LZ4 で圧縮される．
　　イ）Parquet 形式の場合は，圧縮解凍処理が速く，I/O 処理が高速となる snappy を適用する（以下，P+s 圧縮形式）．
　ⅴ）Drill は Hive テーブルからデータを検索する．

3）　データロードを行う前のデータ形式は csv 形式とする．

4）　データロードを行う前のデータは MapR-FS に配置する．

5）　データ検索の処理時間は，クエリを発行してから全結果がファイルに出力完了するまでとする．

4. 4　検証環境の構成

検証環境として，図 2，表 3 のように，VMware を用いた仮想サーバーを 3 台作成する．尚，仮想サーバーは物理サーバーと 1 対 1 となるように作成する．

MapR と Vertica のインストールは，リソースの使用状況など OS に関する条件を同じにするため，同じ仮想サーバー 3 台に対して行う．

ネットワークは，Vertica で推奨されている Private（ノード内通信用）と Public（外部との通信用）の 2 系統とする．

MapR と Vertica の設定値は，1TB 程度のデータ量を処理できるように修正している．

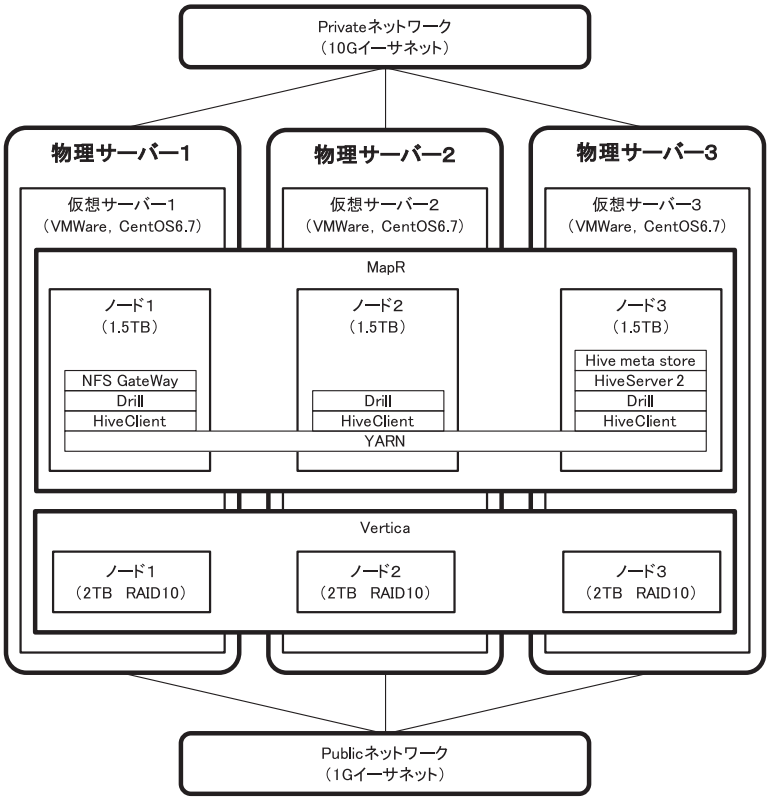

図 2　検証環境構成図

表 3　仮想サーバーの環境構成詳細

項目	設定内容
CPU	6core
Memory	40GB
Storage	MapR，Vertica OS 領域：各仮想サーバー 250GB MapR，Vertica データ領域：各仮想サーバー 3.5TB ※ 3 ノード合計値 10.5TB MapR1.5TB 分，Vertica2.0TB（RAID10） ※ OS 領域とデータ領域は別のデータストア
OS	CentOS6.7（カーネル ver：2.6.32-573.el6.x86_64） ※ VMWare（ESXi5.5.0）を用いて仮想化したサーバー上にインストール
SW	MapR Community Edition（m3）5.0.0（Hive：ver1.0，Drill：ver1.1） YARN の core 数は 4，メモリ量は 16GB とする． Drill の最大メモリ量は 24GB，ヒープメモリ量は 14GB，1 クエリに使用できる最大メモリ量（1 ノード単位）は 8GB とする．
	HP Vertica 7.1.2 Community Edition パラメータはデフォルト値とする．
Network	Public LAN Private 10Gbit Ethernet

4. 5　検証処理概要

4. 2 節で挙げた検証項目の処理概要は，図 3 の通りである．

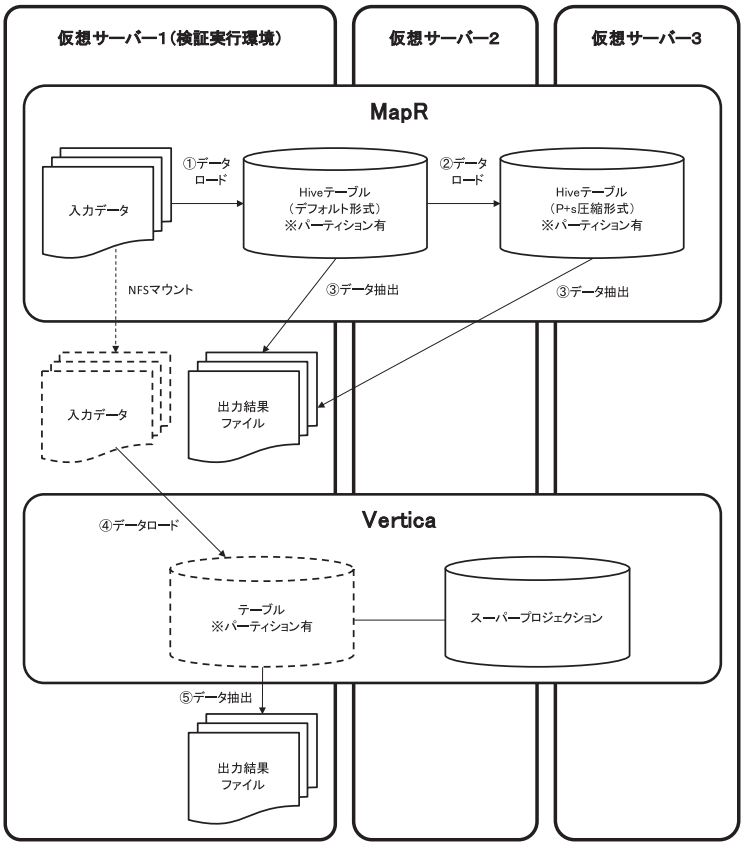

図 3　検証処理概要図

MapR（Hive）のデータロードでは，MapR-FS に格納されたデータをデフォルト形式の Hive テーブルにロードする（①）．P+s圧縮形式の Hive テーブルにデータロードを行う場合は，デフォルト形式の Hive テーブルに格納されたデータをロードする（②）．

MapR（Hive，Drill）のデータ検索では，各形式の Hive テーブルからデータ検索を行い，出力結果を仮想サーバー 1 のローカルディスクに出力する（③）．

Vertica のデータロードでは，NFS マウントした MapR-FS に格納されたデータを Vertica のテーブルにロードする（④）．

Vertica のデータ検索では，MapR と同様，出力結果を仮想サーバー 1 のローカルディスクに出力する（⑤）．

4. 6　検証データ

検証データのテーブル構造は，図 4 の通りである．結合するテーブルの件数，サイズの違いによる処理性能を確認するため，検証テーブルと結合するマスターテーブルを三つ用意している．また，同時実行の検証のみ，検証テーブルのサイズを 0.33TB とした．

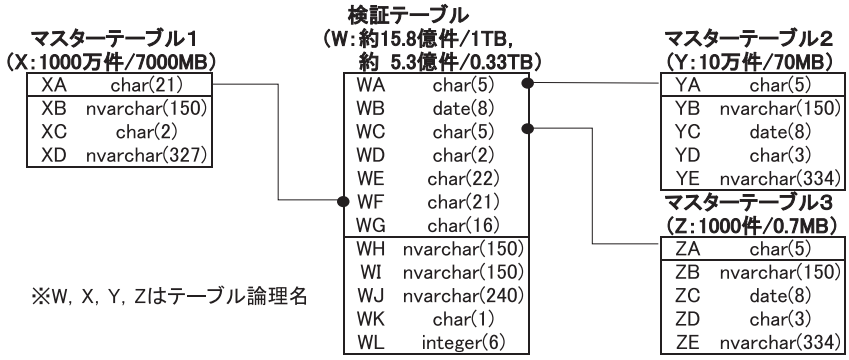

図 4　検証データのテーブル構造

5.　検証結果

5. 1　データロード

1TB のデータに対する Hive，Vertica のデータロードの処理時間を確認する．検証手順は，表 4 の通りである．

表 4　データロードの検証手順

	検証手順
Hive_デフォルト	Hive テーブルに対する LOAD 文を実行.
Hive_P+s 圧縮	Hive テーブルに対する LOAD 文を実行し，その後 P+s 圧縮形式の Hive テーブルに INSERT-SELECT 文を実行.
Vertica	Vertica の一括ロード用コマンドである COPY 文を実行.

検証結果は，図 5 の通りである．

図 5　データロードの検証結果

Hive_デフォルトが 9 秒で処理されているのは，MapR-FS のデータに対する LOAD 文の実行が，内部ではファイルの移動しか行われないためである．ただし，ロードするデータがローカルディスクなどの MapR-FS 以外に格納されている場合，処理時間は増加すると考える．

Hive_P+s 圧縮の処理時間が Hive_デフォルトと比べて大幅に増加（1 時間 12 分 16 秒）しているのは，圧縮やフォーマット変換処理などによるものである．

Vertica はロード時にデータ圧縮処理以外にも，検索処理を効率的に行うためのプロジェクションによるデータソートや，データ格納における独自フォーマットへの変換を行っている．さらに，データ型や精度など，実データとテーブル定義との整合性もチェックするため，Hive_P+s 圧縮に比べて処理時間が長いが，データを分割して各ノードで同時にロードすることで，処理時間を短縮できる[*1]．

5. 2　データ検索（グループ集計）

Hive，Drill，Vertica のデータ検索（グループ集計）の処理時間とリソース状況を確認する．また，データロードとデータ検索を合わせた処理時間を考察するために，Hive，Drill は二つの Hive テーブル形式による検証を行った．検証概要は，表 5 の通りである．

表 5　データ検索（グループ集計）の検証概要

	検証概要
対象テーブル	検証テーブル（データ量：1TB，件数：15.8 億件）
基本クエリ	SELECT WC，SUM(WL) FROM W GROUP BY WC;
検索件数	1,000 件

検証結果は，図 6，7 の通りである．まず図 6 で，P+s 圧縮形式の Hive テーブルに対する Drill，Hive の実行結果と，Vertica の実行結果を掲載する．また，リソース状況についても表 6 に掲載する．

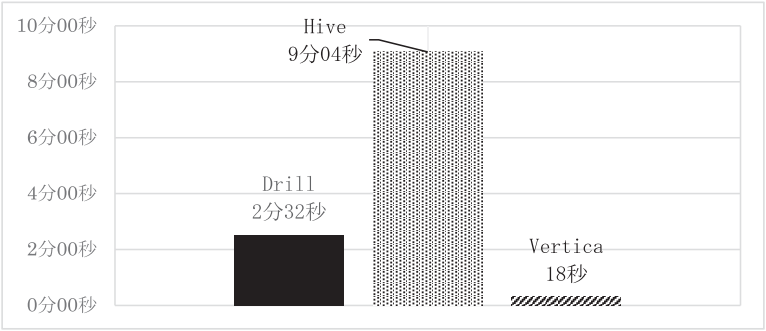

図 6　データ検索の検証結果

表 6　データ検索の I/O 時間と CPU 処理時間

	I/O 時間（3 サーバー合計）	I/O 時間の Drill との比	CPU 時間（3 サーバー合計）	CPU 時間の Drill との比
Drill	4 分 37 秒	—	25 分 18 秒	—
Hive	4 分 39 秒	1.0 倍	1 時間 13 分 57 秒	2.9 倍
Vertica	38 秒	0.14 倍	2 分 48 秒	0.11 倍

Hive，Drill，Vertica の順に処理時間が短くなっている．Hive と Drill は，I/O 時間（sar の tps×svctm×マシン数の処理時間帯での集計）は同程度であったが，CPU 時間（sar の CPU 使用率×マシン数×Core 数の処理時間帯での集計）が Hive は Drill の 2.9 倍であり，それが処理時間の差となっている．Hive はオーバーヘッドが大きい MapReduce 処理を行うため，MapReduce 処理を行わない Drill に比べると処理時間が長いことから，対話的ではなく，バッチ処理での適用が現実的であると考える[*2]．また，Hive のメモリ使用率が平均 40％程度に対し，Drill では平均 70％程度となっており，Drill がメモリを有効に活用して処理時間の短縮を図っていることも確認できる．

Vertica は I/O，CPU 時間ともに，Drill より極めて少なく（I/O 時間 :Drill の 0.14 倍，CPU 時間：Drill の 0.11 倍），ロード時間は長いが検索時間は短い DWH 向け RDBMS の特性が確認できる．

次に，図 7 でデフォルト形式の Hive テーブルに対する Hive，Drill の結果を掲載する．また，リソース状況についても表 7 に掲載する．

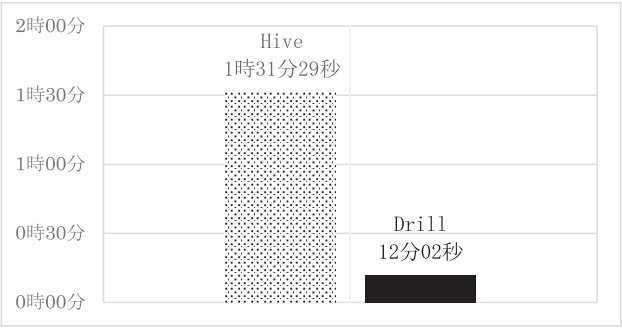

図 7　Hive テーブル（デフォルト形式）からのデータ検索の検証結果

表 7　Hive テーブルからのデータ検索での I/O 時間

	P+s 圧縮形式 (3 サーバー合計)	デフォルト形式 (3 サーバー合計)	デフォルト形式/ P+s 圧縮形式
Hive	4 分 39 秒	5 時間 19 分 54 秒	68.8 倍
Drill	4 分 37 秒	22 分 37 秒	4.9 倍

デフォルト形式の方が P+s 圧縮形式よりも大幅に処理時間が長くなっている（Hive：約 10 倍，Drill：約 5 倍）．ファイルサイズを比較するとデフォルト形式は P+s 圧縮形式の約 2 倍（LZ4：117GB，P+s：55GB）である．しかし，P+s 圧縮形式では処理対象の列だけを I/O 処理するのに対し，デフォルト形式では特定の列だけでなく全列を I/O 処理するため，ファイルサイズの増加以上に I/O 時間が長くなっている（Hive：約 69 倍，Drill：約 5 倍）．Hive が特に差が大きいのは MapReduce 処理内でデータの読み込みだけでなく，Map 処理で作成したデータを一時領域に書き込むためである．ここでも，Hive における MapReduce 処理の重さが確認できる．

5. 3　データ検索（結合）

データ検索（結合）については，Hive，Drill，Vertica で 2 テーブルを内部結合する処理時間を確認する．検証概要は，表 8 の通りである．

表 8　データ検索（結合）の検証概要

	検証概要
結合元テーブル	検証テーブル（データ量：1TB，件数：15.8 億件）
結合先テーブル	マスターテーブル 1 ～ 3 のいずれか 1 テーブル ・マスターテーブル 1（データ量：7000MB，件数：1000 万件） ・マスターテーブル 2（データ量：70MB，件数：10 万件） ・マスターテーブル 3（データ量：0.7MB，件数：1000 件）
基本クエリ	・マスターテーブル 1 と結合 SELECT WC, SUM(WL) FROM W INNER JOIN X ON WF = XA GROUP BY WC; ・マスターテーブル 2 と結合 SELECT WC, SUM(WL) FROM W INNER JOIN Y ON WA = YA GROUP BY WC; ・マスターテーブル 3 と結合 SELECT WC, SUM(WL) FROM W INNER JOIN Z ON WC = ZA GROUP BY WC;
検索件数	1,000 件

検証結果は，図 8 の通りである．尚，比較対象として，5. 2 節で実施したグループ集計の結果もグラフに掲載する（結合なしのパターンが該当）．

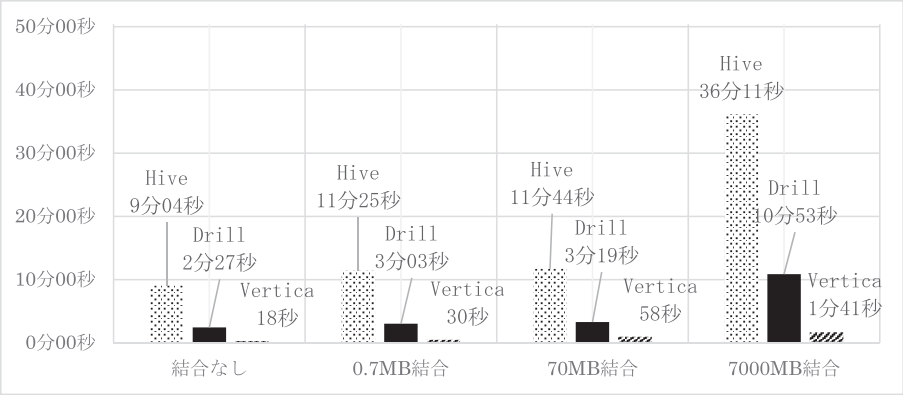

図 8　データ検索（結合）の検証結果

Hive，Drill，Vertica 全てにおいて，結合するマスターテーブルの大きさに伴い処理時間が長くなっている．Hive，Drill は，マスターテーブルが 70MB と 7000MB の間で処理時間に大きな差があるが，これは，Hive，Drill ともにデータ量に応じて処理方式を変更しているためである．

Hive は，テーブルのサイズが 128MB 以下の場合に，テーブルに対する MapReduce 処理を行わず，単一マシンで処理を行うローカルモードで処理を実行する．実行計画，実行ログからも，検証テーブルは 1TB のサイズなので，全ての結合ケースにおいて MapReduce 処理が行われているが，マスターテーブルに対しては，0.7MB，70MB はローカルモードで処理して，7000MB のみ MapReduce 処理が行われている．

Drill は，マスターテーブルのレコード数が 1000 万件未満であれば，マスターテーブルのデータを全ノードにブロードキャストして結合処理を行うが，1000 万件以上の場合は，結合キーのハッシュ値に基づき，ノード間に分散する仕組みとなっている．実行計画からも 7000MB テーブル（レコード数：1000 万件）の結合は分散処理が行われている．

また，Hive，Drill ともに 0.7MB 結合と 70MB 結合でほとんど処理時間に差がないのは，検証用テーブルのデータ量（1TB）と比べて，マスターテーブル 2（70MB），マスターテーブル 3（0.7MB）のデータ量が極めて小さいためであると考える．

Vertica は，結合するテーブルサイズによる閾値などは特になく，テーブルの大きさに応じて処理時間が増加している．

5. 4　データ検索（同時実行）

データ検索を複数同時に実行した場合の Hive，Drill，Vertica の処理時間の増加傾向とリソース状況を確認する．検証概要は，表 9 の通りである．

表 9　データ検索（同時実行）の検証概要

	検証概要
対象テーブル	検証テーブル（データ量：0.33TB，件数：5.3 億件）
基本クエリ	SELECT WC，SUM（WL）FROM W GROUP BY WC; ※ 5. 2 節で実施したグループ集計と同じクエリ
検索件数	1,000 件

まず，Hive，Drill は，YARN の Core 数を 4 と設定したため，4 クエリまでの検証を行った．検証結果は，図 9 の通りである．

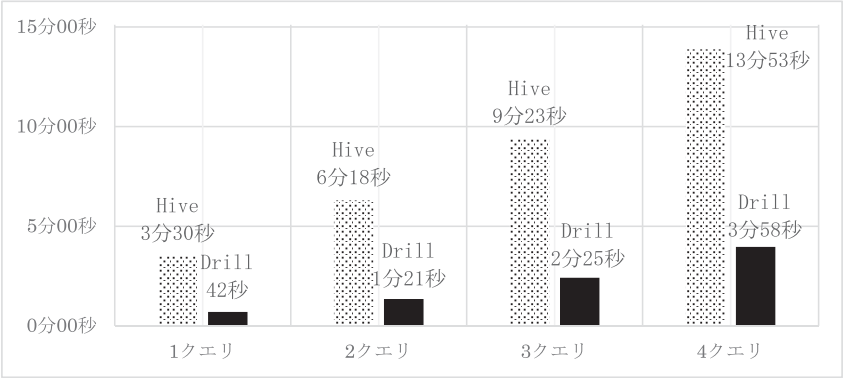

図 9　Hive，Drill によるデータ検索（同時実行）の検証結果

Hive はクエリ数に比例して処理時間が長くなっており，各クエリの開始，終了時刻は同じであるため，すべての処理が同時に実行されている．MapReduce ではデフォルトのジョブスケジューラである Fair Scheduler によって，同時に実行される処理に対して均等にリソースが割り当てられることから，並列度が増えれば 1 処理に割り当てるリソースは相対的に少なくなるため，処理時間も増加する．クエリ数を 2 倍にすると各クエリの処理時間も 2 倍になっていることから，同時実行性が高いとは言えない．

Drill もクエリ数に比例して処理時間が長くなっており，各クエリの開始，終了時刻は同じであるため，すべての処理が同時に実行されているが，3 クエリと 4 クエリでの同時実行では，実行中に CPU 使用率が 100％に達して，CPU 待ちが発生したことにより，1 クエリをシリアルに実行した場合に比べて処理時間が長くなった．core 数を増加することで，処理時間の増加は抑えられると考えるが，Hive 同様，同時実行性が高いとは言えない．

次に，Vertica は，1 サーバーの core 数が 6 であるため，6 クエリまで検証を行った．検証結果は，図 10 の通りである．

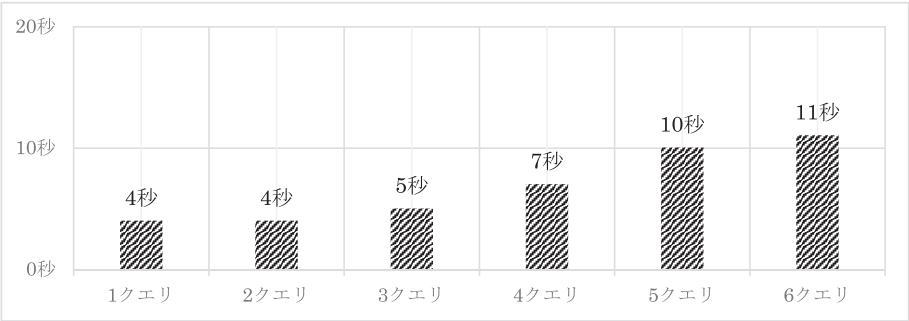

図 10　Vertica によるデータ検索（同時実行）の検証結果

1クエリを基準に見ると，クエリ数の増加に比例して処理時間は増加していないため，Hive，Drillに比べて同時実行性の高さが確認できる．実行したクエリの開始，終了時刻は同じであるため，すべての処理が同時に実行されている[*3]．

3クエリ以上で処理時間が長くなっているが，これはCPU使用率が100％に達しているためで，今回のケースでは一つのクエリで2core程度のCPUリソースを使って処理を行っているため，1サーバーに割り当てるcore数を倍の12coreに増やすことで6クエリでも1クエリと同じ処理時間で終わらせることが期待できる．

5.5　データロード＋データ検索

データロード＋データ検索として，5.1節データロードの時間と5.2節データ検索（グループ集計）の検索時間の合計処理時間を確認する．

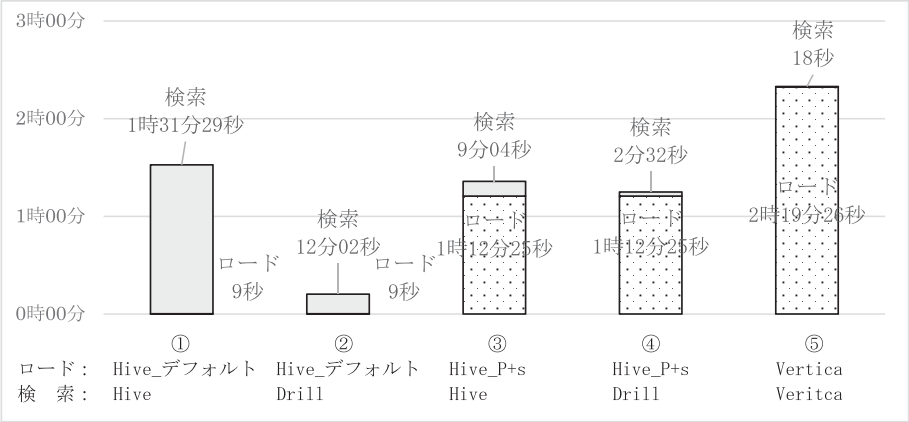

図11　データロードとデータ検索の合計処理時間（データ量：1TB）

Hive（①，③）に比べてDrill（②，④）の合計処理時間の方が短いため，検索はDrillを使うべきである．また，Verticaの合計処理時間（⑤）とHive_デフォルト，Drillの組み合わせ（②）との合計処理時間の差は2時間7分33秒となっている．処理時間の差のほとんどは，ロード時間であるため，検索頻度によって最適な方式を選択する．繰り返し検索しないデータはHiveのデフォルト形式にロード，繰り返し検索するデータはP+s圧縮形式にロード，頻繁に検索するデータはVerticaにロードすべきである[*4]．

6.　SQL on Hadoopの適用範囲についての考察

今回の検証では，データ量・ハードウェア・割り当てリソースが同じ条件で性能検証を行った．ロードが早く，メモリを有効活用することによりHiveに比べて検索時間が短く，対話型による試行錯誤的なデータ検索にも適用できるという仮説を証明できた．各技術の適用は，図12の通りである．

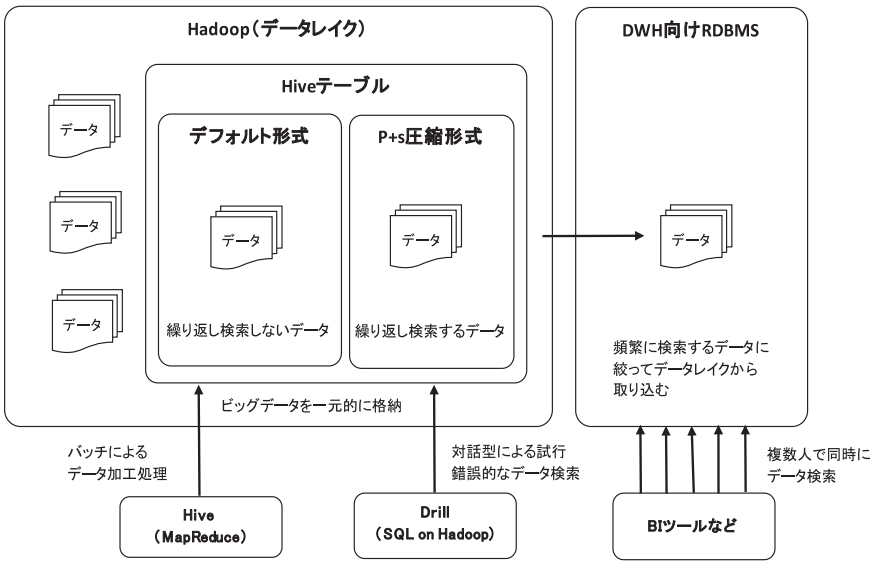

図 12　各技術の適用イメージ

しかし，ソリューションの選定には，性能だけではなく，ユーザの利用特性やコストなども勘案することが一般的である．DWH 向け RDBMS は検索の性能は明らかに高いが，分析対象データを全て取り込むとライセンス費用が Hadoop と比較して高価になることが多い．また，ロード時間もかかることから，分析目的が明確なデータしか取り込めない．

一方，Hadoop は DWH 向け RDBMS に比べてライセンス費用が安価でロードが速いため，分析目的が明確ではないデータも取り込み，大容量データを扱えるファイルシステム（データレイク）として活用することができる．

検索性能，同時実行性能に関しても，コストを評価軸に加えた場合には，Hadoop は，DWH 向け RDBMS よりもノード数を増やすことができるため，今回の検証結果よりも処理時間の差を縮めることができる．

また，データを利用するユーザ数が限定され，データベースの維持管理を行う専任要員を配置するのが難しい場合には，DWH 向け RDBMS よりも，SQL on Hadoop で構築，運用する方が適している．

したがって，コスト・ユーザー利用特性・専任要員の有無の観点などを考慮した場合には，必ずしも DWH 向け RDBMS が最善とは言えず，SQL on Hadoop の選択が最適なケースもある．

7. お わ り に

本稿では，SQL on Hadoop の適用範囲について考察した．性能面での検証を行った結果，データレイクにおける対話型の検索ツールとして SQL on Hadoop は有用であることが確認できた．今後，SQL on Hadoop のさらなる進化や新しい Hadoop エコシステムが登場し，DWH 向け RDBMS の性能により近づくことを期待する．

日本ユニシスで提供している「データ統合・分析共通 PaaS」[12]は，今回検証を行った MapR と Vertica で構成している．今後，Apache Spark などの検証を行い，機能拡充による商品力向上に取り組んでいきたい．

日本ユニシスでは，基幹系システムだけでなく情報系システムの構築実績も多数あり，多岐にわたる顧客要望にビッグデータ技術を駆使して対応している．今後も顧客の期待に応え続けるために，最適なビッグデータソリューションを提供していく所存である．

本稿執筆にあたってご協力，ご指導いただいた皆様に深く感謝し，厚く御礼申し上げます．

＊1　ここで4.1節の仮説1)を立証している．
＊2　ここで4.1節の仮説4)を立証している．
＊3　ここで4.1節の仮説5)を立証している．
＊4　ここで4.1節の仮説2)と3)を立証している．

参考文献

［1］ James Manyika，Michael Chui，Brad Brown，Jacques Bughin，Richard Dobbs，Charles Roxburgh，Angela Hung Byers「Big data: The next frontier for innovation, competition, and productivity」，McKinsey Global Institute，2011年5月
［2］ Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung,「The Google File System」，2003年
［3］ Jeffrey Dean and Sanjay Ghemawat,「MapReduce: Simplified Data Processing on Large Clusters」，2004年
［4］ Michael Manoochehri，小林啓倫,「ビッグデータテクノロジー完全ガイド，2014年11月
［5］ 山崎慎一,「企業情報システムとデータの活用範囲の拡大」，ユニシス技報，日本ユニシス，Vol.31 No.4 通巻111号，2012年3月
［6］ 羽生貴史,「データベース技術の動向」，ユニシス技報，日本ユニシス，Vol.29 No.2，通巻101号，2009年3月
［7］ 総務省,「平成24年版情報通信白書」，2012年6月，P153～160，
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h24/html/nc121400.html
［8］ 総務省,「平成25年版情報通信白書」，2013年7月，P143～153，
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h25/html/nc113000.html
［9］ 鈴木良介,「ビッグデータビジネスの時代」，株式会社翔泳社，2012年3月
［10］ Edward Capriolo，Dean Wampler．Jason Rutherglen，玉川竜司訳,「プログラミングHive」，オライリー・ジャパン（発行元），2013年6月
［11］ W. H. Inmon，藤本康秀，小畑喜一,「初めてのデータウェアハウス構築」，1995年12月
［12］ 日本ユニシス,「データ統合・分析共通PaaS」
http://www.unisys.co.jp/solution/biz/bigdata/solution/paas.html
（上記参考文献のURLは2016年2月25日時点で存在することを確認）

執筆者紹介

清　酒　文　人（Fumito Seishu）
2010年日本ユニシス(株)入社．2010年9月より，公共システム本部に所属し，電力，ガスのエネルギー事業に関わるシステム開発等に携わる．現在，アドバンスド技術統括部のデータ基盤技術部に所属し，ビッグデータに関する技術評価を担当している．

小　原　未　生（Mio Ohara）
2009 年日本ユニシス（株）入社．2009 年 10 月より，公共システム本部に所属し，エアライン事業に関わるシステム開発・保守に携わる．現在，アドバンスド技術統括部のデータ基盤技術部に所属し，ビッグデータに関する技術評価を担当している．