ビッグデータビジネスを加速するスケールアウト型データベース GridStore/NewSQL

GridStore/NewSQL Scale-Out Database Accelerating Big Data Utilization

服部　雅一　■HATTORI Masakazu
浜口　泰平　■HAMAGUCHI Taihei
近藤　雄二　■KONDO Yuji

キーバリューをデータモデルとするスケールアウト型データベース（DB）GridStore/NoSQLに対してSQL（Structured Query Language）機能を強化した，GridStore/NewSQLを開発した。RDB（Relational Database）と同等のインタフェースを提供し，データ量やアクセス数に応じてノード数を増やすことで，既存のRDBでは処理できない大容量のデータを管理できる。また，既存のBI（Business Intelligence）やETL（Extract, Transform, Load）などのツールと容易に連携可能になり，ビッグデータ分析や企業内データ統合など適用範囲を大きく広げることができた。

Toshiba has developed the GridStore/NewSQL database, which provides enhanced Structured Query Language (SQL) functions compared with the GridStore/NoSQL scale-out, key-value type database. The NewSQL database not only offers interfaces equivalent to those of relational databases (RDBs) but also manages large volumes of data that cannot be handled by conventional RDBs by increasing the number of nodes according to the volume of data and/or number of accesses. Moreover, this database can be easily used in conjunction with existing tools such as business intelligence (BI) tools and extract, transform and load (ETL) tools, thereby expanding the range of applications including big-data analytics and enterprise data integration.

1　まえがき

近年，情報のデジタル化が進み，車載機器や，家電機器，スマートフォンなど様々な製品がインターネットに接続されるようになり，多種多様な情報が扱えるようになった。また，情報処理技術の高度化とIT（情報技術）インフラの充実により，これらの大量のデータを収集し分析することも可能になった。

データ量の予測が不可能で大量に発生するために捨てていたデータを含め，日々増加する情報を収集し分析するには情報を蓄積するデータストアの存在が重要である。東芝は，時々刻々発生するデータを高速に処理する分散キーバリューストア（KVS：Key-Value Store）型のGridStore/NoSQLを開発し，社会インフラシステムでの適用を開始している[1]。

今回，GridStore/NoSQLに対して更にSQL（Structured Query Language）機能を強化したGridStore/NewSQLを開発し，高速性と，SQL利用によるアプリケーション開発の容易性の両面を達成した。ここでは，GridStore/NoSQL及びGridStore/NewSQLの概要と，応答性とスケールアウト性の検証結果について述べる。

2　GridStore/NoSQL

GridStore/NewSQLのベースであるGridStore/NoSQLについて，その概要を述べる。

2.1　分散KVSの課題

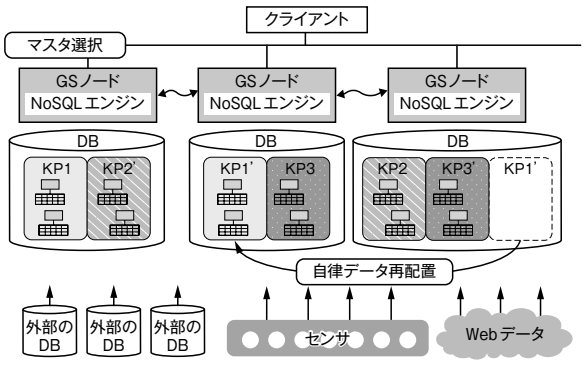
近年，一貫性の制限を緩和してデータの分散配置により処理性能を向上させる分散KVSと呼ばれるDBが注目されている。このDBは，その機能的なシンプルさのためにサーバ数増加による性能改善，すなわちスケールアウトが容易であり，通常はコモディティハードウェアを使いながら安価に構築される。その結果，従来のRDB（Relational Database）では処理できない規模の大容量データを管理できる。また，スケールアウト可能であれば，将来のデータ量やアクセス数などの需要増を見越した過剰なシステムリソースが不要になり，当面必要なリソースを準備するだけで済むため，むだな投資を最小化できる。

しかし，分散KVSは必ずしも良いことばかりではない。多くはデータ一貫性のレベルが低く，トランザクション機能も備えていない。また，RDBの標準言語であるSQLがなく，独自インタフェースの利用を前提とするため，アプリケーション開発者にとって開発効率が悪いことが弱みとなっている。

2.2　GridStore/NoSQLの特長

これらの課題を踏まえて開発したGridStore/NoSQLのシステム構成を**図1**に示す。

データを分割して各ノードに配置するDBクラスタは，データ配置に関するメタ情報をマスタノードが管理するマスタスレーブ型と，メタ情報を各ノードで管理しノード間で問い合わせるピアツーピア型に分類される。前者は，データの一貫性を維持するのが容易な反面，マスタノードが単一障害点になることに加えて，ノードを追加してデータを再配置するのが難しいという課題があった。一方，後者は，ノードを追加してデータを再配置するのが容易な反面，データ一貫性維持のためのノード間の通信オーバヘッドが大きく，結果としてデータの一貫性と



GSノード：GridStore ノード

*KPはキーバリューのパーティションデータを意味し，KP1，2，3はキーバリュー全体を分割したもの

図1．GridStore/NoSQLのシステム構成 — NoSQLエンジンを内蔵するGSノード群をネットワークで接続することでDBクラスタを構成する。

Configuration of GridStore/NoSQL database system

処理速度がトレードオフの関係になってしまう課題があった。

そこでGridStore/NoSQLは，これらの中間的な挙動を示すハイブリッド方式を開発することで，前述の欠点を克服した。DBクラスタ起動時やマスタノード障害時にマスタノードは不在になる。このとき，ノードが実行する投票に基づく選択プロトコルにより，DBクラスタが自律的にマスタノードを選ぶ。選ばれたマスタノードは，他のノード（スレーブノード）が持つキーバリューに関するメタ情報を収集し集中管理したうえで，スレーブノードへのメタ情報の伝達を制御する。

この仕組みにより，メタ情報をDBクラスタで矛盾なく共有することが可能になり，データの一貫性を維持しながら，処理速度を向上させることができた。また，自律的にノード間でキーバリューデータを高速に再配置するアルゴリズムを開発することで，ノード追加における高速性かつ耐障害性に優れたデータ再配置を実現した。GridStore/NoSQLの特長を以下に述べる。

(1) 高可用性　　ノード間でキーバリューデータの複製を互いのノードで持ち合う仕組みを備えており，万一ノード障害が発生しても，他ノードの複製を使うことにより，数秒で自動フェールオーバが可能である。また，ノード障害時にDBクラスタが分断されて複数の小さなDBクラスタが出現するスプリットブレインはDBクラスタとしてもっともやっかいな状態であるが，クオーラムポリシー(注1)により回避するメカニズムを搭載している。

(2) 高速性　　RDBではメモリを大容量化しても，バッファ管理などに大きなオーバヘッドが発生するため本質的なデータ処理にCPUリソースの10％前後しか割り当てられずCPUパワーを十分発揮できないことが知られている。GridStore/NoSQLは，大容量化されたメモリを前提に，バッファ処理の軽量化，リカバリ処理の軽量化，及びデータ処理時のロックフリー化を行うことで，これまでのオーバヘッドを最小化した。

(3) ノンストップスケールアウト　　DBクラスタが高いスケールアウト性を維持するためには特定ノードに負荷が集中しないよう，ノード間でバランスよくデータ配置する必要がある。前述した技術により，ノンストップなスケールアウトを実現した。

(4) 高度なデータモデルと操作モデル　　従来の分散KVSでは，Put，Get，及びRemoveという操作によりデータを操作する。GridStore/NoSQLは，これを大幅に拡張し，構造化データの定義機能と，SQLの構文に類似したクエリ機能，トランザクション機能，Java(TM)及びC言語のAPI

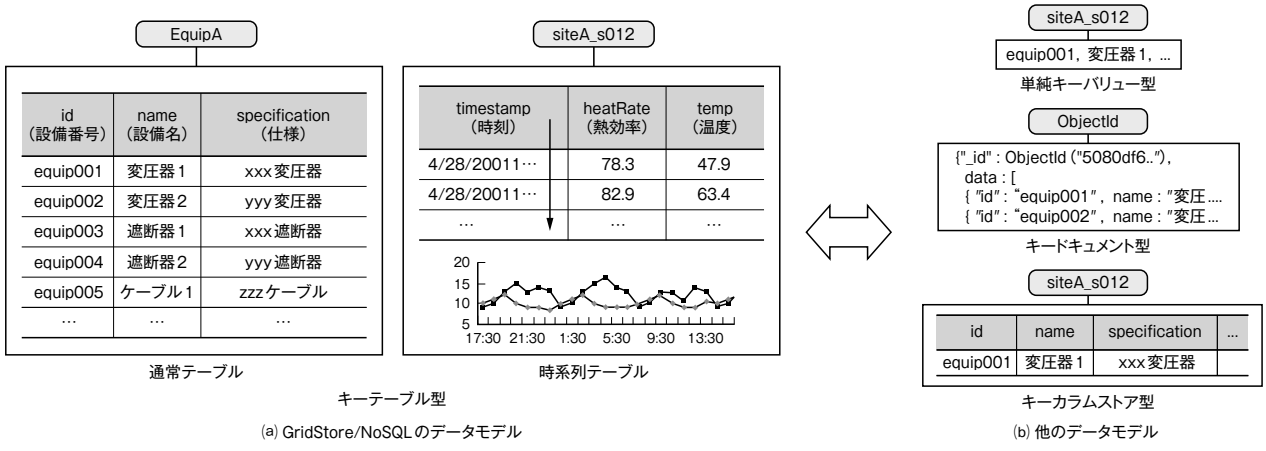


図2．GridStore/NoSQLのデータモデル例 — 通常テーブルと時系列テーブルという二つのキーテーブルの例である。

Examples of data models of GridStore/NoSQL

(注1)　全ノード数の過半数を占めたDBクラスタだけを正規のクラスタとして扱うこと。

（Application Programming Interface）をサポートしており，RDBユーザーがスムーズに導入できるようになっている。**図2**に示すように，キーバリューデータをキーテーブルと呼ぶレコードの集合体で表現する。これはRDBのテーブル名とテーブルの関係に類似している。また，センサデータ管理向けの応用機能も備わっている。

3 GridStore/NewSQL

3.1 GridStore/NewSQLの特長

GridStore/NoSQLのクエリ機能を大幅に強化したのがGridStore/NewSQLである。そのシステム構成を**図3**に示す。構成上の特長は，NoSQLエンジンの上位にSQLエンジンが付け加わっていることである。このSQLエンジンを通して，SQL処理を分割し複数のGridStoreノード（GSノード）に分散して割り当て，並列処理することでSQL処理の性能を向上させている。

GridStore/NewSQLの特長を以下に述べる。

(1) SQL機能とNoSQL機能のデュアルインタフェース

ANSI（米国規格協会）によって制定されたSQL-92が持つSQL機能をサポートし，ODBC（Open Database Connectivity）とJDBC（Java(㊟) Database Connectivity）というRDB標準のインタフェースを提供している。これらはSQLのスキルと親和性が高く，導入が容易になる。更に，これらのインタフェースを使って既存のBI（Business Intelligence）ツール(注2)やETL（Extract，Transform，Load）ツール(注3)との連携も可能である。

また，テーブルに対して高速なPutや，Get，RemoveなどのNoSQLが持つネイティブ命令が利用可能である。インターネットで得られる膨大なIoT（Internet of Things）データを登録する場合，このネイティブな命令を使えば，SQLを利用するのと比べて10倍以上の高速化が可能である。

(2) スケールアウト性と高可用性の維持　従来のRDBは，データ量やアクセス数の増加に伴い柔軟にスケールアウトできない課題があった。また，分散KVSはSQLを使ってデータアクセスできないという課題があった。GridStore/NewSQLは，GridStore/NoSQLが持つスケールアウト性や高可用性のメリットを生かしつつSQL機能を使うことができる。加えて，CPUのコア数増加やメモリ増加による性能改善，すなわちスケールアップも可能である。

(3) SQL処理のオーバヘッド削減と並列化　GridStore/NoSQLのキーテーブルをそのままRDBのデータモデルに直接マッピングし，データマッピングコストを最小化することで，GridStore/NoSQLの処理能力をむだなくSQL処理に生かすことができる。

また，複数のGSノードがリソースを同時に共有しないで並列処理するMPP（Massively Parallel Processing）アーキテクチャを採用して，複雑で非定型的なSQL処理の高速化を図っている。MPPアーキテクチャを採用しているDBとして，シェアドナッシング型のデータウェアハウス（DWH：Data Warehouse）が挙げられる。DWHはカラム単位のレコード圧縮をデータ登録時に行うことで，検索時のデータスキャンを高速化させている。このため，DWHは分析処理に適している反面，登録性能が遅く，更新機能も弱い。そのため，変化する大量のデータを取り込んでリアルタイムに分析するのは苦手である。

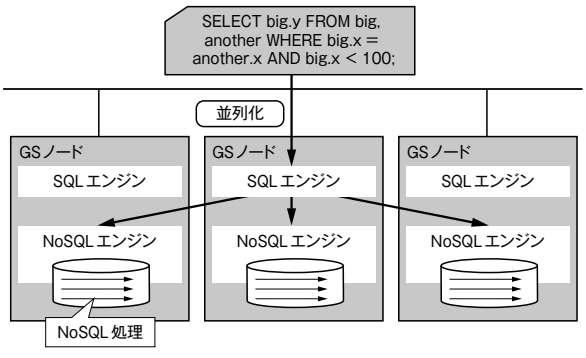


図3. GridStore/NewSQLのシステム構成 ― GSノードにSQLエンジンを付け加えることで，SQL機能を実現した。
Configuration of GridStore/NewSQL database system

(注2)　企業内の膨大なデータを意思決定に役だてる手法やツール。
(注3)　企業内のデータを抽出し，変換して，別のDBに書き出す手法やツール。

前述のように，GridStore/NewSQLは登録・更新機能と検索機能のバランスを取ることで，変化する大量のデータを取り込んでリアルタイムに分析する基盤に適している。GridStore/NoSQL及びGridStore/NewSQLの性能をRDB，DWH，既存KVSの各種DBと比較して**表1**に示す。

3.2 データ分散とパラレルクエリ

GridStore/NewSQLにおけるテーブル作成，レコード登録，及びレコード検索の流れを**図4**に示し，以下に述べる。

(1) テーブル作成　CREATE TABLE文を使ってテーブルを作成する。巨大なテーブルを作成する場合，複数のテーブルに分割するテーブルパーティショニングのオプションを使ってテーブルを定義する。SQLエンジンは，クライアントから送信されるCREATE TABLE文を解析し，Putや，Get，Removeというネイティブ命令から成る最適なNoSQL処理プランを生成し，NoSQLエンジン群にプランを同期して実行するよう要求する。テーブルパーティショニング情報は関連するメタテーブルに格納される。

(2) レコード登録　クライアントからINSERT文，UPDATE文，DELETE文を使って，それぞれレコード登録，レコー

一般論文

表1. 各種DBの性能比較
Comparison of various databases

項　目	GridStore/ NewSQL	GridStore/ NoSQL	RDB	DWH	一般的 KVS
拡張性	○	○	×	△	△
可用性	○	○	△~○	×~△	△
SQL対応	○	△	○	○	×
検索性能	△	○	△	○	△
更新性能	△	○	△	×	△
トランザクション対応	△	△	○	×~△	×
永続性	○	○	○	○	△

○：良い　△：普通　×：悪い

ド更新，及びレコード削除を行う。

(3) レコード検索　SQLエンジンは，クライアントから送信されるSQL文を解析し，最適なNoSQL処理プランと，NoSQL処理結果を統合するための結果統合プランを作成する。更に，作成したNoSQL処理プランをNoSQLエンジン群に非同期に実行するよう要求することで，複数のNoSQLエンジンが並列動作する。プラン生成において，パーティション情報によるテーブル展開や，プレディケートプッシュダウン，アグリゲーションプッシュダウンなどの最適化を行う[2]。プレディケートプッシュダウンとは，抽出すべきレコードの条件を指定するWHERE句の述語集合を可能な限りNoSQL処理プランに展開することで，レコード集合を早期にフィルタリングする最適化のことである。アグリゲーションプッシュダウンは，抽出したレコードを指定された軸で分類するGROUP BY句に対する同様の最適化である。

3.3 NoSQL機能とSQL機能のハイブリッド効果

NoSQL機能とSQL機能のハイブリッド構成が持つ特長を生かして，各アプリケーションの要件に応じてNoSQL機能とSQL機能の構成や形態を使い分けることができる（**図5**）。

タイプ1は，NoSQL機能だけを利用する性能重視の形態である。大規模データ集計処理や計算科学シミュレーションなどのハイパフォーマンスコンピューティングやセンサデータを中心としたエネルギー監視などへの適用が挙げられる。

タイプ2は，NoSQL機能を分散メモリキャッシュとする形態であり，既存RDBのフロントエンドに設置して，分散メモリキャッシュから既存RDBに一括登録（バルク登録，バルク挿入）することで既存RDBの処理を高速化する効果がある。

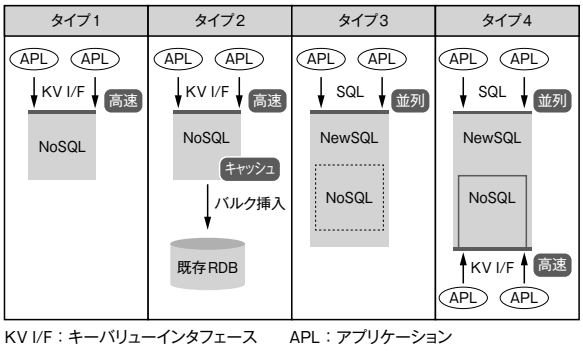


図5. GridStoreの利用形態 — 各アプリケーションの要件に応じて，NoSQL機能とSQL機能の構成や形態を使い分けることができる。
Modes of GridStore utilization

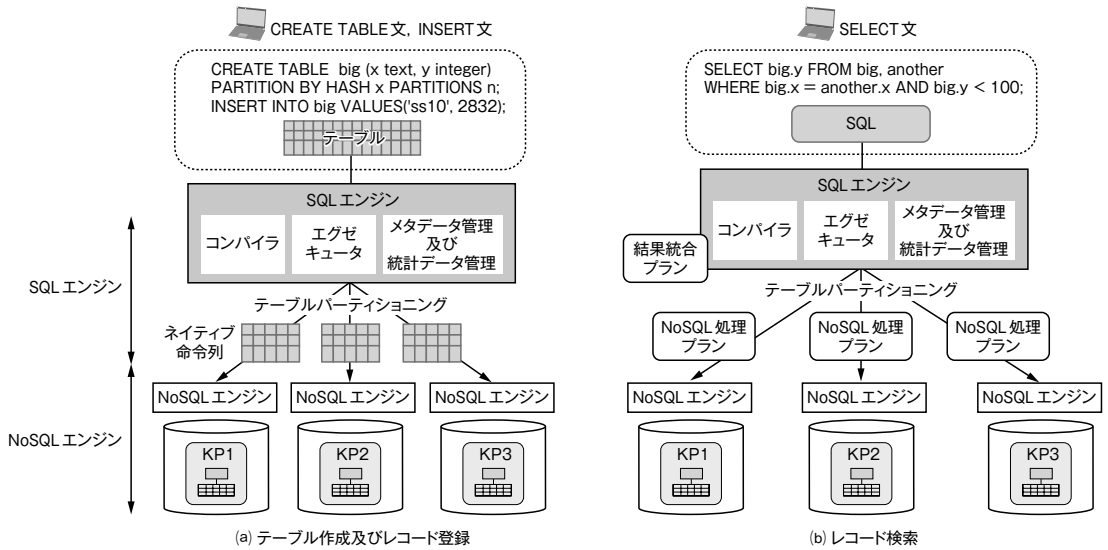


図4. GridStore/NewSQLの処理フロー — SQL機能を通してテーブル作成や，レコード登録，レコード検索を行う。CREATE TABLE文は，テーブル名bigのカラムxのハッシュ値でテーブルを分割するテーブル作成文の例で，SELECT文は，bigとanotherという二つのテーブルの結合を行うレコード検索の例である。
Flow of processes of GridStore/NewSQL database

タイプ3は，既存SQLアプリケーションをそのままにして，DBの処理を高速化したい場合の形態である。

タイプ4は，SQL機能とNoSQL機能のデュアル構成であり，ネットワークを通して送り込まれる大量のIoTデータの登録や更新はNoSQL機能で高速に処理しつつ，参照や分析はSQLで行いたい場合に適している。

4　性能検証

RDBのベンチマークTPC-H(注4)のデータセットを基に応答性とスケールアウト性に関する性能検証を行った。比較対象は，OSS（Open Source Software）の分散フレームワークHadoop(†)系のSQLエンジンである。ハードウェアとして，CPUはIntel® Xeon® E5-2420，メモリは48 Gバイト，ストレージはSATA（Serial Advanced Technology Attachment）1 T（テラ：10^{12}）バイトHDD（ハードディスクドライブ），ネットワークはギガビットEthernet(†)を使用した。

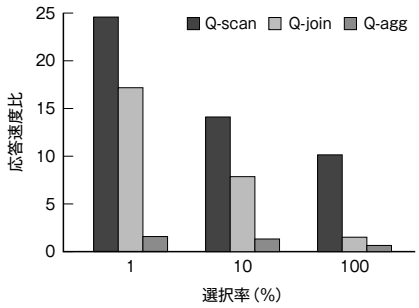


図6．Hadoop系SQLエンジンとの応答速度比 — Q-scanはレコードの選択率1～100 %全てで，Q-joinは1 %と10 %で高い応答速度比が得られた。

Comparison of response speed ratios of NewSQL database and SQL engines for Hadoop

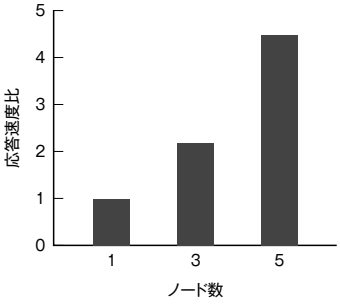


図7．ノード数と応答速度の関係 — Q-scan，Q-join，及びQ-aggを統合したクエリでは，ノード数に応じて応答速度が増加している。

Response speed ratio according to number of nodes

（注4）　TPC（Transaction Processing Performance Council）により策定された意志決定支援システム向けのベンチマーク。

属性条件を評価しテーブル全体をスキャンするクエリQ-scan，結合演算を伴うクエリQ-join，及び集計演算を伴うクエリQ-aggについて，3ノード構成での応答速度比を**図6**に示す。クエリでレコードの選択率を変化させているが，Q-scanはレコードの選択率1～100 %全てで，Q-joinは1 %と10 %で，約10倍以上の速度を達成した。また，それ以外の場合についても，並列化技術を深耕することで，更なる高速化が可能であることがわかった。

次に，これら3クエリを統合したクエリについて，最大5ノードでのスケールアウト性を検証した。**図7**に示すように，ノード数に応じて応答速度が増加していることが確認できた。これは主に，テーブルパーティショニング機能やMPPアーキテクチャによる効果である。実際，各サーバにおいてデータ読出し処理が同時並行で行われていることを確認できた。

5　あとがき

GridStore/NoSQLに対してSQL機能を強化したGridStore/NewSQLについて述べた。GridStore/NewSQLにより，コモディティハードウェアでクラスタを安価に構築しながら，従来のRDBでは処理できない規模の大容量データを管理することが可能になった。今後も，更なる高速化を図っていく。

文　献

(1)　服部雅一 他．M2Mビジネスを支えるスケールアウト型データベース GridStore™/NoSQL．東芝レビュー．**69**，7，2014，p.23－27.

(2)　片山大河．データベース統合エンジン．東芝レビュー．**68**，9，2013，p.56－57.

服部　雅一　HATTORI Masakazu

インダストリアルICTソリューション社 IoTテクノロジーセンター 先端ソフトウェア開発部主幹。スケールアウト型データベースの研究・開発に従事。情報処理学会，日本データベース学会会員。

IoT Technology Center

浜口　泰平　HAMAGUCHI Taihei

インダストリアルICTソリューション社 IoTテクノロジーセンター 先端ソフトウェア開発部。スケールアウト型データベースの研究・開発に従事。

IoT Technology Center

近藤　雄二　KONDO Yuji

東芝ソリューション（株）プラットフォームセンター ソフトウェア開発部参事。データベースの設計，開発，及び製品化に従事。

Toshiba Solutions Corp.