(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7333424号
(P7333424)

(45)発行日　令和5年8月24日(2023.8.24)

(24)登録日　令和5年8月16日(2023.8.16)

(51)国際特許分類
Ｇ０６Ｆ　16/28　(2019.01)
Ｇ０６Ｆ　16/245(2019.01)

ＦＩ
Ｇ０６Ｆ　16/28
Ｇ０６Ｆ　16/245

請求項の数　9　外国語出願　（全74頁）

(21)出願番号　特願2022-2960(P2022-2960)
(22)出願日　令和4年1月12日(2022.1.12)
(62)分割の表示　特願2019-514315(P2019-514315)の分割
原出願日　平成29年9月12日(2017.9.12)
(65)公開番号　特開2022-62036(P2022-62036A)
(43)公開日　令和4年4月19日(2022.4.19)
審査請求日　令和4年2月1日(2022.2.1)
(31)優先権主張番号　62/395,216
(32)優先日　平成28年9月15日(2016.9.15)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　15/700,862
(32)優先日　平成29年9月11日(2017.9.11)
(33)優先権主張国・地域又は機関
最終頁に続く

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
弁理士法人深見特許事務所
(72)発明者　パーク，ホヨン
アメリカ合衆国、９５１２９ カリフォルニア州、サン・ノゼ、オスナ・プレイス、１１９６
(72)発明者　ゲイシュテル，ジェルジ
ハンガリー、１０２９ ブダペスト、マグドルナ・ストリート、１
最終頁に続く

(54)【発明の名称】　分散イベント処理システムのためのグラフ生成

(57)【特許請求の範囲】
【請求項１】
コンピュータによって実行される方法であって、前記コンピュータは、前記方法を実行する１つ以上のプロセッサと、前記方法を前記コンピュータに実行させるためのプログラムを格納したメモリとを備え、前記方法は、
イベント処理システムによって処理されている連続クエリ言語（ＣＱＬ）クエリの段を判断するステップと、
前記段に対応付けられた段の種類を判断するステップと、
前記段の種類に少なくとも一部基づいて、前記段に対して計算すべき変換を判断するステップと、
複数のルールに少なくとも一部基づいて、前記ＣＱＬクエリの分類を判断するステップと、
パーティショニング基準を前記段に適用することにより、パーティションされた段またはパーティションされていない段として、前記段をマークするステップと、
前記段の前記パーティショニング基準に少なくとも一部基づいて、前記段のデータ変換パイプラインの有向非巡回グラフ（ＤＡＧ）における変換を生成するステップと、
前記変換に少なくとも一部基づいて、前記段のパーティショニングを判断するステップとを含む、方法。
【請求項２】
前記段に対して計算すべき変換を判断するステップは、前記ＣＱＬクエリを１つ以上の

トークンにパースすることを含む、請求項１に記載の方法。
【請求項３】
前記複数のルールは、前記ＣＱＬクエリを、ステートレスクエリ、セミステートフルクエリ、またはフルステートフルクエリのうちの少なくとも１つに分類するためのルールを含む、請求項１または２に記載の方法。
【請求項４】
前記パーティショニング基準は、１つ以上のパーティショニング属性を含む、請求項１に記載の方法。
【請求項５】
前記ＣＱＬクエリの意味解析を実行するステップをさらに含む、請求項４に記載の方法。
【請求項６】
前記１つ以上のパーティショニング属性は、前記ＣＱＬクエリの前記意味解析に少なくとも一部基づいて判断される、請求項５に記載の方法。
【請求項７】
前記データ変換パイプラインを前記データ変換パイプラインのソースから前記データ変換パイプラインのシンクまでトラバースするステップをさらに含む、請求項１～６のいずれか１項に記載の方法。
【請求項８】
請求項１～７のいずれかに記載の方法をコンピュータに実行させるためのプログラム。
【請求項９】
請求項８に記載のプログラムを記憶するためのメモリと、
前記プログラムを実行するためのプロセッサとを備える、システム。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願の相互参照
本願は、米国特許法第１１９条（ｅ）に従い、２０１６年９月１５日に出願され「FAST SERIALIZATION OF TUPLE BATCHES」と題された米国仮出願第６２／３９５，２１６号および２０１７年９月１１日に出願され「GRAPH GENERATION FOR A DISTRIBUTED EVENT PROCESSING SYSTEM」と題された米国非仮出願第１５／７００，８６２号に基づく利益および優先権を主張し、各々の内容全体をあらゆる目的のために本明細書に引用により援用する。
【０００２】
本願はまた、２０１７年９月１１日に出願され「DATA SERIALIZATION IN A DISTRIBUTED EVENT PROCESSING SYSTEM」と題された代理人整理番号０８８３２５－１０４３３４（１７７６００ＵＳ）の出願第１５／７００７８４号、２０１７年９月１１日に出願され「CLUSTERING EVENT PROCESSING ENGINES」と題された代理人整理番号０８８３２５－１０４３３６（１７７６２０ＵＳ）の出願第１５／７００９１４号、および２０１７年９月１１日に出願され「DATA PARTITIONING AND PARALLELISM IN A DISTRIBUTED EVENT PROCESSING SYSTEM」と題された代理人整理番号０８８３２５－１０４３３７（１７７６３０ＵＳ）の出願第１５／７０１０１９号に関連する。米国特許法第１２０条に基づき、各出願の内容全体を、本明細書に完全に記載されているかのごとく、本明細書に引用により援用する。
【背景技術】
【０００３】
背景
従来のデータベースシステムにおいて、データは、通常はテーブルの形態である１つ以

上のデータベースに記憶される。そして、記憶されるデータは、構造化照会言語（ＳＱＬ）などのデータ管理言語を使用して照会および操作される。たとえば、ＳＱＬクエリは、データベースに記憶されるデータから関連するデータを識別するために定義および実行され得る。したがって、ＳＱＬクエリは、データベースに記憶されるデータの有限集合に対して実行される。さらに、ＳＱＬクエリが実行される時、それはひとたび有限データ集合に対して実行され、有限の静的結果（finite static result）を作成する。したがって、データベースは、有限の記憶されるデータ集合に対してクエリを実行するように最良に実装される。

【０００４】

しかしながら、いくつかの最新のアプリケーションおよびシステムは、有限のデータ集合の代わりに、連続的なデータまたはイベントのストリームの形態のデータを生成する。このようなアプリケーションの例としては、限定されるものではないが、センサデータアプリケーション、株式相場表示装置、ネットワーク性能測定ツール（たとえば、ネットワークモニタリングおよびトラフィック管理アプリケーション）、クリックストリーム分析ツール、自動車トラフィックモニタリングなどが含まれる。このようなアプリケーションは、データストリームを処理することができる新しい種類のアプリケーションの必要を生じさせた。たとえば、温度センサは、温度測定値を発信するように構成され得る。

【０００５】

これらのタイプのイベントストリームベースのアプリケーションについてのデータの管理および処理では、厳密な時間によるフォーカスを使用してデータ管理およびクエリ機能が作成される。連続的なデータの非有界集合に対する長期実行クエリ（long-running queries）を含む、異なる種類の照会メカニズムが必要である。現在、一部のベンダーはイベントストリーム処理を目的とした製品スーツを提供しているが、提供されるこれらの製品は、今日のイベント処理の需要に対処するために必要な処理の柔軟性に依然として欠けている。

【発明の概要】

【課題を解決するための手段】

【０００６】

簡単な概要

イベントストリームのイベントを処理するための技術（たとえば、方法、システム、１つ以上のプロセッサによって実行可能なコードまたは命令を格納する非一時的なコンピュータ可読媒体）が提供される。ある実施形態において、イベント処理システムが開示される。システムは、アプリケーションを特定する情報を受信し、アプリケーションを特定する情報に基づいてアプリケーションの共通アプリケーションランタイムモデルを生成するように構成される。システムは、アプリケーションの共通アプリケーションランタイムモデルをアプリケーションの第１のジェネリック表現に変換するように構成される。特定の例において、アプリケーションの第１のジェネリック表現は、複数のターゲットイベント処理システムのうちの第１のターゲットイベント処理システムにおいて実行されるように構成される。特定の例において、アプリケーションの第１のジェネリック表現は、アプリケーションのコンポーネントのランタイム有向非巡回グラフ（Directed Acyclic Graph）（ＤＡＧ）を含む。

【０００７】

特定の実施形態において、イベント処理システムは、アプリケーションの第１のジェネリック表現を、第１のターゲットイベント処理システムによる実行のために、第１のターゲットイベント処理システムに送信するように構成される。

【０００８】

特定の実施形態において、アプリケーションはコンポーネントのイベント処理ネットワーク（ＥＰＮ）として表現され、アプリケーションを特定する情報は、ＥＰＮ構成情報、クエリ情報、およびアプリケーションに対応付けられたルールを含む。

【０００９】

特定の実施形態において、アプリケーションの共通アプリケーションランタイムモデルを生成することは、アプリケーションを、１つ以上の構成ブロックのセットとして表すことを含む。いくつかの例において、各構成ブロックは、メタデータが対応付けられているイベントビーンを表す。

【００１０】

特定の実施形態において、１つ以上の構成ブロックのセットは、インバウンドソケットイベントビーン、アウトバウンドソケットイベントビーン、連続クエリ言語（ＣＱＬ）プロセッサイベントビーン、または１つ以上のチャネルイベントビーンのうちの少なくとも１つを含む。

【００１１】

特定の実施形態において、アプリケーションの共通アプリケーションランタイムモデルをアプリケーションの第１のジェネリック表現に変換することは、アプリケーションの共通アプリケーションランタイムモデルにおいて表される１つ以上の構成ブロックを、アプリケーションのコンポーネントのランタイムＤＡＧに変換することを含む。

【００１２】

特定の実施形態において、イベント処理システムは、アプリケーションの共通アプリケーションランタイムモデルを、複数のターゲットイベント処理システムのうちの第２のターゲットイベント処理システムにおける実行のために、アプリケーションの第２のジェネリック表現に変換するように構成される。特定の例において、第２のターゲットイベント処理システムは、第１のターゲットイベント処理システムと異なる。

【００１３】

アプリケーションの共通アプリケーションランタイムモデルをアプリケーションの第２のジェネリック表現に変換することは、アプリケーションの共通ランタイムアプリケーションモデルにおいて表される１つ以上の構成ブロックを、アプリケーションのターゲット表現に変換することを含む。

【００１４】

特定の実施形態において、イベント処理システムは、アプリケーションのターゲット表現を、第２のターゲットイベント処理システムによる実行のために送信するように構成される。いくつかの例において、第２のターゲットイベント処理システムはイベントプロセッサシステムを含む。

【００１５】

上記および下記の技術は、多くの方法で、および多くの状況で実施することができる。以下でより詳細に説明するように、いくつかの例示的な実現例および状況が、以下の図を参照して提供される。しかしながら、以下の実現例および状況は多くのもののうちのほんの一部である。

【図面の簡単な説明】

【００１６】

【図１】本開示の実施形態に係る、異なる実行環境でイベント処理アプリケーションを処理できる環境を提供するイベント処理システムアーキテクチャの一例の局面を示す図である。

【図２】本開示の実施形態に係る、イベント処理アプリケーションのためのイベント処理ネットワーク（ＥＰＮ）を図示する図である。

【図３】本開示の実施形態に係る、アプリケーション処理エンジンのコンポーネントを示す簡略化されたブロック図である。

【図４】本開示の実施形態に係る、共通アプリケーションモデルジェネレータによって生成された「共通アプリケーションランタイムモデル」を表現したものの一例を示す図である。

【図５】本開示の実施形態に係る、ＤＡＧジェネレータが生成したコンポーネントのランタイム有向非巡回グラフ（ＤＡＧ）の一例を示す図である。

【図６】本開示の実施形態を組み込むことが可能なイベント処理システムの簡略化された

ハイレベル図である。
【図７】本開示の実施形態に係る、分散イベント処理システムのコンポーネントを示すブロック図である。
【図８】本開示の一実施形態に係る、アプリケーションの共通アプリケーションランタイムモデルを生成するためにイベント処理アプリケーションを処理するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図９】本開示の別の実施形態に係る、アプリケーションの共通アプリケーションランタイムモデルを生成するためにイベント処理アプリケーションを処理するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１０】本開示の実施形態に係る、分散イベント処理システムのコンポーネントを示す簡略化されたブロック図である。
【図１１】本開示の実施形態に係る、レジリエント分散データセット（ＲＤＤ）オブジェクトにおけるデータのシリアライズおよびデシリアライズを実行するためのプロセスのハイレベルデータフローを示す図である。
【図１２】本開示の実施形態に係る、イベントのバッチに含まれるデータをシリアライズできる一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１３Ａ】本開示の実施形態に係る、イベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１３Ｂ】本開示の実施形態に係る、精度低減圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１３Ｃ】本開示の実施形態に係る、通常の圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１３Ｄ】本開示の実施形態に係る、精度低減値インデックス圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１４】本開示の実施形態に係る、イベントの非数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１５】本開示の実施形態に係る、イベントストリームにおけるイベントの属性のデータタイプの判断に基づいてイベントストリームデータをシリアライズできる手法の一例を示す図である。
【図１６】本開示の実施形態に係る、イベントのバッチに含まれるデータをデシリアライズできる一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１７】本開示の実施形態に係る、イベントのバッチにおけるイベントの１つ以上の属性のデシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１８】本開示の実施形態に係る、値インデックス圧縮を用いてイベントのバッチにおけるイベントの数値属性または非数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図１９】本開示の実施形態に係る、精度低減圧縮技術を用いてイベントのバッチにおけるイベントの数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図２０】本開示の実施形態に係る、イベントのバッチにおけるイベントの数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図２１】本開示の実施形態に係る、イベントのバッチにおけるイベントの数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明する

プロセスのフロー図の一例を示す図である。
【図２２】本開示の実施形態に係る、複数のＣＥＰエンジンをスケジューリングし管理するために構成された分散イベント処理システムのコンポーネントを示す簡略化されたブロック図である。
【図２３】本開示の実施形態に係る、複数のＣＥＰエンジンをスケジューリングし管理するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図２４】本開示の実施形態に係る、データのパーティショニングおよびパラレル処理のために構成された分散イベント処理システムのコンポーネントを示す簡略化されたブロック図である。
【図２５】本開示の実施形態に係る、クエリ句およびオブジェクトＩＤを用いてデータを自動的にパーティショニングおよびパラレル処理するための一組の動作を説明するプロセスのフロー図の一例を示す図である。
【図２６】本開示の実施形態を実装するための分散システムの簡略化されたブロック図を示す。
【図２７】本開示の実施形態に係る、実施形態のシステムの１つ以上のコンポーネントが提供するサービスをクラウドサービスとして提供することを可能にするシステム環境の１つ以上のコンポーネントの簡略化されたブロック図である。
【図２８】本開示の実施形態を実装するために使用することが可能なコンピュータシステムの一例を示す図である。
【発明を実施するための形態】
【００１７】
詳細な説明
以下の説明では、さまざまな実施形態について説明する。説明のために、具体的な構成および詳細は、実施形態の完全な理解を提供するために記載される。しかしながら、実施形態が具体的な詳細なしに実施されてもよいことも、当業者には明らかであろう。さらに、周知の特徴は、説明される実施形態を不明瞭にしないために省略または簡略化されてもよい。
【００１８】
<u>複合イベント処理（ＣＥＰ）の概要</u>
複合イベント処理（Complex Event Processing）（ＣＥＰ）は、イベント駆動型アーキテクチャに基づいてアプリケーションを構築するためのモジュール式プラットフォームを提供する。ＣＥＰプラットフォームの中心には、アプリケーションが宣言型ＳＱＬ状言語を使用してデータのストリームに対してパターンマッチング操作をフィルタリング、照会および実行できるようにする連続問い合わせ言語（Continuous Query Language）（ＣＱＬ）がある。開発者はアプリケーションを書くためにＣＱＬを軽量なＪａｖａ（登録商標）プログラミングモデルと組み合わせて使用できる。他のプラットフォームモジュールには、機能豊富なＩＤＥ、管理コンソール、クラスタ化、分散キャッシュ化、イベントリポジトリ、およびモニタリングなどがある。
【００１９】
イベント駆動型アーキテクチャおよび複合イベント処理はエンタープライズコンピューティング環境の顕著な機能となっているため、ＣＥＰ技術を使用してミッションクリティカルなアプリケーションを構築し始めている企業が増えている。今日、ミッションクリティカルなＣＥＰアプリケーションは、さまざまな業界で見出され得る。たとえば、ＣＥＰ技術は、電力産業においては、電気需要の変化に瞬時に反応できるようにすることにより、ユーティリティをより効率的にするために使用されている。ＣＥＰ技術は、クレジットカード業界では、潜在的に不正な取引がリアルタイムで発生した場合にそのような取引を検出するために使用されている。ミッションクリティカルなＣＥＰアプリケーションのリストは増加を続けている。ミッションクリティカルなアプリケーションを構築するためにＣＥＰ技術を使用することにより、ＣＥＰアプリケーションの高可用性とフォールトトレ

ラント化が求められるに至った。

【００２０】

今日の情報技術（ＩＴ）環境は、金融市場およびネットワークパフォーマンスのモニタリングから、ビジネスプロセスの実行およびＲＦＴＤタグ付き資産のトラッキングまで、あらゆることに対して連続したデータのストリームを生成する。ＣＥＰは、ビジネス処理の有効性を向上させるためにイベント処理アプリケーションを開発するための豊富で宣言的な環境を提供する。ＣＥＰは、複数のイベントストリームを処理してパターンおよび傾向をリアルタイムで検出して、企業に対して、新たな機会を活用したり開発リスクを軽減したりするために必要な可視性を提供する。

【００２１】

連続したデータのストリーム（イベントストリームとも呼ばれる）は、明示的な終了のない事実上連続的または非有界であってもよいデータまたはイベントのストリームを含むことができる。論理的に、イベントまたはデータストリームは、データ要素（イベントともいわれる）のシーケンスであり得て、各データ要素は、関連付けられたタイムスタンプを有する。連続イベントストリームは、要素のバッグまたはセット（ｓ，Ｔ）として論理的に表わされ得て、ここで「ｓ」はデータ部分を表わし、「Ｔ」は時間ドメインに属する。「ｓ」部分は、概してタプルまたはイベントといわれる。したがって、イベントストリームは、タイムスタンプを持つタプルまたはイベントのシーケンスであり得る。

【００２２】

一部の局面において、ストリームにおけるイベントに関連付けられるタイムスタンプは、クロックタイムと同等とされ得る。しかしながら、他の例において、イベントストリームにおけるイベントに関連付けられる時間は、アプリケーションドメインによって定義され得て、クロックタイムに対応しない場合があるが、たとえば、代わりにシーケンス番号によって表わされ得る。このため、イベントストリームにおけるイベントに関連付けられる時間情報は、数字、タイムスタンプ、または時間の概念を表わす他の情報によって表され得る。入力イベントストリームを受信するシステムについては、イベントは、タイムスタンプの増加順でシステムに到達する。同じタイムスタンプを有する２つ以上のイベントがあり得る。

【００２３】

一部の例において、イベントストリームにおけるイベントは、いくぶん世俗的なイベント（たとえば、温度センサが値を新しい値に変更した場合や、株式表示記号の価格が変化した場合など）の発生を表わし得て、イベントに関連付けられる時間情報は、データストリームイベントによって表わされる世俗的なイベントが発生した場合を示し得る。

【００２４】

イベントストリームを介して受信したイベントについては、イベントストリームにおけるイベントが確実にタイムスタンプ値の増加順で到達するように、イベントに関連付けられる時間情報が使用され得る。これにより、イベントストリームにおいて受信されたイベントを、それらに関連付けられる時間情報に基づいて順序付けることができる。この順序付けを可能にするために、後に生成されるイベントが前に生成されるイベントよりも後のタイムスタンプを有するように、タイムスタンプがイベントストリームにおけるイベントに対して非減少の態様で関連付けられ得る。他の例として、シーケンス番号が時間情報として使用されている場合、後に生成されるイベントに関連付けられるシーケンス番号は、前に生成されるイベントに関連付けられるシーケンス番号よりも大きくなり得る。一部の例において、たとえばデータストリームイベントによって表される世俗的なイベントが同時に発生した場合に、複数のイベントが、同じタイムスタンプまたは同じシーケンス番号に関連付けられ得る。同じイベントストリームに属するイベントは、概して、関連付けられる時間情報によってイベントに対して加えられる順序で処理され得て、前のイベントは後のイベントの前に処理される。

【００２５】

イベントストリームにおけるイベントに関連付けられる時間情報（たとえば、タイムス

タンプ）は、ストリームのソースによって設定され得る、または代替的にストリームを受信するシステムによって設定され得る。たとえば、特定の実施形態において、イベントストリームを受信するシステムに対してハートビートが維持され得て、イベントに関連付けられる時間は、ハートビートによって測定されるようにシステムへのイベントの到達の時間に基づき得る。イベントストリームにおける２つのイベントが同じ時間情報を有することは可能である。なお、タイムスタンプ順序付け要件は１つのイベントストリームに対して特定のものであるが、異なるストリームのイベントが適宜インターリーブされ得る。

【００２６】

イベントストリームは、関連付けられるスキーマ「Ｓ」を有し、スキーマは、時間情報と、１つ以上の指定される名前付き属性の集合とを含む。特定のイベントストリームに属するすべてのイベントは、その特定のイベントストリームに関連付けられるスキーマに準拠する。このため、イベントストリーム（ｓ，Ｔ）については、イベントストリームは、スキーマ「Ｓ」を(〈time_stamp〉,〈attribute(s)〉)として有し得て、ここで〈attributes〉
は、スキーマのデータ部分を表わし、１つ以上の属性を含み得る。たとえば、株式相場表示装置イベントストリームについてのスキーマは、〈stock symbol〉および〈stock price〉
の属性を含み得る。このようなストリームを介して受信される各イベントは、タイムスタンプと２つの属性とを有する。たとえば、株式相場表示装置イベントストリームは、以下のイベントおよび関連付けられるタイムスタンプを受け取り得る。

【００２７】

…
(〈timestamp_N〉,〈NVDA,4〉)
(〈timestamp_N+1〉,〈ORCL,62〉)
(〈timestamp_N+2〉,〈PCAR,38〉)
(〈timestamp_N+3〉,〈SPOT,53〉)
(〈timestamp_N+4〉,〈PDCO,44〉)
(〈timestamp_N+5〉,〈PTEN,50〉)
…

上記のストリームにおいて、ストリーム要素(〈timestamp_N+1〉,〈ORCL,62〉)については
、イベントは、「stock_symbol」および「stock_value」を伴う〈ORCL,62〉である。ストリーム要素に関連付けられるタイムスタンプは、「timestamp_N+1」である。したがって、
連続イベントストリームはイベントのフローであり、各イベントは同じ一連の属性を有する。

【００２８】

記載したように、ストリームはＣＱＬクエリが作用し得るデータの主要なソースであり得る。ストリームＳは、要素（ｓ，Ｔ）のバッグ（「マルチセットともいわれる）であり得て、ここで「s」はＳのスキーマに属し、「Ｔ」は時間ドメインに属する。加えて、ストリーム要素は、タプルとタイムスタンプのペアであり得て、タイムスタンプを持つタプル挿入のシーケンスとして表わされ得る。言い換えると、ストリームは、タイムスタンプを持つタプルのシーケンスであり得る。一部の場合において、同じタイムスタンプを有する２つ以上のタプルがあり得る。加えて、入力ストリームのタプルは、タイムスタンプの増加順でシステムに到達する必要があり得る。代替的に、リレーション（「時間で変化するリレーション」ともいわれ、リレーショナルデータベースからのデータを含み得る「リレーショナルデータ」と混同されない）は、時間ドメインからスキーマＲのタプルの非有界バッグへのマッピングであり得る。一部の例において、リレーションは、順序付けされていない、時間で変化するタプルのバッグ（すなわち、瞬間的なリレーション）であり得る。一部の場合において、時間の各瞬間では、リレーションは有界集合であり得る。これ

は、挿入、消去、および／または更新を含んでリレーションの変化する状態を捕捉し得る、タイムスタンプを持つタプルのシーケンスとしても表わされ得る。ストリームと同様に、リレーションは、リレーションの各タプルが準拠し得る固定スキーマを有し得る。さらに、本願明細書で使用される連続クエリは、概してストリームおよび／またはリレーションのデータを処理する（すなわち、照会する）ことが可能であり得る。加えて、リレーションは、ストリームのデータを参照し得る。

【００２９】

イベント処理アプリケーション

ローインフラストラクチャおよびビジネスイベントの両方の量ならびに速度は、ＩＴ環境で急激に増加している。それは、金融サービスのために株式データをストリーミングするのであれ、軍事用に衛星データをストリーミングするのであれ、運輸および物流ビジネスのためにリアルタイムの車両位置データをストリーミングするのであれ、複数の業界の企業が大量の複雑なデータをリアルタイムで処理しなければならない。さらに、モバイル機器の急増および高速接続の普及は、モバイルデータの急増を助長する。同時に、ビジネスプロセスの俊敏性および実行に対する需要も高まっている。これら２つの傾向は、組織に対して、組織の、実装のイベント駆動型アーキテクチャパターンをサポートする能力を高めるよう、圧力をかけている。リアルタイムのイベント処理では、インフラストラクチャおよびアプリケーション開発環境の両方がイベント処理要件を実行する必要がある。これらの要件には、おそらくは秒単位ではなくマイクロ秒単位の応答時間で測定されるレイテンシで日常のユースケースから非常に高速のデータおよびイベントのスループットへ基準化するニーズが含まれることがよくある。さらに、イベント処理アプリケーションは、これらのイベントの流れの中で複雑なパターンを検出しなければならないことがよくある。

【００３０】

Oracle Stream Analyticsプラットフォームは、多くの業界と機能分野をターゲットとしている。以下は、いくつかのユースケースである。

【００３１】

電気通信：リアルタイム呼詳細（ＣＤＲ）レコードモニタリングおよび分散型サービス妨害攻撃検出を実行する能力。

【００３２】

金融サービス：ミリ秒またはマイクロ秒のウィンドウ内に存在する裁定機会を利用する能力。金融証券取引のリアルタイムのリスク分析、モニタリングおよび報告を実行し、外国為替価格を計算する能力。

【００３３】

運輸：地方または目的地の都市の天候、地上業務員のオペレーション、空港のセキュリティなどによる飛行の不具合が発生した場合に、旅客警報を発令して手荷物の場所を検出する能力。

【００３４】

公共部門/軍：分散した地理上の敵の情報を検出し、それを抽象化し、高確率の敵の攻撃を解読する能力。緊急事態に対応するために最も適切なリソースに警告する能力。

【００３５】

保険：潜在的に不正な請求を学習し、検出する能力。

ＩＴシステム：故障したアプリケーションやサーバをリアルタイムで検出し、是正措置をトリガする能力。

【００３６】

サプライチェーンおよび物流：リアルタイムで出荷を追跡し、潜在的な到着の遅れを検出して報告する能力。

【００３７】

リアルタイムストリーミングおよびイベント処理分析

接続デバイス数の増加によるデータの爆発的増加に伴い、大量の動的に変化するデータが増加しており、データは組織内だけでなく、ファイアウォールの外側にも移動する。高

速データは、特に変わり易いビジネスプロセスに高い価値をもたらす。しかしながら、このデータの一部は短い時間枠でその運用上の価値を失う。ビッグデータは、有効な洞察のための処理において時間的な余裕がある。一方、ファストデータでは、非常に動的で戦略的なデータから最大値を抽出する必要がある。それははるかに速く処理することを必要とし、生成されたデータのできるだけ近くでタイムリーなアクションをとることを容易にする。Oracle Stream Analyticsプラットフォームは、即応性のあるファストデータを実現する。Oracle Edge Analyticsは、リアルタイムで実践的な洞察を得るために、ネットワークエッジに対するデータの処理、相関付け、フィルタリング、および分析をプッシュする。

【００３８】

Oracle Stream Analyticsプラットフォームは、着信ストリーミングイベントを永続データと結合する機能を提供し、それによってコンテキストを意識したフィルタリング、相関付け、集約およびパターンマッチングを実現する。それは、一般的なイベントソースのために、軽量で、そのまま使用できるアダプタを提供する。それは、カスタムアダプタ開発用の使いやすいアダプタフレームワークも提供する。このプラットフォームで、組織は、機会と、一見無関係のイベントによって表される脅威とを識別し、予測することができる。その増分処理パラダイムは、最小量のリソースを使用してイベントを処理することができ、極端に低いレイテンシの処理を可能にする。それはまた、それが、非常にタイムリーな警報を発令し、次のような紛失イベントや遅延イベントを即座に検出することを可能にする。

【００３９】

相関付けられたイベント：イベントＡが発生した場合、イベントＢはほぼ常にそれの後２秒以内に発生する。

【００４０】

紛失イベントまたはシーケンス外イベント：イベントＡ、Ｂ、Ｃは順番に発生するべきである。ＣはＡの直後に、Ｂなしで見られる。

【００４１】

原因となるイベント：製造品目の重量がゆっくりと低下する傾向にあるか、または読み取り値が許容基準外である。これは潜在的な問題または将来のメンテナンスの必要性を示している。

【００４２】

リアルタイムのイベントソーシングに加えて、Oracle Stream Analyticsプラットフォーム設計環境およびランタイム実行は、イベントストリームとデータベースや高性能データグリッドなどの永続データストアとの両方にわたる標準ベースの連続クエリ実行をサポートする。これにより、プラットフォームは、他の態様では気付かれないであろうパターンおよび傾向を識別するためにマイクロ秒単位または分単位で回答を必要とするシステムに対するインテリジェンスの中心として機能することができる。イベント処理のユースケースでは、標準データベースＳＱＬの数学的精度および信頼性を備えたインメモリ処理の速度が必要である。このプラットフォームクエリは、着信イベントストリームをリスニングし、クエリ最適化のための高度な自動化されたアルゴリズムを利用して、登録されたクエリを各イベントについてインメモリで継続的に実行する。インメモリ実行モデルに基づいているが、このプラットフォームはクエリ開発に標準のＡＮＳＩ　ＳＱＬ構文を利用しているため、クエリ構築の正確性および拡張性が保証される。このプラットフォームは、ＡＮＳＩ　ＳＱＬ’９９規格に完全に準拠しており、リアルタイムの連続クエリパターンマッチングのために標準ＳＱＬに対するＡＮＳＩ　ＳＱＬレビュー拡張をサポートするべく業界で利用可能な最初の製品の１つである。ＣＱＬエンジンはプロセッサ内のクエリの実行を最適化し、開発者は最適化よりもビジネスロジックに集中するようになる。

【００４３】

Oracle Stream Analyticsプラットフォームでは、ＳＱＬおよびＪａｖａコードの両方を組み合わせて、堅牢なイベント処理アプリケーションを提供できる。標準の業界用語を

利用してイベントソース、プロセッサ、およびイベント出力またはシンクを記述して、このプラットフォームは、アプリケーション内でイベントを定義および操作するためにメタデータ駆動型のアプローチを提供する。その開発者は、アプリケーションの設計に視覚的な有向グラフのキャンバスおよびパレットを用いて、イベントの流れおよびイベントとデータソースとの両方にわたる処理を迅速に概説する。ドラッグアンドドロップモデリングおよび設定ウィザードを介してフローを開発して、開発者は、次いで、適切なメタデータ定義を入力して設計を実装に結び付けることができる。必要または好まれる場合には、次いで、ワンクリックで、開発者は、カスタムＪａｖａコード開発に立ち寄ること、またはSpring（登録商標）フレームワークを直接使用して高度な概念を彼らのアプリケーションにコーディングすることができる。

【００４４】

イベント駆動型アプリケーションは、非常に高速のストリーミング入力データを処理しながら、低くかつ決定論的なレイテンシを与えることに対するニーズによって特徴付けられることがよくある。Oracle Stream Analyticsプラットフォームの基盤は、ＯＳＧｉ（登録商標）バックプレーンに基づく軽量のＪａｖａコンテナである。それは、セキュリティ、ロギング、および作業管理アルゴリズムなど、WebLogic JEEアプリケーションサーバからの成熟したコンポーネントが含むが、リアルタイムのイベント処理環境でそれらのサービスを利用する。統合されたリアルタイムカーネルは、独自のサービスを提供して、ＪＭＸフレームワークによってサポートされるスレッドおよびメモリ管理を最適化し、パフォーマンスおよび構成のためにコンテナとの対話を可能にする。Ｗｅｂ２．０リッチインターネットアプリケーションはＨＴＴＰパブリッシュおよびサブスクライブサービスを用いてプラットフォームと通信でき、これにより、それらは、アプリケーションチャネルにサブスクライブし、イベントをクライアントにプッシュされることができる。フットプリントが小さいため、このプラットフォームは軽量のＪａｖａベースのコンテナであり、製品化までの時間がより速くなり、総オーナーシップコストが削減される。

【００４５】

Oracle Stream Analyticsプラットフォームには、標準の商品ハードウェア上でマイクロ秒の処理レイテンシで、またはOracle Exalogicおよびその他の工学システムのそのポートフォリオで最適に、毎秒数百万のイベントを処理する機能がある。これは、高性能イベント処理のユースケースに設計重点を置いただけでなく、エンタープライズクラスのリアルタイム処理インフラストラクチャコンポーネントとの緊密な統合で、完全な「トップダウン」の階層化されたソリューションを通して達成される。パフォーマンス指向のサーバクラスタのプラットフォームアーキテクチャは、Oracle Coherence技術への緊密な統合で、信頼性、故障許容、および非常に高い柔軟性に重点を置き、企業はデータグリッド全体に渡ってミッションクリティカルなアプリケーションを予測可能に基準化でき、継続的なデータの可用性およびトランザクションの保全性を確保できる。

【００４６】

さらに、このプラットフォームでは決定論的処理が可能であり、つまり、同じイベントを異なるレートで複数のサーバまたは同じサーバに供給して、毎回同じ結果を得ることができる。これにより、稼働中のサーバのシステムクロックにのみ依存するシステムに対して驚くべき利点が得られる。

【００４７】

上記および下記の技術は、いくつかの方法およびいくつかの状況において実施され得る。以下においてより詳細に記載されるように、いくつかの例示的な実施および状況が以下の図面を参照して提供される。しかしながら、以下の実施および状況は多くのうちの一部である。

【００４８】

<u>分散イベント処理</u>

特定の状況において、企業のユーザは、企業内で生じる重要なイベントを素早く特定しそれに反応することで、このようなイベントを特定すると直ちに対応策を取ることができ

るようにすることを望む場合がある。たとえば、ユーザは、企業内の限界値を超えた販売注文に関連する重要なイベントを特定することを望む場合がある。このようなシナリオにおいて、ユーザは、データストア／データウェアハウスに対して１つ以上のクエリを提出し、数分または数時間ではなく数秒以内にクエリの結果を閲覧し、異常が検出された場合は直ちに対応策を取ることができるようにすることを望む場合がある。企業は、リアルタイムのデータ処理およびデータ解析を用いることにより、より多くの意思決定を行うためにリアルタイムでイベントストリームを処理し、数秒または数分で行動を起こすことが重要なときは直ちに対応策を取ることができるようにする場合がある。

【００４９】

本開示の実施形態に従い、ＣＥＰと分散イベントストリーム処理との組み合わせを用いて比較的素早くかつリアルタイムで非常に多くの量のデータを処理またはクエリすることができる分散イベント処理システムが開示される。分散イベント処理システムは、連続して受信したデータストリーム（たとえばライブフィード）に対してクエリ（たとえばＣＱＬクエリ）を実行することにより、データストリームをリアルタイムで処理することができる。分散イベント処理システムは、１つ以上の連続データストリームを受信し、データストリームに対する連続クエリを登録し、ストリーム内に新たなデータが現れるとクエリを連続的に実行することができる。この種の連続クエリは実行時間が長いので、分散イベント処理システムは、ユーザに対して結果の連続ストリームを提供することができる。

【００５０】

特定の実施形態において、開示されている分散イベント処理システムは、このシステム内のマシンのクラスタにアプリケーションの実行を分散させることにより、アプリケーション（たとえばイベント処理アプリケーション）をデプロイし実行するように構成し得る。本明細書に記載のイベント処理アプリケーションは、入力ストリームを処理するために使用される連続クエリの形態で表現することが可能な一組のルールを含み得る。連続クエリは、受信したイベントに対して実行すべき処理を特定する命令（たとえばロジック）を含み得る。これは、注目すべきイベントとしてどのようなイベントを選択しクエリ処理の結果として出力すべきかを含む。連続クエリは一般的に、フィルタリングおよびアグリゲーション機能を実行することにより、入力イベントストリームから注目すべきイベントを発見して抽出することが可能である。アプリケーションは、１つ以上の入力イベントストリームをリッスンし、１つ以上の入力イベントストリームから１つ以上の注目すべきイベントを選択するためにロジック（たとえばクエリ）を実行し、１つ以上の出力イベントストリームを介して、選択した注目すべきイベントを出力するように構成することができる。

【００５１】

たとえば、イベント処理アプリケーションは、一組の入力テキストの中の特定のワードに対するリファレンスの量をカウントするワードカウントアプリケーションを含み得る。このようなアプリケーションは、たとえば、一組のテキストを読み取り、各テキスト内に各ワードが現れる回数をカウントする連続クエリを含み得る。入力テキストは、たとえば、Facebook（登録商標）またはTwitter（登録商標）等のオンラインアプリケーションからのストリームにおいて受信したショートメッセージを含み得る。上述のように、連続クエリはＣＱＬ言語を用いて構成することが可能である。たとえば、ワードカウントストリーミングアプリケーションにおいて実行すべきワードカウントタスク／動作を指定するために、ユーザは、FROM location GROUP BY word SELECT countといった形態を取ることができるＣＱＬクエリを書き込むことができる。このようなクエリは、指定された場所からすべてのセンテンスを集め、これらのセンテンスから得た固有のワードを異なるグループに分けてから、各グループ内のワードの量をカウントすることができる。

【００５２】

アプリケ-ションの実行をマシンのクラスタに分散させることにより、開示されている分散イベント処理システムを、アプリケーションの実行に関連する結果をユーザに対して素早くかつリアルタイムで提供するように構成することが可能である。分散イベント処理システムは、アプリケーションに関連するデータを別々のコンピューティングノードにパ

ーティションするように構成することが可能であり、各コンピューティングノードは、別のコンピュータマシン上において別のファイルとして維持することができる。このような各マシンは、アプリケーション内のクエリを、その他のマシンと並列に、上記マシン上において維持されるデータに対し、実行するように構成することができる。

【００５３】

分散イベント処理システムのための効率的なＤＡＧの生成

本開示の特定の実施形態において、アプリケーション（たとえばイベント処理アプリケーション）に関する情報を処理するためのアプリケーション処理エンジンが開示される。アプリケーション処理エンジンは、イベント処理アプリケーションを特定する情報を受信するように構成される。特定の例において、イベント処理アプリケーションは、コンポーネントのイベント処理ネットワーク（Event Processing Network）（ＥＰＮ）として表現され、イベント処理アプリケーション情報を特定する情報は、イベント処理アプリケーションの各種コンポーネント（たとえば、アダプタ、プロセッサ、ストリーム、またはイベントビーン（bean））に関する情報を含む。たとえば、イベント処理アプリケーションを特定する情報は、イベントに関する、構成情報、クエリ情報、およびその他の種類の情報を含み得る。

【００５４】

特定の実施形態において、アプリケーション処理エンジンは、アプリケーションを特定する情報を処理しこのアプリケーションの「共通アプリケーションランタイムモデル」を生成するように構成し得る。本明細書に記載のように、アプリケーションの「共通アプリケーションランタイムモデル」は、１つ以上の構成ブロックのセットとしてアプリケーションを表現したものであり、各構成ブロックは処理ステージを表し、対応するメタデータがアプリケーションを記述する。アプリケーション処理エンジンは、アプリケーションの「共通アプリケーションランタイムモデル」を、アプリケーションの１つ以上のジェネリック表現に変換するように構成し得る。そうすると、アプリケーション処理エンジンは、異なるターゲットイベント処理システムがサポートする異なる実行（ランタイム）環境においてアプリケーションの１つ以上のジェネリック表現を実行させるように構成することができる。

【００５５】

本開示の実施形態に従って「共通アプリケーションランタイムモデル」を生成することにより、ターゲットエンジンにおけるその実行に先立ってターゲットエンジンの特定の物理実行（ランタイム）環境に適するようにアプリケーションコードをアプリケーションの開発者（たとえばユーザ）が再度書き込まなくても、異なる物理実行（ランタイム）環境においてアプリケーションのジェネリック表現を実行することができる。

【００５６】

上記技術はいくつかのやり方でいくつかのコンテキストにおいて実装することが可能である。以下では、イベント処理アプリケーションのデプロイ、処理、および実行に関する動作を開示される分散イベント処理システムが実行し得る手法のさらに他の詳細を記載している図１～図９を参照しながら、実装およびコンテキストのいくつかの例を提供する。

【００５７】

図１は、本開示の実施形態に係る、異なる実行環境でイベント処理アプリケーションを処理できる環境を提供する、一例としてのイベント処理システムアーキテクチャ１００の局面を示す。ある実施形態において、このアーキテクチャ（イベント処理システム）１００は、ネットワーク１０８を介して１つ以上のユーザデバイス１０２に通信可能に接続されたアプリケーション処理エンジン１１０を含む。

【００５８】

ネットワーク１０８は、ユーザデバイス１０２とアプリケーション処理エンジンとの間におけるデータの通信およびやりとりを容易にすることが可能である。ネットワーク１０８は、ＴＣＰ／ＩＰ、ＳＮＡ、ＩＰＸ、AppleTalk（登録商標）等を含むがこれらに限定されないさまざまな商用プロトコルのうちのいずれかを用いてデータ通信をサポートでき

る、当業者によく知られたいずれかのタイプのネットワークであればよい。ほんの一例として、ネットワーク１０８は、イーサネット（登録商標）ネットワークやトークンリングネットワークおよび／または同様のネットワーク等の、ローカルエリアネットワーク（ＬＡＮ）であってもよく、ワイドエリアネットワークであってもよく、仮想プライベートネットワーク（ＶＰＮ）を含むがこれに限定されない仮想ネットワークであってもよく、インターネット、イントラネット、エクストラネット、公衆交換電話網（ＰＳＴＮ）、赤外線ネットワーク、無線ネットワーク（たとえば、ＩＥＥＥ８０２．１Ｘプロトコルスイート、当該技術において周知のBluetooth（登録商標）、および／またはその他任意の無線プロトコル）、および／またはこれらおよび／またはその他のネットワークの任意の組み合わせであってもよい。

【００５９】

ユーザデバイス１０２は、汎用パーソナルコンピュータ（一例として、各種バージョンのMicrosoft Windows（登録商標）および／またはApple Macintosh（登録商標）のオペレーティングシステムを実行するパーソナルコンピュータおよび／またはラップトップコンピュータを含む）、携帯電話もしくはＰＤＡ（Microsoft Windows Mobile等のソフトウェアを実行し、インターネット、電子メール、ＳＭＳ、Blackberry（登録商標）もしくはその他の通信プロトコルに対応する）、各種商用ＵＮＩＸ（登録商標）もしくはＵＮＩＸ系オペレーティングシステム（各種ＧＮＵ／Ｌｉｎｕｘ（登録商標）オペレーティングシステムを含むがこれらに限定されない）を実行するワークステーションコンピュータ、または、その他任意のコンピューティングデバイスであってもよい。たとえば、ユーザデバイス１０２は、ネットワーク（たとえばネットワーク１０８）を介して通信可能な、シンクライアントコンピュータ、インターネット対応ゲームシステム、および／またはパーソナルメッセージングデバイス等の、その他任意の電子デバイスであってもよい。一例としてのシステム環境１００は１つのユーザデバイスとともに示されているが、その他の実施形態では任意の数のユーザおよび／またはクライアントコンピューティングデバイスをサポートすることが可能である。

【００６０】

特定の実施形態において、アプリケーション処理エンジン１１０は、異なるランタイム環境における実行のためにイベント処理アプリケーションを処理するように構成し得る。特定の例において、イベント処理アプリケーションは、ユーザデバイス１０２のユーザによって生成されてもよい。たとえば、ユーザ１０２は、クライアントアプリケーション１０４が提供するアプリケーション設計ユーザインターフェイス１０６を用いて、ユーザデバイス内にクライアントアプリケーション１０４（たとえばブラウザ）を用いるアプリケーション（たとえばイベント処理アプリケーション）を構築することができる。上述のように、イベント処理アプリケーションは、イベントソースからのデータの入力ストリームを処理するために使用される一組のルール（たとえば連続クエリの形態で表現される）を含み得る。イベントソースは、モニタリングデバイス、金融サービス企業、または自動車両等の各種データソースを含み得る。このデータを用いて、イベント処理アプリケーションは、パターンを特定してこのパターンに反応する、異常なイベントを探してその他のアプリケーションに警告する、または、急速に変化するデータに基づいて即時の処置を必要とするその他何らかの作業を実行することができる。

【００６１】

アプリケーション処理エンジン１１０は、１つ以上のコンピュータおよび／またはサーバを含み得る。これは、汎用コンピュータ、専用サーバコンピュータ（例として、ＰＣサーバ、ＵＮＩＸサーバ、ミッドレンジサーバ、メインフレームコンピュータ、ラックマウントサーバ等を含む）、サーバファーム、サーバクラスタ、または、その他任意の適切な構成および／または組み合わせであってもよい。アプリケーション処理エンジン１１０を構成するコンピューティングデバイスは、オペレーティングシステムまたは各種その他のサーバアプリケーションおよび／またはミッドティアアプリケーションのうちのいずれかを実行し得る。これは、ＨＴＴＰサーバ、ＦＴＰサーバ、ＣＧＩサーバ、Ｊａｖａサーバ

、データベースサーバ等を含む。データベースサーバの例は、Oracle、Microsoft、Sybase、ＩＢＭ等から市販されているものを含むが、これらに限定されない。

【００６２】

特定の実施形態において、アプリケーション処理エンジン１１０は、先に述べたようなアプリケーション（たとえばイベント処理アプリケーション）をユーザデバイス１０２から受信し、このアプリケーションにおける情報を処理することにより、このアプリケーションの「共通アプリケーションランタイムモデル」を生成するように構成し得る。上述のように、アプリケーションの「共通アプリケーションランタイムモデル」は、１つ以上の構成ブロックのセットとしてアプリケーションを表現したものであり、各構成ブロックはイベントビーンを表し、対応するメタデータがアプリケーションを記述する。アプリケーション処理エンジン１１０は、アプリケーションの「共通アプリケーションランタイムモデル」を、アプリケーションの１つ以上のジェネリック表現に変換するように構成し得る。特定の実施形態において、そうすると、アプリケーション処理エンジン１１０は、異なるターゲットイベント処理システムがサポートする異なる実行（ランタイム）環境においてアプリケーションのジェネリック表現を実行させるように構成することができる。

【００６３】

特定の実施形態において、アプリケーション処理エンジン１１０は、共通アプリケーションランタイムモデルジェネレータ１１２と、共通アプリケーションランタイムモデルオプティマイザ１１４と、ターゲットＤＡＧジェネレータ１１６とを含み得る。これらのコンポーネントは、ハードウェア、ファームウェア、ソフトウェア、またはこれらの組み合わせにおいて実現し得る。共通アプリケーションランタイムモデルジェネレータ１１２は、アプリケーションに対応付けられた情報に基づいてアプリケーションのための「共通アプリケーションランタイムモデル」を生成するように構成し得る。共通アプリケーションランタイムモデルオプティマイザ１１４は、「共通アプリケーションランタイムモデル」を最適化することにより、アプリケーションのための最適化された共通アプリケーションランタイムモデルを生成するように構成し得る。ターゲットＤＡＧジェネレータ１１６は、最適化された共通アプリケーションランタイムモデルを、ターゲットイベントストリーム処理エンジン（システム）のうちの１つによって実行できるアプリケーションの１つ以上のジェネリック表現に変換するように構成し得る。アプリケーション処理エンジン１１０のコンポーネント１１２、１１４、および１１６が実行する動作は、以下において図２との関連で詳細に説明する。

【００６４】

特定の実施形態において、ターゲットイベント処理エンジン１１８は、アプリケーション処理エンジン１１０から共通アプリケーションランタイムモデルを受信し、共通アプリケーションランタイムモデルにおける情報を、ターゲットイベントストリームエンジン１１８が提供するランタイム（実行）環境において実行可能なアプリケーション（すなわちターゲットイベント処理アプリケーション１２０）のプラットフォーム固有の実装に変換するように構成し得る。そうすると、ターゲットイベントストリーム処理エンジン１１８は、ストリームに新たなデータが現れるとターゲットイベント処理アプリケーション１２０を連続的に実行し結果の連続ストリームをユーザに提供するように構成し得る。ターゲットイベントストリーム処理エンジン１１８は、連続的に受信したデータストリーム（たとえばライブフィード）に対しターゲットイベント処理アプリケーション１２０において定められた１つ以上の動作（たとえばＣＱＬクエリ）を実行することにより、データストリームのリアルタイム処理を実行することができる。たとえば、ターゲットイベントストリーム処理エンジン１１８は、１つ以上の連続データストリームを受信し、データストリームに対するターゲットイベント処理アプリケーション１２０を登録し、ストリーム内に新たなデータが現れると、ターゲットイベント処理アプリケーション１２０において定められた１つ以上のクエリを連続的に実行することができる。この種の連続クエリは実行時間が長いので、ターゲットイベントストリーム処理エンジンは、結果の連続ストリームをユーザに提供することができる。ターゲットイベントストリーム処理エンジン１１８が実

行するその他の動作は、図３との関連で詳細に説明する。
【００６５】
特定の実施形態において、各ターゲットイベントストリーム処理エンジン１１８は、ターゲットイベント処理アプリケーションを実行するための特定の物理実行環境を表し得る。たとえば、第１のターゲットイベントストリーム処理エンジンは、第１の物理実行（ランタイム）環境においてターゲットイベント処理アプリケーションを実行するように構成された第１のイベントストリーミングプラットフォーム、第２の物理実行（ランタイム）環境においてアプリケーションを実行するように構成された第２のイベントストリーミングプラットフォーム、第３の物理実行（ランタイム）環境においてアプリケーションを実行するように構成された第３のイベントストリーミングプラットフォームなどを含み得る。第１、第２、および第３のイベントストリーミングプラットフォームは互いに異なり得る。たとえば、第１のイベントストリーミングプラットフォームは、Oracle（登録商標）が管理するオラクルイベントプロセッサ（Oracle Event Processor）（ＯＥＰ）システムを表し得る。第２のイベントストリーミングプラットフォームは、Spark（登録商標）システムが管理するSpark（登録商標）フレームワーク等の第１のタイプの分散イベント処理プラットフォームを表し得る。第３のイベントストリーミングプラットフォームは、Flink（登録商標）システムが管理するFlink（登録商標）フレームワーク等の第３のタイプの分散イベント処理プラットフォームを表し得る。
【００６６】
図２は、本開示の実施形態に係る、イベント処理アプリケーションのためのイベント処理ネットワーク（ＥＰＮ）２００を図示する。特定の例において、イベント処理アプリケーションは、コンポーネントのネットワークとして表すことができる。このようなコンポーネントのネットワークは通常、イベント処理ネットワーク（event processing network）（ＥＰＮ）２００と呼ばれる。ＥＰＮ２００は、イベント処理アプリケーションのコンポーネント間におけるイベントベースのインタラクションおよびイベント処理仕様を表現するための概念的モデルである。イベント処理アプリケーションのコンポーネントは、アダプタ、ストリーム、プロセッサ、ビジネスロジックプレーンオールドＪａｖａオブジェクト（Plain Old Java Object）（ＰＯＪＯ）、およびビーンを含み得る。ＥＰＮ２００における各コンポーネントは、イベントストリームを介して受信したデータの処理において、ある役割を果たす。上述のように、イベント処理ネットワーク（ＥＰＮ）２００は、これらの各種コンポーネント、これらのコンポーネントがどのようにしてともに接続されるか、アプリケーションが処理するイベントタイプ、アプリケーションが使用するイベントの選択のための連続クエリまたはロジック、アプリケーションにおいて定められたビジネスルールなどを記述する、情報を、含み得る。
【００６７】
特定の実施形態において、ユーザデバイス１０２のユーザは、ユーザデバイスにおいてクライアントアプリケーション（たとえば１０４）が提供するアプリケーション設計ユーザインターフェイス（たとえば１０６）を用いてイベント処理アプリケーションのためのＥＰＮ２００を生成することができる。その他の実施形態において、ユーザは、アプリケーション設計ユーザインターフェイスを介してアプリケーションを特定する情報を提供することができる。このような情報は、たとえば、アプリケーションにおいて定められた１つ以上の連続クエリ、アプリケーションのデプロイのタイプを指定するアプリケーションパラメータ、アプリケーションのランタイム構成（たとえば、使用するエグゼキュータ（executor）の数、パラレル処理パラメータ、メモリのサイズ、高可用性パラメータ）などを含み得る。クライアントアプリケーションは、この情報に基づいて、イベント処理アプリケーションのためにＥＰＮ２００を構築および／または生成することができる。
【００６８】
特定の実施形態において、図２に示されるように、イベント処理アプリケーションのためのＥＰＮ２００は、以下のコンポーネントタイプで構成することができる。
【００６９】

（１）入力および出力ストリームならびにリレーションソースおよびシンクに直接インターフェイスする１つ以上のアダプタ（２０２，２０４）。アダプタは、入力および出力ストリームプロトコルを理解するように構成され、イベントデータを、アプリケーションプロセッサによってクエリ可能な正規化された形態に変換する役割を担う。アダプタは、正規化されたイベントデータをチャネルまたは出力ストリームおよびリレーションシンクに転送し得る。イベントアダプタは、さまざまなデータソースおよびシンクについて定義され得る。図２に示す実施形態において、アダプタは、ストリームまたはリレーションソース１アダプタ２０２およびストリームまたはリレーションソース２アダプタ２０４を含む。

【００７０】

（２）イベント処理終点の役割を果たす１つ以上のチャネル（２０６，２０８，２１０）。とりわけ、チャネルは、イベント処理エージェントがイベントデータに対して作用することができるまでイベントデータを待ち行列に入れる役割を担う。

【００７１】

（２）１つ以上のアプリケーションプロセッサ（または、イベント処理エージェント）２１２は、正規化されたイベントデータをチャネルから消費し、クエリを使用してそれを処理して注目すべきイベントを選択し、選択された注目すべきイベントを出力チャネル２１０に転送（または、コピー）するように構成される。

【００７２】

（４）１つ以上のビーン２１４，２１６，および２１８は、出力チャネル２２０をリッスンするように構成され、出力チャネル２２０への新たなイベントの挿入によってトリガされる。いくつかの実施形態では、このユーザコードは、プレーンオールドＪａｖａオブジェクト（ＰＯＪＯ）である。ユーザアプリケーションは、ＪＭＳ、ウェブサービスおよびファイルライタ等の外部サービス一式を活用して、生成されたイベントを外部イベントシンクに転送することができる。

【００７３】

（５）イベントビーン２１４，２１６，および２１８は、出力チャネル２２０をリッスンするように登録されてもよく、出力チャネルへの新たなイベントの挿入によってトリガされてもよい。いくつかの実施形態では、このユーザコードは、ビーンがOracleのＣＥＰによって管理できるようにOracleのＣＥＰイベントビーンＡＰＩを使用し得る。

【００７４】

一実施形態では、イベントアダプタ（２０２，２０４）は、イベントデータを入力チャネル（２０６，２０８）に提供する。入力チャネル（２０６、２０８）は、入力チャネル（２０６，２０８）によって提供されるイベントに対して動作する１つ以上のＣＱＬクエリに関連付けられたＣＱＬプロセッサ（２１２）に接続される。ＣＱＬプロセッサ（２１２）は、クエリ結果が書き込まれる出力チャネル（２２０）に接続される。

【００７５】

いくつかの実施形態では、イベント処理アプリケーションのさまざまなコンポーネント、コンポーネントがどのようにしてともに接続されるか、および、アプリケーションが処理するイベントタイプを記述するアセンブリファイルを、イベント処理アプリケーションに対して提供することができる。イベントの選択のための連続クエリまたはロジックを指定するために別々の構成ファイルが提供されてもよい。特定の実施形態において、イベント処理アプリケーションにおける情報を、Spring（登録商標）ＸＭＬフレームワークを用いてアセンブルしてもよい。以下でより詳細に説明するように、この手法により、アプリケーションを、既存のSpring（登録商標）ビーン、および、依存性注入（dependency injection）メカニズムに基づくその他の軽量プログラミングフレームワークと、簡単に統合することが可能である。たとえば、イベントサーバが制御の反転（Inversion of Control）（ＩｏＣ）コンテナ全体を強化することによりＥＰＮのアセンブリ内のSpring（登録商標）ビーンをユーザがシームレスに使用できるよう、アセンブリファイルは、Spring（登録商標）フレームワークコンテキストＸＭＬ構成ファイルのカスタム拡張であってもよ

い。

【００７６】

図３は、本開示の実施形態に係る、アプリケーション処理エンジンのコンポーネントを示す簡略化されたブロック図３００である。特定の実施形態において、アプリケーション処理エンジン３１４は、ユーザデバイス３０２のユーザから、イベント処理アプリケーション（たとえば３０４）を特定する情報を受信し、この情報に基づいて共通ランタイムアプリケーションモデルを生成するように構成し得る。上述のように、イベント処理アプリケーション３０４は、ユーザデバイス３０２のユーザが、ユーザデバイスにおいて、クライアントアプリケーション（たとえば１０４）が提供するアプリケーション設計ユーザインターフェイス３０６を用いて生成することができる。

【００７７】

いくつかの例において、アプリケーションを特定する情報は、アプリケーションの各種コンポーネント（たとえば、アダプタ、プロセッサ、ストリーム、またはイベントビーン）を記述する情報を含み得る。この情報は、たとえば、構成情報、クエリ情報、および、その他の種類の情報を含み得る。構成情報は、たとえば、イベント処理アプリケーションの各種コンポーネント、コンポーネントがどのようにしてともに接続されるか、および、アプリケーションが処理するイベントタイプを記述する情報を、含み得る。たとえば、構成情報は、コンポーネントのネットワーク（すなわちＥＰＮ２００）としてイベント処理アプリケーションを記述する情報を含み得る。クエリ情報は、アプリケーションによるイベントの選択のための連続クエリまたはロジックを指定する情報を含み得る。その他の種類の情報は、プレーンオールドＪａｖａオブジェクト（ＰＯＪＯ）およびアプリケーションにおいて定められたビジネスルールを含み得る。

【００７８】

特定の例において、アプリケーションを特定する情報は、構成ファイルにおいて指定することができる。たとえば、イベント処理アプリケーション３０４のＥＰＮ２００における各コンポーネントは、対応付けられた構成ファイルを有することができる。その他の例において、アプリケーション３０４における情報は、アプリケーションにおけるすべてのコンポーネントについての情報を含む単一の構成ファイルで表すことができる。一実装例において、構成ファイルは、その構造が、共通ランタイムモデルによって定められる構成スキーマに基づく標準ＸＭＬスキーマを用いて定められる、通常のＸＭＬドキュメントとして表現することができる。

【００７９】

特定の例において、アプリケーションを特定する情報は、ＥＰＮ構成ファイル、クエリ構成ファイル、およびその他のファイル等の、各種構成ファイルを用いて、指定することができる。イベント処理アプリケーションのためのＥＰＮ構成ファイルおよびクエリ構成ファイルの一例を以下に示す。示されている例において、イベント処理アプリケーションは、イベントの連続ストリームを受信して処理するように構成されたオーダーイベント処理アプリケーション３０４であり、各イベントは、ある企業が販売する商品に対するオーダーを表す。オーダーイベントストリームにおける各オーダーは、商品に関する、オーダー識別子、オーダーステータス、およびオーダー量等の属性を含み得る。イベント処理アプリケーション３０４のためのＥＰＮ構成ファイル３０８の一例を以下に示す。ＥＰＮ構成ファイル３０８はサブエレメントのシーケンスを含み、各サブエレメントは、イベント処理アプリケーションにおけるイベント処理コンポーネントについての構成情報を含む。

【００８０】

【数１】

ＥＰＮコンフィギュレーションファイル

```
<wlevs:event-type-repository>
 <wlevs:event-type type-name=“OrderEvent">
 <wlevs:properties>
 <wlevs:property name=“orderId" type=“int" />
 <wlevs:property name=“status" type=“char" />
 <wlevs:property name=“amount" type=“int" />
 </wlevs:properties>
 </wlevs:event-type>
</wlevs:event-type-repository>
<wlevs:adapter id=“socketAdapter“ provider=“socket” />
<wlevs:channel id=“orders" event-type=“OrderEvent" >
 <wlevs:listener ref=“orderProcessor"/>
 <wlevs:source ref=" socketAdapter "/>
</wlevs:channel>

<wlevs:processor id=" orderProcessor " />

 <wlevs:channel id=“otutputChannel" event-type=" OrderEvent “>
 <wlevs:listener ref=“outputAdapter"/>
 <wlevs:source ref=" orderProcessor "/>
</wlevs:channel>
 <wlevs:adapter id=”outputAdapter” provider=”stdout” />
```

【００８１】
イベント処理アプリケーション３０４のための、一例としてのクエリ構成ファイル３１０を以下に示す。クエリ構成ファイル３１０は、イベント処理アプリケーションにおけるイベントの選択のための１つ以上の連続クエリまたはロジックを指定する。
【００８２】
【数２】

クエリコンフィギュレーションファイル

```
<processor>
 <name>orderProcessor</name>
 <rules>
 <query id="helloworldRule"> <![CDATA[
 select status, count(*) from orders group by status
 ]]>
 </query>
 </rules>
</processor>
```

【００８３】
特定の実施形態において、アプリケーション処理エンジン３１４は、ＥＰＮ構成ファイル３０８、クエリ構成ファイル３１０、およびその他のファイル３１２において指定された、構成情報、クエリ情報、およびその他の情報に基づいて、アプリケーション（たとえば３０４）のための「共通アプリケーションランタイムモデル」を生成するように構成し得る。次に、ターゲットＤＡＧジェネレータ３２０が、「共通アプリケーションランタイムモデル」を、異なるターゲットイベントストリーム処理エンジン２２６、２２８、および２３０がサポートする異なる物理実行（ランタイム）環境における実行のために、アプリケーションの１つ以上のジェネリック表現に変換することができる。「共通アプリケーションランタイムモデル」を生成することにより、ターゲットエンジンにおけるその実行に先立ってターゲットエンジンの特定の物理実行（ランタイム）環境に適するようにアプリケーションコードをアプリケーションの開発者（たとえばユーザ）が再度書き込まなくても、異なる物理実行（ランタイム）環境においてアプリケーションのジェネリック表現を実行することができる。共通アプリケーションランタイムモデルは、物理実行環境から独立している。プラットフォーム独立抽象化により、システムは、物理実行環境のためのＤＡＧおよびコードを容易に生成することができる。
【００８４】
特定の実施形態において、アプリケーション処理エンジン３１４が「共通アプリケーションランタイムモデル」を生成することは、対応付けられた構成情報を用いて１つ以上のイベントビーンのセットとしてアプリケーションを表すことを含み得る。アプリケーション処理エンジン３１４がイベントビーンのセットとして「共通アプリケーションランタイムモデル」を表すことができる手法のさらに他の詳細については以下で説明する。
【００８５】
特定の実施形態において、アプリケーション処理エンジン３１４は、共通アプリケーションランタイムモデルジェネレータ３１６と、共通アプリケーションランタイムモデルオプティマイザ３１８と、ターゲットＤＡＧジェネレータ３２０とを含む。共通アプリケーションランタイムモデルジェネレータ３１６は、ＥＰＮ構成ファイル３０８、クエリ構成ファイル３１０、およびその他のファイル３１２において指定されたアプリケーションを特定する情報に基づいて、アプリケーションのための「共通アプリケーションランタイムモデル」を生成するように構成される。特定の実施形態において、共通アプリケーションランタイムモデルジェネレータ３１６が「共通アプリケーションランタイムモデル」を生成することは、ＥＰＮ構成ファイル３０８、クエリ構成ファイル３１０、およびその他のファイル３１２を、Spring（登録商標）アプリケーションフレームワークを用いて実装されたＥＰＮローダを用いてメモリにロードすることを含む。その結果は、制御注入（control injection）のSpring（登録商標）の反転によって接続されたSpring（登録商標）ビーンのセットである。そうすると、共通アプリケーションランタイムモデルジェネレータ３１６は、ＪＡＸＢ（Java Architecture for XML Binding（ＸＭＬバインディングのためのＪａｖａアーキテクチャ））等のＸＭＬパーサを用いてＥＰＮ構成ファイル３０８をパースし、パースされた構成ファイルを、ＥＰＮネットワークにおける対応付けられた各ビーンにセットするように、構成される。特定の例において、パースされた各構成ブロックまたはファイルは、このブロックまたはファイルがイベントビーンを発見し構成ブロックまたはファイルをそのイベントビーンにセットすることができるよう、識別子を有するであろう。よって、ある実施形態において、アプリケーションのための「共通アプリケーションランタイムモデル」を生成することは、１つ以上の構成ブロックのセットとしてアプリケーションを表すことを含み、各構成ブロックは、メタデータが対応付けられているSpring（登録商標）イベントビーンを表す。「共通アプリケーションランタイムモデル」の表現の一例を図４に示す。特定の例において、「共通アプリケーションランタイムモデル」は、その中に何ら処理ロジックを伴うことなく、イベント処理アプリケーションのコンポーネントに関するメタデータを維持する。
【００８６】

図４は、本開示の実施形態に係る、共通アプリケーションモデルジェネレータ３１６によって生成された「共通アプリケーションランタイムモデル」４００を表現したものの一例を示す。ある実施形態において、「共通アプリケーションランタイムモデル」４００は、（たとえばＥＰＮ構成ファイル３０８、クエリ構成ファイル３１０、およびその他のファイル３１２からの）アプリケーションを特定する情報に基づいて生成することができ、１つ以上の構成ブロックのセットとして表すことができ、各構成ブロックは、メタデータが対応付けられているSpring（登録商標）イベントビーンを表す。ある実施形態において、共通ランタイムアプリケーションモデル４００における構成ブロック４０２，４０４、４０６、４０８、および４１０は以下の情報を含み得る。

【００８７】

【数３】

ソケットインバウンド　402

host = “localhost”

port = 9999

チャネル１　404

tableName = “orders”

eventType = “OrderEvent”

relation = false

ＣＱＬプロセッサ　406

rule= “select status, count(*) from orders group by status”

チャネル２　408

eventType = “OrderEvent”

標準アウトバウンド　410

<no additional info>

【００８８】

たとえば、構成ブロック４０２において、ソケットインバウンドはＥＰＮにおける「socketAdapter」アダプタを表し、構成ブロック４０４において、チャネル１はＥＰＮにおける「orders」チャネルを表し、構成ブロック４０６において、ＣＱＬプロセッサはＥＰＮにおける「orderProcessor」プロセッサを表し、構成ブロック４０８において、チャネル２はＥＰＮにおける「outputChannel」チャネルを表し、構成ブロック４１０において、ソケットアウトバウンド４１０はＥＰＮにおける「outputAdapter」アダプタを表す。

【００８９】

上述のように、イベント処理アプリケーションを「共通アプリケーションランタイムモデル」４００として表すことにより、ターゲットエンジンにおけるその実行に先立ってターゲットエンジンの特定の物理実行（ランタイム）環境に適するようにアプリケーションコードをユーザが再度書き込まなくても、異なるターゲットイベントストリーム処理エンジン（たとえば３２６、３２８、または３３０）においてアプリケーションを実行することができる。

【００９０】

図３の説明に戻ると、特定の実施形態において、共通アプリケーションランタイムモデ

ルジェネレータ３１６が生成した共通アプリケーションランタイムモデルはさらに、共通アプリケーションランタイムモデルオプティマイザ３１８により、最適化することができる。共通アプリケーションランタイムモデル（たとえば４００）の最適化は、たとえば、ある構成ブロック内の複数の段を組み合わせて１つの段にすること（たとえば複数の連続クエリを組み合わせて１つの最適化された連続クエリにすること）または、１つの段を並列処理のために複数の段に分割すること（たとえば、連続クエリを、マッピングを実行し動作を減じることができる複数の連続クエリに分割すること）を含み得る。たとえば、再度のパーティショニングなしの連続クエリを組み合わせて、１つのＣＱＬプロセスを有する１つのＣＱＬ段にすることができる。たとえば、「select * from orders where orderStatus='open'」および「select count(*) from orders group by orderId」という２つの段を組み合わせて、「select count(*) from orders group by orderId where orderStatus=''open''」という１つのＣＱＬ段にすることができる。オプティマイザは、スケーラビリティを最大にするために１つの段を複数の段に分割することもできる。たとえば、「select count(*) from orders」という十分にステートフルなクエリ段を、パーティショニングを用いて、「select count(*) from order group by orderId」および「select sum(*) from counts」という２つの段に分割することができる。これにより、パーティショニングを用いてイベントのグローバルカウントを部分カウントで処理することができ、その後、部分カウントを合計してグローバルカウントにすることができる。

【００９１】

特定の実施形態において、ターゲットＤＡＧジェネレータ３２０は、最適化された共通アプリケーションランタイムモデルを、ターゲットイベントストリーム処理エンジンのうちの１つ（３２６、３２８、または３３０）が実行できるアプリケーションの１つ以上のジェネリック表現に変換するように構成し得る。たとえば、ターゲットＤＡＧジェネレータ３２０は、アプリケーションが実行されるであろうターゲットイベントストリーム処理エンジンに応じて、最適化された共通アプリケーションランタイムモデルを、ランタイムＤＡＧ３２２またはアプリケーションのターゲット表現３２４に変換するように、構成し得る。たとえば、ターゲットイベントストリーム処理エンジン（たとえば３２６）が、Oracle（登録商標）が管理するオラクルイベントプロセッサ（Oracle Event Processor）（ＯＥＰ）である場合、ターゲットＤＡＧジェネレータ３２０は、共通アプリケーションランタイムモデル（または最適化された共通アプリケーションランタイムモデル）における構成ブロックを、ターゲットイベントストリーム処理エンジン３２６が実行するアプリケーションのターゲット表現３２４に変換するように、構成し得る。特定の実施形態において、この変換は、ターゲットＤＡＧジェネレータ３２０が、実際の処理ロジックを有する適切なビーンを用いて共通アプリケーションランタイムモデルにおけるオブジェクトを複製することを含み得る。たとえば、共通アプリケーションランタイムモデルにおけるＣＱＬプロセッサビーンのメタデータを、ＣＱＬエンジンを実行するイベント処理コンポーネント（たとえばＣＱＬプロセッサ）の新たなインスタンスにコピーして、所与のクエリ（たとえば共通ランタイムアプリケーションモデルにおけるＣＱＬプロセッサからコピーしたもの）を用いて入力イベントを処理してもよい。

【００９２】

たとえば、ターゲットイベントストリーム処理エンジン（たとえば３２８）が、Spark（登録商標）分散システムが管理する分散イベント処理プラットフォームである場合、ターゲットＤＡＧジェネレータ３２０を、共通アプリケーションランタイムモデル（または最適化された共通アプリケーションランタイムモデル）におけるオブジェクト（構成ブロック）を、アプリケーションを表すコンポーネントのランタイムＤＡＧ３２２に変換するように、構成し得る。ターゲットＤＡＧジェネレータが生成するコンポーネントのランタイムＤＡＧ３２２の一例を、図５に示す。次に、コンポーネントのランタイムＤＡＧ３２２を、ターゲットイベントストリーム処理エンジン３２８が、ターゲットアプリケーション（Spark（登録商標）ＣＱＬアプリケーション）に変換する。ターゲットイベントスト

リーム処理エンジン３２８が、オーダーイベント処理アプリケーションのためのオーダーステータスによってグループ分けされたオーダーの数を計算するために生成するターゲットアプリケーションの一例を以下に示す。

【００９３】

【数４】

ターゲットアプリケーションの例

セットアップ

```
1. val sparkConf = new SparkConf
2. val sc = new SparkContext(sparkConf)
3. val cc = new CQLContext(sc, Seconds(1))
```

イベントタイプ、ストリーム登録

```
4. val orderEvent = EventType("orders", Attribute("orderId",INT),
   Attribute("status",CHAR), Attribute("amount", INT) )
5. cc.registerEventType(orderEvent)
6. cc.registerStream(orderEvent)
```

データをロード

```
7. val lines = cc.socketTextStream("localhost", 9999)
8. val rows = lines.map(_.split(","))
9. val kv_orders = rows.map(r => (r(1), EventUtil.createTupleValue(orderEvent,
   r(0).toInt, r(1), r(2).toInt) )
```

データをパーティショニング

```
10. val orders = rorders.transform( rdd => rdd.partitionBy(new
    OrderPartitioner(numPartitions)).map { case(k,v) => v }
)
```

ＣＱＬ処理

```
11. val result = cc.cql(orders, "select status, count(*) from orders group by status")
```

出力

```
12. val sresult = result.map(x => x.mkString(","))
13. sresult.print
```

【００９４】

図５は、本開示の実施形態に係る、ターゲットＤＡＧジェネレータが生成した、一例としての、コンポーネントのランタイム有向非巡回グラフ（ＤＡＧ）５００を示す。特定の実施形態において、上述のように、ターゲットＤＡＧジェネレータ３２０を、ターゲット

アプリケーションの実行（ランタイム）環境が分散イベント処理システム（たとえばSpark（登録商標）分散イベント処理システム）であるときに、共通アプリケーションランタイムモデル（たとえば４００）におけるオブジェクト（４０２、４０４、４０６、４０８、および４１０）を、アプリケーションを表すコンポーネントのランタイムＤＡＧ５００に変換するように、構成することができる。

【００９５】

ある実施形態において、コンポーネントのランタイムＤＡＧは、以下のコンポーネント、すなわち、SocketText５０２、Map-1　５０４、Map-2　５０６、PartitionBy５０８、ＣＱＬ５１０、Map-3　５１２、およびPrint５１４を含む。SocketInbout４０２（図４に示す）は、SocketText５０２、Map-1　５０４、およびMap５０６に変換される。SocketText５０２コンポーネントは、上記ターゲットアプリケーションの例におけるライン７を含み、これは、ソケットからストリングをロードする。Map-1　５０４コンポーネントは、
ターゲットアプリケーションの例におけるライン８を含み、これは、ストリングを、カンマで分離された値に変換する。Map-2　５０６コンポーネントは、ターゲットアプリケーションの例におけるライン９を含み、これは、カンマで分離された値をタプルに変換する。ＣＱＬプロセッサ４０６およびChannel-1　４０４は、PartitionBy５０８およびＣＱＬ５１０に変換される。PartitionBy５０８コンポーネントは、ターゲットアプリケーションの例におけるライン１０を含み、これは、ＣＱＬにおけるgroup by基準に基づいてパーティショニングを生成する。ＣＱＬ５１０コンポーネントは、ターゲットアプリケーションの例におけるライン１１を含み、これは、主要なＣＱＬ処理段である。Channel-2　４０８およびStdoutOutbound４１０は、Map-3　５１２およびPrint５１に変換される。Map-3　５１２コンポーネントは、ターゲットアプリケーションの例におけるライン１２を含
み、これは、タプルを、カンマで分離されたストリングに変換し、Print５１４コンポーネントは、ターゲットアプリケーションの例におけるライン１３を含み、これは、出力ストリングをstdoutコンソールにプリントする。

【００９６】

図６は、本開示の実施形態を組み込むことが可能なイベント処理システム６００の簡略化されたハイレベル図である。ある実施形態において、イベント処理システム６００は、Oracle（登録商標）が管理するオラクルイベントプロセッサ（ＯＥＰ）システムを表していてもよい。イベント処理システム６００は、１つ以上のイベントソース（６０４、６０６、６０８）と、イベントストリームを処理するための環境を提供するように構成されたイベント処理サービス（ＥＰＳ）６０２（ＣＱサービス６０２とも呼ばれる）と、１つ以上のイベントシンク（６１０、６１２）とを含み得る。イベントソースは、ＥＰＳ６０２が受信するイベントストリームを生成する。ＥＰＳ６０２は、１つ以上のイベントソースから１つ以上のイベントストリームを受信することができる。

【００９７】

たとえば、図６に示すように、ＥＰＳ６０２は、イベントソース６０４から第１の入力イベントストリーム６１４を受信し、イベントソース６０６から第２の入力イベントストリーム６１６を受信し、イベントソース６０８から第３のイベントストリーム６１８を受信する。１つ以上のイベント処理アプリケーション（６１４、６１６、および６１８）を、ＥＰＳ６０２上にデプロイしＥＰＳ６０２によって実行することができる。ＥＰＳ６０２が実行するイベント処理アプリケーションは、１つ以上の入力イベントストリームをリッスンし、入力イベントストリームから１つ以上のイベントを注目すべきイベントとして選択する処理ロジックに基づいて、１つ以上のイベントストリームを介して受信したイベントを処理するように、構成し得る。次に、注目すべきイベントを、１つ以上の出力イベントストリームの形態で１つ以上のイベントシンク（６１０、６１２）に送信することができる。たとえば、図６において、ＥＰＳ６０２は、第１の出力イベントストリーム６２０をイベントシンク６１０に出力し、第２の出力イベントストリーム６２２をイベントシ

ンク６１２に出力する。特定の実施形態において、イベントソース、イベント処理アプリケーション、およびイベントシンクは、これらのコンポーネントのうちのいずれかを、その他のコンポーネントに変更を加えることなく追加または削除できるよう、互いに分離されている。

【００９８】

一実施形態において、ＥＰＳ６０２は、共有されるサービスとともに、Equinox OSGiに基づくもののような、軽量Ｊａｖａアプリケーションコンテナを含む、Ｊａｖａサーバとして実装してもよい。いくつかの実施形態において、ＥＰＳ６０２は、たとえば、JRockit Real Timeを使用することによりイベントを処理するために超高スループットおよびマイクロ秒レイテンシをサポートし得る。ＥＰＳ６０２は、イベント処理アプリケーションを開発するためのツール（たとえばOracle CEP VisualizerおよびOracle CEP IDE）を含む開発プラットフォーム（たとえば完全なリアルタイム・エンドツーエンドＪａｖａイベント駆動型アーキテクチャ（ＥＤＡ）開発プラットフォーム）を提供することもできる。

【００９９】

イベント処理アプリケーションは、１つ以上の入力イベントストリームをリッスンし、１つ以上の入力イベントストリームから１つ以上の注目すべきイベントを選択するためにロジック（例えばクエリ）を実行し、選択した注目すべきイベントを１つ以上の出力イベントストリームを介して１つ以上のイベントソースに出力するように、構成される。図６は、このようなあるイベント処理アプリケーション６１４に対するドリルダウンを提供する。図６に示すように、イベント処理アプリケーション６１４は、入力イベントストリーム６１８をリッスンし、入力イベント６１８から１つ以上の注目すべきイベントを選択するためのロジックを含む連続クエリ６３０を実行し、出力イベントストリーム６２２を介してイベントシンク６１２に選択した注目すべきイベントを出力するように、構成される。イベントソースの例は、アダプタ（たとえばＪＭＳ、ＨＴＴＰ、およびファイル）、チャネル、プロセッサ、テーブル、キャッシュなどを含むが、これらに限定されない。イベントシンクの例は、アダプタ（ＪＭＳ、ＨＴＴＰ、およびファイル）、チャネル、プロセッサ、キャッシュなどを含むが、これらに限定されない。

【０１００】

図６のイベント処理アプリケーション６１４は、１つの入力ストリームをリッスンし選択したイベントを１つの出力ストリームを介して出力するものとして示されているが、これは限定を意図したものではない。代替の実施形態において、イベント処理アプリケーションは、１つ以上のイベントソースから受信した複数の入力ストリームをリッスンし、モニタリングされたストリームからイベントを選択し、選択したイベントを、１つ以上の出力イベントストリームを介して１つ以上のイベントシンクに出力するように、構成することができる。同一のクエリを、２つ以上のイベントシンクと、異なる種類のイベントシンクとに、対応付けることができる。

【０１０１】

その無制限という性質のため、イベントストリームを介して受信するデータの量は大抵、非常に多い。結果として、クエリするという目的のためにすべてのデータを格納またはアーカイブすることは通常は非実用的であり望ましくない。イベントストリームを処理するには、受信したイベントデータをすべて格納する必要を伴うことなく、ＥＰＳ６０２がイベントを受信するとイベントをリアルタイムで処理することが必要である。このように、ＥＰＳ６０２は、受信したイベントすべてを格納する必要を伴うことなく、ＥＰＳ６０２がイベントを受信するとイベントの処理を実行できるようにする特別なクエリ機構を提供する。

【０１０２】

イベント駆動型アプリケーションはルール駆動型であり、これらのルールは、入力ストリームを処理するのに用いられる連続クエリの形態で表現することができる。連続クエリは、受信したイベントに対して実行すべき処理を特定する命令（たとえばロジック）を含み得る。これは、注目すべきイベントとしてどのようなイベントを選択しクエリ処理の結

果として出力すべきかを含む。連続クエリは、データストアにパーシストされイベントの入力ストリームを処理しイベントの出力ストリームを生成するために使用されてもよい。連続クエリは一般的に、フィルタリングおよびアグリゲーション機能を実行することにより、入力イベントストリームから注目すべきイベントを発見して抽出する。その結果、出力イベントストリームにおけるアウトバウンドイベントの数は通常、イベントの選択元である入力イベントストリームにおけるイベントの数よりも遙かに少ない。

【０１０３】

有限データセットに対して一度実行されるＳＱＬクエリとは異なり、特定のイベントストリームに対しＥＰＳ６０２にアプリケーションによって登録された連続クエリは、そのイベントストリームにイベントを受け入れるたびに実行することができる。連続クエリ実行の一部として、ＥＰＳ６０２は、連続クエリによって指定された命令に基づいて受信イベントを評価することにより、１つ以上のイベントを注目すべきイベントとして選択し連続クエリの実行結果として出力するするか否かを判断する。

【０１０４】

連続クエリは異なる言語を用いてプログラミングしてもよい。特定の実施形態において、連続クエリを、Oracle社によって提供されOracleの複合イベント処理（ＣＥＰ）製品によって使用されるＣＱＬを用いて構成してもよい。OracleのＣＱＬは、イベントストリームに対して実行できるクエリ（ＣＱＬクエリと呼ぶ）をプログラミングするのに使用することができる宣言型言語である。特定の実施形態において、ＣＱＬは、ストリーミングイベントデータの処理をサポートする構成が追加されたＳＱＬに基づく。

【０１０５】

図６に示されるシステム６００は、図６に示されているコンポーネント以外のコンポーネントを有し得ることが理解されるべきである。さらに、図６に示される実施形態は、本開示の実施形態を組み込むことができるシステムの一例に過ぎない。いくつかの他の実施形態では、システム６００は、図６に示されるコンポーネントよりも多くのコンポーネントもしくは少ないコンポーネントを有していてもよく、２つ以上のコンポーネントを組み合わせてもよく、またはコンポーネントの異なる構成もしくは配置を有していてもよい。システム６００は、サービスプロバイダコンピュータ、パーソナルコンピュータ、ポータブルデバイス（たとえば、携帯電話もしくはデバイス）、ワークステーション、ネットワークコンピュータ、メインフレーム、キオスク、サーバ、またはその他のデータ処理システムを含むさまざまなタイプのシステムであってもよい。

【０１０６】

図７は、本開示の実施形態に係る、分散イベント処理システム７１０のコンポーネントを示す簡略化されたブロック図７００である。図７に示す実施形態は、本開示の実施形態を組み込むことができる分散イベント処理システムの一例である。その他いくつかの実施形態において、システム７１０は、図７に示すコンポーネントよりも多いコンポーネントもしくは少ないコンポーネントを有していてもよく、２つ以上のコンポーネントを組み合わせてもよく、または、コンポーネントの異なる構成もしくは配置を有していてもよい。システム７１０は、モバイル、デスクトップ、シンクライアント、および／またはクラウドコンピューティングデバイス、サーバ、またはその他任意のデータ処理システム等であるがこれらに限定されない、いずれかの種類のコンピューティングデバイスであればよい。

【０１０７】

いくつかの例において、分散イベント処理システム７１０は、ソフトウェアリソース、ハードウェアリソース、ネットワーキングリソース、およびその他のリソースの、予め組み込まれ最適化された組み合わせで構成することができる。ハードウェアリソースは、サーバ、データストレージデバイス、サーバ、プリンタ等を含み得るが、これらに限定されない。ソフトウェアリソースは、コンピューティングプログラム、アプリケーション（たとえばクラウドベースのアプリケーション、企業アプリケーション、またはその他任意のアプリケーション）、コンピュータプログラムプロダクト（たとえばソフトウェア）、サービス（たとえばクラウドベースのサービス）等を含み得るが、これらに限定されない。

データリソースは、ファイル（たとえばネットワーク化されたファイルまたはディレクトリ情報）、データベース等の、アクセス可能な任意のデータオブジェクトを含み得るが、これらに限定されない。

【０１０８】

特定の実施形態において、分散イベント処理システム７１０は、受信機７０４とコンピューティングノードのクラスタ７０８とを含み得る。受信機７０４は、連続入力イベントストリーム７０２を受信し、このイベントストリームを、分散イベント処理システム７１０におけるコンピューティングノードのクラスタ７０８による後続の処理のために特定期間（たとえばＸ秒）１つ以上のイベントのバッチ７０６に、離散化（discretize）（分割）するように、構成し得る。イベントの各バッチを本明細書ではＤストリーム（Dstream）と呼ぶ。いくつかの例において、各Ｄストリームは内部では受信機７０４によってレジリエント分散データセット（Resilient Distributed Dataset）（ＲＤＤ）として表される。これは、特定期間中に（たとえばあるイベントバッチにおいて）インジェストされたデータ（イベント）の入力ストリームすべてのスナップショットである。よって、いくつかの実施形態において、入力データストリーム７０２は、Ｄストリームのシーケンスとして表され、各Ｄストリームは内部ではＲＤＤとして表され、各ＲＤＤは、特定のバッチ間隔中に受信したイベント（タプル）を含む。特定の例において、各ＲＤＤは、不変のパーティションされたエレメントのコレクションを表し、これは、キャッシュメモリに格納し分散イベント処理システムにおいて並列に実行することができる。

【０１０９】

特定の実施形態において、コンピューティングノードのクラスタ７０４は、コンピューティングノードのクラスタにわたり各ＲＤＤに含まれるデータをパーティションし、このデータに対し、アプリケーションにおいて定められた一組のクエリに対して並列に動作を実行し、分散イベント処理システムのユーザに処理結果を与えるように、構成することができる。よって、コンピューティングノードのクラスタ７０８は、ＲＤＤにおけるイベントデータの処理を、コンピューティングノードのクラスタ７０８にわたって分散させ、イベントデータに対するアプリケーションの実行に関連する結果を素早くリアルタイムでユーザに提供するように、構成することができる。ある実施形態において、分散イベント処理システム７１０は、Apache（登録商標）Spark Streamingフレームワークを用いて構成することにより、データの連続ストリームの分散リアルタイム処理およびイベント処理アプリケーションのデプロイを実行するように、構成することができる。

【０１１０】

図８は、本開示の一実施形態に係る、アプリケーションの共通アプリケーションランタイムモデルを生成するためにイベント処理アプリケーションを処理するための一組の動作を説明する、プロセス８００のフロー図の一例である。特定の実施形態において、プロセス８００は、図３に記載のアプリケーション処理エンジン（３１４）における１つ以上のコンポーネント（たとえば３１６、３１８、および３２０）によって実行されてもよい。プロセス８００は、８０２においてアプリケーションを特定する情報を受信することで始まる。この情報は、たとえば、構成情報、クエリ情報、およびその他の種類の情報等の、アプリケーションの各種コンポーネント（たとえばアダプタ、プロセッサ、ストリーム、またはイベントビーン）を記述する情報を含み得る。上述のように、この情報は、図３に示される構成ファイル（３０８、３１０、および３１２）において表現することができる。

【０１１１】

８０４において、プロセスは、アプリケーションを特定する情報に基づいてアプリケーションの「共通アプリケーションランタイムモデル」を生成することを含む。ある実施形態において、アプリケーションのための「共通アプリケーションランタイムモデル」を生成することは、アプリケーションを、１つ以上の構成ブロックのセットとして表すことを含み得る。ここで、各構成ブロックは、対応するメタデータを有するイベントビーンを表す。構成ブロックは、図４を参照しながら説明したように、インバウンドソケットイベントビーン、アウトバウンドソケットイベントビーン、連続クエリ言語（ＣＱＬ）プロセッ

サイベントビーン、または、図４に関連して説明した１つ以上のチャネルイベントビーンを含み得る。

【０１１２】

８０６において、このプロセスは、アプリケーションの「共通アプリケーションランタイムモデル」を、このアプリケーションの第１のジェネリック表現に変換することを含む。アプリケーションの第１のジェネリック表現は、複数のターゲットイベント処理システムのうちの第１のターゲットイベント処理システムにおいて実行されるように構成し得る。一例において、アプリケーションの「共通アプリケーションランタイムモデル」をアプリケーションの第１のジェネリック表現に変換することは、共通アプリケーションランタイムモデルにおける構成ブロックを、アプリケーションのコンポーネントのランタイムＤＡＧに変換することを含み得る。アプリケーションのために生成されるコンポーネントのランタイムＤＡＧの一例を図５に示す。

【０１１３】

いくつかの実施形態では、８０８において、このプロセスは、アプリケーションの第１のジェネリック表現を、第１のターゲットイベント処理システムによる実行のために第１のターゲットイベント処理システムに送信することを含む。一例において、第１のターゲットイベント処理システムは、分散イベント処理システムである。

【０１１４】

図９は、本開示の別の実施形態に係る、アプリケーションの共通アプリケーションランタイムモデルを生成するためにイベント処理アプリケーションを処理するための一組の動作を説明する、プロセス９００のフロー図の一例である。特定の実施形態において、プロセス９００は、図３に記載のアプリケーション処理エンジン（３１４）における１つ以上のコンポーネント（たとえば３１６、３１８、および３２０）によって実行されてもよい。プロセス９００は、９０２においてアプリケーションを特定する情報を受信することで始まる。上述のように、この情報は、アプリケーションの各種コンポーネントを記述する、構成情報、クエリ情報、およびその他の種類の情報を含み得る。

【０１１５】

９０４において、このプロセスは、アプリケーションを特定する情報に基づいてアプリケーションの「共通アプリケーションランタイムモデル」を生成することを含む。ある実施形態において、アプリケーションのための「共通アプリケーションランタイムモデル」を生成することは、アプリケーションを、１つ以上の構成ブロックのセットとして表すことを含み得る。ここで、各構成ブロックは、対応するメタデータを有するイベントビーンを表す。構成ブロックは、図４を参照しながら説明したように、インバウンドソケットイベントビーン、アウトバウンドソケットイベントビーン、連続クエリ言語（ＣＱＬ）プロセッサイベントビーン、または、図４に関連して説明した１つ以上のチャネルイベントビーンを含み得る。

【０１１６】

９０６において、このプロセスは、アプリケーションの「共通アプリケーションランタイムモデル」を、このアプリケーションの第２のジェネリック表現に変換することを含む。アプリケーションの第２のジェネリック表現は、複数のターゲットイベント処理システムのうちの第２のターゲットイベント処理システムにおいて実行されるように構成し得る。いくつかの例において、アプリケーションの「共通アプリケーションランタイムモデル」をアプリケーションの第２のジェネリック表現に変換することは、共通アプリケーションランタイムモデルにおける構成ブロックを、アプリケーションのターゲット表現に変換することを含み得る。

【０１１７】

いくつかの実施形態では、９０８において、プロセスは、アプリケーションの第２のジェネリック表現を、第２のターゲットイベント処理システムによる実行のために、第２のターゲットイベント処理システムに送信することを含む。一例において、第２のターゲットイベント処理システムは、Oracle（登録商標）イベントプロセッサ（ＯＥＰ）システム

である。

【０１１８】

イベントデータのシリアライズおよびデシリアライズ

特定の実施形態において、開示する分散イベント処理システムは、連続イベントストリームを介して受信したイベントデータのシリアライズおよびデシリアライズを実行するように構成し得る。イベントデータのシリアライズおよびデシリアライズにより、メモリにおける複合データオブジェクトを、分散イベント処理システムのコンピューティングノードに転送することができるビットのシーケンスに変換することができる。イベントデータのシリアライズおよびデシリアライズにより、分散イベント処理システムによるイベントデータの処理の前に、分散イベント処理システムにおける処理ノードによってデータを効率的に格納し表現することができる。加えて、イベントデータのシリアライズおよびデシリアライズにより、分散イベント処理システムにおける処理ノード間の入力イベントおよび出力イベントのやり取りにおけるレイテンシを低減し、分散イベント処理システムの全体的な性能を改善する。

【０１１９】

上記技術はいくつかのやり方でいくつかのコンテキストにおいて実装することが可能である。以下では、イベント処理アプリケーションのデプロイ、処理、および実行に関連する動作を開示する分散イベント処理システムが実行し得る手法のさらに他の詳細を記述する図１～図１０を参照しながら、いくつかの実装およびコンテキストの例を提供する。

【０１２０】

図１０は、本開示の実施形態に係る、分散イベント処理システムのコンポーネントを示す簡略化されたブロック図１０００である。分散イベント処理システム１０００は、図１に記載の分散イベント処理システム１１０と同一または同様であってもよい。図１０に示す実施形態は、本開示の実施形態を組み込むことができる分散イベント処理システムの一例である。その他の実施形態において、分散イベント処理エンジンは、図１０に示すコンポーネントよりも多いコンポーネントもしくは少ないコンポーネントを有していてもよく、２つ以上のコンポーネントを組み合わせてもよく、または、コンポーネントの異なる構成もしくは配置を有していてもよい。これらのコンポーネントは、ハードウェア、ファームウェア、ソフトウェア、またはこれらの組み合わせにおいて実現し得る。いくつかの実施形態において、ソフトウェアは、メモリ（たとえば非一時的なコンピュータ可読媒体）に、メモリデバイスに、または、その他何らかの物理メモリに格納されてもよく、１つ以上の処理ユニット（たとえば１つ以上のプロセッサ、１つ以上のプロセッサコア、１つ以上のＧＰＵなど）によって実行されてもよい。図１０に示す実施形態はしたがって、ある実施形態のシステムを実現するための分散イベント処理エンジンの一例であって、限定を意図したものではない。

【０１２１】

特定の実施形態において、分散イベント処理システム１００２は、受信機１００４と、アプリケーションデプロイモジュール１００８と、コンピューティングノードのクラスタ１０１２とを含み得る。受信機１００６は、図２に記載のように（たとえばイベントソース２０４、２０６、または２０８から）データの入力ストリーム１００４を受信することが可能であってもよく、入力データストリームを、本明細書ではＤストリームと呼ぶイベントの１つ以上のバッチ１０１０に分割することができる。上述のように、各Ｄストリーム（すなわちイベントバッチ）は、特定期間中にインジェストされたデータ（イベント）の入力ストリームすべてを含み、内部では受信機１００６によってＲＤＤオブジェクトとして表すことができる。これは、分散イベント処理システム１００２におけるコンピューティングノードのクラスタ１０１２において並列に実行できるエレメントの不変のパーティションされたコレクションである。

【０１２２】

アプリケーションデプロイモジュール１００６は、コンピューティングノードのクラスタ１０１２におけるコンピューティングノードによる処理および実行のためにアプリケー

ション（たとえばイベント処理アプリケーション）をデプロイするように構成し得る。本明細書に記載のアプリケーションは、分散イベント処理システムのコンピュータプログラム（たとえばユーザが構築したもの）を言う場合がある。たとえば、アプリケーションは、一組の入力テキスト内の特定のワードに対するリファレンスの量をカウントするワードカウントアプリケーションを含み得る。このようなアプリケーションは、たとえば、一組のテキストを読み取り、各テキスト内に各ワードが現れる回数をカウントする１つ以上の連続クエリを用いて構築することができる。入力テキストは、たとえば、Facebook（登録商標）またはTwitter（登録商標）等のオンラインアプリケーションからのストリームにおいて受信したショートメッセージを含み得る。上述のように、連続クエリはＣＱＬ言語を用いて構成することが可能である。たとえば、ワードカウントストリーミングアプリケーションにおいて実行すべきワードカウントタスク／動作を指定するために、ユーザは、SELECT count FROM location GROUP BY wordといった形態を取ることができるＣＱＬクエリを書き込むことができる。このようなクエリは、指定された場所からすべてのセンテンスを集め、これらのセンテンスから得た固有のワードを異なるグループに分けてから、各グループ内のワードの量をカウントすることができる。

【０１２３】

特定の実施形態において、アプリケーションデプロイモジュール１００８は、分散イベント処理システムのユーザからアプリケーションを特定する情報を受信するように構成し得る。たとえば、アプリケーションデプロイモジュール１００８は、アプリケーションデプロイモジュール１００８においてアプリケーション設計ユーザインターフェイスを介してアプリケーションを特定する情報を受信するように構成し得る。アプリケーションを特定する情報は、アプリケーションにおいて定められた１つ以上の連続クエリのセットを含み得る。アプリケーションを特定する情報はまた、アプリケーションに対応付けられたアプリケーションパラメータを含み得る。アプリケーションパラメータは、たとえば、ノードのクラスタ１０１２におけるアプリケーションのデプロイのタイプ（たとえば「クラスタモード」）を指定するデプロイタイプパラメータを含み得る。その他のアプリケーションパラメータは、アプリケーションのランタイム構成に関するパラメータ（たとえば、使用するエグゼキュータの数、パラレル処理パラメータ、メモリのサイズ、高可用性パラメータなど）を含み得る。

【０１２４】

特定の実施形態において、アプリケーションデプロイモジュール１００８は、アプリケーションに関連する情報を受信すると、クラスタにおけるコンピューティングノード上にアプリケーションをデプロイするためにコンピューティングノードのクラスタ１０１２に対して命令を送信するように構成し得る。特定の例において、コンピューティングノードのクラスタ１０１２は、コンピューティングノードのクラスタ１０１２上のマスタコンピューティングノード１０１４にアプリケーションをデプロイするように構成してもよい。マスタコンピューティングノード１０１４は、アプリケーションの「アプリケーションコンテキスト」を格納するように構成し得る。「アプリケーションコンテキスト」は、たとえば、アプリケーションのトポロジ、スケジューリング情報、アプリケーションパラメータ等といったアプリケーションの内容を含み得る。

【０１２５】

特定の実施形態において、マスタコンピューティングノード１０１４を、アプリケーションを走らせる／実行する「ドライバプログラム」またはアプリケーションマスタと呼ぶ場合がある。ドライバプログラムは、アプリケーションのメイン（）機能を実行しアプリケーションの「アプリケーションコンテキスト」を生成するプロセスとして定義することができる。ドライバプログラムは、アプリケーションを駆動する役割と、リソースマネージャ１０１６のリソースを要求する役割との両方を有し得る。リソースマネージャ１０１６は、コンピューティングノードのクラスタ１０１２上のコンピューティングノードのためのリソースを取得してアプリケーションを実行するサービスであってもよい。クラスタ上でアプリケーションを走らせる／実行するために、マスタコンピューティングノード１

０１４はリソースマネージャ１０１６に接続する。すると、これは、アプリケーションのリソースを割り当てる。接続されると、マスタコンピューティングノード１０１４は、クラスタ内の１つ以上のコンピューティングノード（ワーカーノード１０１８としても知られている）上の１つ以上のエグゼキュータを取得する。エグゼキュータは、計算を実行しアプリケーション用のデータを格納するプロセスである。マスタコンピューティングノード１０１４は、アプリケーションコード（たとえばＪＡＲファイルによって定められる）をエグゼキュータに送信する。アプリケーションにおいて定められた変換およびアクションに基づいて、マスタコンピューティングノード１０１４はタスク１０２０をエグゼキュータに送信することができる。

【０１２６】

特定の実施形態において、マスタコンピューティングノード１０１４は、ＤＡＧジェネレータ１０２２と、ＤＡＧスケジューラ１０２４と、タスクスケジューラ１０２６と、アプリケーションコンテキスト情報１０２８とを含み得る。上述のように、アプリケーションコンテキスト情報１０２８は、アプリケーションのトポロジ、スケジューリング情報、アプリケ-ションパラメータ等のアプリケーションに関する情報を含み得る。ＤＡＧジェネレータ１０２２は、受信機から受信したＲＤＤオブジェクトに基づいてＲＤＤオブジェクトの有向非巡回グラフ（ＤＡＧ）を規定および／または作成するように構成し得る。いくつかの例において、ＤＡＧジェネレータ１０２２は、特定期間中に受信したすべてのＲＤＤのＲＤＤリネージグラフ（lineage graph）を表していてもよい。ＲＤＤリネージグラフにおける各ＲＤＤオブジェクトは、１つ以上の親に対するポインタを、その親に対してどのような種類の関係を有しているかに関するメタデータとともに、維持する。また、ＲＤＤリネージグラフは、各ＲＤＤオブジェクトに対して実行すべき変換のＤＡＧを特定する。マスタコンピューティングノード１０１４がアプリケーションにおけるジョブを実行することを要求されたとき、ＤＡＧジェネレータ１０２２は変換のＤＡＧを実行する。この変換は、ＲＤＤオブジェクトに対して実行すべき１つ以上の動作を特定し、ＲＤＤオブジェクトにおけるデータをある形態から別の形態に変換することができる。これらの動作は、たとえば、アプリケーションデプロイモジュール１００８によってアプリケーション生成プロセスの一部として定められてもよい。ＲＤＤオブジェクトに動作が適用されると、変換後のデータとともに新たなＲＤＤオブジェクトが得られる。ＲＤＤオブジェクトに対して実行される動作の例は、たとえば、map、filter、flatMap、reduce、union、groupByKey、distinct、join、collect、count等を含み得る。ＣＱＬ言語を要する変換のＤ
ＡＧを本明細書ではＣＱＬ変換と呼ぶ場合がある。

【０１２７】

ＤＡＧスケジューラ１０２４は、ＤＡＧジェネレータが生成したＲＤＤオブジェクトリネージグラフに基づいて物理実行プランを生成するように構成される。一実施形態において、ＤＡＧスケジューラ１０２４は、ＲＤＤリネージグラフを複数の段に分割することによって物理実行プランを生成し、各段は、各ＲＤＤオブジェクトにおけるデータに対して実行する必要がある変換に基づいて特定される。たとえば、ＲＤＤオブジェクトに対して実行すべき変換がmap変換およびreduce変換を含む場合、map変換は、ある段にグループ化してもよく、reduce変換は、別の段にグループ化してもよい。次に、ＤＡＧスケジューラ１０２４は、これらの段をタスクスケジューラ１０２６にサブミットする。

【０１２８】

タスクスケジューラ１０２６は、アプリケーション（ジョブ）を複数の段に分割する。各段は１つ以上のタスクからなる。一実施形態において、特定の段のタスクの数は、ＲＤＤオブジェクトにおける入力データのパーティションの数に基づいて決まる。たとえば、上述のように、ＤＡＧスケジューラ１０２４は、すべてのmap動作を１つの段にスケジューリングしてもよい。次に、これらの段はタスクスケジューラ１０２６に送られ、タスクスケジューラ１０２６はリソースマネージャを介してタスクを開始する。次に、タスクがエグゼキュータノード１０１８によって実行される。タスクスケジューラ１０２６は、各

ＲＤＤオブジェクト（すなわち処理中のイベントの各バッチ）に対するアプリケーション（ジョブ）において定められている動作を実行するコンピューティングノードのクラスタ１０１２におけるノードを特定する。

【０１２９】

特定の実施形態において、エグゼキュータノード１０１８は、ＲＤＤオブジェクトを受信すると、コンピューティングノードのクラスタにおける他のエグゼキュータ（ワーカー）ノードにＲＤＤオブジェクトを送信する必要がある場合は、ＲＤＤオブジェクトにおけるデータのシリアライズおよびデシリアライズを実行する。上述のように、ＲＤＤオブジェクトにおけるデータの処理は、アプリケーションにおいて定められている１つ以上の連続クエリの実行を必要とする場合がある。ある実施形態において、タスクスケジューラ１０２６が特定したエグゼキュータノード（たとえば１０１８）は、ＣＱＬエンジン（ＣＱＬプロセッサ２３０等）を呼び出してＲＤＤオブジェクトにおけるデータの処理を実行し、この処理の結果をマスタコンピューティングノード１０１４に返す。エグゼキュータノード１０１８がその処理に先立ってＲＤＤオブジェクト内のデータのシリアライズおよびデシリアライズを実行し得る手法は、以下において図１１との関連で詳細に説明する。

【０１３０】

図１１は、本開示の実施形態に係る、ＲＤＤオブジェクトにおけるデータのシリアライズおよびデシリアライズを実行するためのプロセスのハイレベルデータフローを示す。特定の例において、図１０における１つ以上の動作は、コンピューティングノードのクラスタにおけるノード（たとえばマスタノードまたはエグゼキュータノード）が、処理のために受信機からＲＤＤオブジェクトを受けたときに、実行することができる。一組の動作において、受信機１１０２は、入力データストリームを受信し、入力データストリームを、特定期間（たとえばＸ秒）のイベント（タプル）の１つ以上のバッチに分割する。ある実施形態において、図１０との関連で述べたように、イベント（タプル）の各バッチは、内部で受信機によってＲＤＤオブジェクトとして表すことができる。ＤＡＧジェネレータ１１０４は、ＲＤＤオブジェクトを受信し、ＲＤＤオブジェクトのＤＡＧ１１０６を生成する。図１０に記載のように、特定の例において、ＲＤＤオブジェクトのＤＡＧ１１０６は、各ＲＤＤオブジェクトに対して実行すべきＣＱＬ変換１１０８を含む。マスタコンピューティングノード（たとえば図１０に示す１０１４）は、アプリケーションにおいてジョブを実行することを要求され、ＤＡＧジェネレータ１１０４は、ＣＱＬ変換１１０８を実行し、ＲＤＤオブジェクトによって表される入力タプルのセットを処理する。いくつかの例において、処理すべき入力タプルのセットは、ＣＱＬ変換１１０８の親（parent）変換から得られる。次に、ＣＱＬ変換の子（child）変換は、ＣＱＬ変換１１０８において表される入力タプルのセットに対して実行すべき特定の動作を呼び出す。

【０１３１】

特定の例において、ＣＱＬ変換１１０８は、ＲＤＤオブジェクトにおける入力タプルのセットに対するバッチシリアライザプロセス１１１２を呼び出してＲＤＤオブジェクトにおけるデータのシリアライズを実行する。ある実施形態において、バッチシリアライザプロセス１１１２は、ＲＤＤオブジェクトを処理しているコンピューティングノードのクラスタにおけるノード（たとえばエグゼキュータノード）によって実行されてもよい。上述のように、ＲＤＤオブジェクトにおけるデータは、イベントストリームを介して受信した入力タプル（イベント）のバッチを表す。バッチシリアライザプロセス１１１２は、ＲＤＤオブジェクトにおけるデータをシリアライズし、バッチシリアライザプロセスから得られた結果のシリアライズされたブロックは、入力タプルのセットを処理するためにネットワークを通してＣＱＬエンジン１１１６に送られる。特定の実施形態において、ＲＤＤオブジェクトを処理しているノードは、ＣＱＬエンジン１１１６を呼び出してＲＤＤオブジェクトにおける入力タプルのセットを処理する。ＣＱＬエンジン１１１６は、たとえば、エグゼキュータノード上にデプロイされたイベント処理エンジン（たとえば図６に記載の６３０）であってもよい。ＣＱＬエンジン１１１６は、入力タプルのセットを受信し、アプリケーションにおいて定められた処理ロジック（動作／変換）に基づいて入力タプルの

セットを処理し、この処理の結果として出力タプルのセットを生成するように構成し得る。
【０１３２】
特定の実施形態において、ＣＱＬエンジン１１１６は、ＲＤＤオブジェクトにおけるデータの処理に先立って、バッチシリアライザプロセス１１１２から受けたデータのシリアライズされたブロックに対するバッチデシリアライザプロセス１１１４を呼び出すことができる。シリアライズされたブロックはネットワークを通して転送するのに適したバイナリフォーマットまたはワイヤフォーマットであり、ＣＱＬエンジン１１１６が処理できるようにするためにはＪＡＶＡオブジェクトとしてデシリアライズされる必要があるからである。ＣＱＬエンジン１１１６は、入力タプルのセットを処理し、この処理に基づいて出力タプルのセットを生成する。特定の例において、ＣＱＬエンジン１１１６は、出力タプルのセットをシリアライズするために別のバッチシリアライザプロセス１１１８を呼び出し、シリアライズの結果は出力タプルのシリアライズされたブロックである。ＤＡＧジェネレータがシリアライズされた出力タプルのセットを受けると、ＣＱＬ変換１１０８は、受けた出力タプルのシリアライズされたブロックに対する別のバッチデシリアライザプロセス１１２０を呼び出す。バッチデシリアライザプロセス１１２０の結果は、デシリアライズされた出力タプルのセットである。ＣＱＬ変換１１０８は、出力タプルのセットをＣＱＬ変換における子変換に戻し、ＲＤＤオブジェクトにおけるデータに対して次の一組の処理動作を実行する。いくつかの実施形態において、その後、出力タプルのセット１１１０は分散イベント処理システムのユーザに送信される。
【０１３３】
特定の実施形態において、上述のバッチシリアライザプロセスおよびバッチデシリアライザプロセスは、ＲＤＤオブジェクトを処理している分散イベント処理システムにおけるノードのクラスタのうちのノード（たとえばエグゼキュータノード）によって実行される、ソフトウェアモジュールまたは命令によって実施されてもよい。バッチシリアライザプロセスおよびバッチデシリアライザプロセスによって実行される動作のその他の詳細は、以下において図１２～図１５との関連で詳しく説明する。
【０１３４】
図１２は、本開示の実施形態に係る、イベントのバッチに含まれるデータをシリアライズできる一組の動作を説明する、プロセス１２００のフロー図の一例である。特定の実施形態おいて、プロセス１２００は、分散イベント処理システムにおけるバッチシリアライザプロセス（１１１２）によって実行されてもよい。上述のように、いくつかの実施形態において、バッチシリアライザプロセスは、アプリケーションにおいて定められたジョブ／動作を実行することを、マスタコンピューティングノード（たとえば図１０に示される１０１４）が要求されたときに、マスタコンピューティングノードによって呼び出されてもよい。先に述べたように、マスタコンピューティングノードは、分散コンピューティングシステム１００２におけるコンピューティングノードのクラスタ１０１２の中のノード（たとえばエグゼキュータノード１０１８）を特定し、アプリケーションにおいて定められたジョブ／動作に対するイベントのバッチを処理し、この処理の結果として出力イベントのセットを生成する。特定の実施形態において、イベントのバッチを処理することは、イベントのバッチにおけるデータのシリアライズを含み得る。図１２のプロセスは、イベントのバッチにおけるデータをシリアライズできる１つの技術を説明する。図１２に示す特定の一連の処理ステップは、限定を意図したものではない。代替の実施形態は、各種配置および組み合わせにおいて図１２に示すステップよりも多くのまたは少ないステップを有し得る。
【０１３５】
特定の実施形態において、図１２に記載の処理は、タスク１０２０（図１０に示される）を介してイベントのバッチを受信するたびに、分散コンピューティングシステム１００２におけるコンピューティングノードのクラスタ１０１２の中のノードによって実行されてもよい。このプロセスは、１２０４においてＣＱＬ変換１１０８からイベントのバッチを受信したときに１２０２で開始される。特定の例において、イベントのバッチにおける

各イベントはタプルと呼ぶ場合があり、イベントのバッチは、入力タプルのバッチまたは入力タプルのセットと呼ぶ場合がある。上述のように、イベントストリームを介して受信する各イベントは、イベントストリームに対応付けられたスキーマに従っており、このスキーマは、イベントストリームを介して受信した各イベントの１つ以上の属性を特定する。

【０１３６】

たとえば、連続イベントストリームは、ある企業が販売する製品に関する製品関連情報を表していてもよく、このイベントストリームにおける各イベントは、商品に対するオーダーを表していてもよい。このような連続イベントストリームは、商品に関する、オーダー識別子（order identifier）、オーダーステータス（order status）、およびオーダー量（order amount）等の属性を含み得る。このような入力ストリームに対するスキーマは、S(timestamp, 〈orderId〉, 〈orderStatus〉, 〈orderAmount〉)として表すことができる。

このようなストリームを介して受信した各イベントはしたがって、タイムスタンプと３つの属性とによって特定されるであろう。特定の実施形態において、イベントの１つ以上の属性を、入力タプルのセット（イベントのバッチ）における１つ以上のカラムで表すことができる。したがって、いくつかの例において、属性は、入力タプルのセットにおけるあるタプル（イベント）のデータ値を格納するカラムを指していてもよい。

【０１３７】

１２０６において、プロセスは、イベントのバッチにおけるイベントの属性（たとえば第１の属性）を特定することを含む。１２０８において、プロセスは、属性のデータタイプを特定することを含む。たとえば、上記オーダー処理ストリームの例ごとに、１２０６および１２０８のプロセスは、属性がイベントの「orderId」に対応すること、および、属性のデータタイプが数値データタイプであることを確認してもよい。１２１０において、プロセスは、属性のデータタイプが数値データタイプか否かを判断することを含む。特定された属性のデータタイプが数値データタイプである場合、特定の実施形態において、プロセスは１２１２に進んで属性が表すデータ値に対して第１のタイプのデータ圧縮を実行すると判断する。たとえば、１２１２において、プロセスは、属性が表すデータ値に対し、数値圧縮技術（たとえば、ベース値圧縮、精度低減圧縮、または精度低減値インデックス）を適用すると判断する。１２１４において、プロセスは、属性によって格納されたデータ値に対して数値圧縮技術を適用した結果として、属性のシリアライズされたデータ値のセットを生成することを含む。イベントの数値属性に対してシリアライズされたデータ値のセットを生成するプロセスは、図１３Ａ、図１３Ｂ、図１３Ｃ、および図１３Ｄに記載される。１２１６において、プロセスは、属性が表すシリアライズされたデータ値のセットを格納することを含む。

【０１３８】

特定の実施形態では、１２１８において、プロセスは、処理する必要があるイベントのその他の属性があるか否かを判断することを含む。処理すべきその他の属性がある場合、プロセスは、１２０６にループバックしてイベントバッチにおけるイベントの次の属性（たとえば第２の属性）を特定し、１２０８～５１６のプロセスを次の属性に対して実行する。

【０１３９】

特定の実施形態において、１２１０で属性の特定されたデータタイプが数値データタイプであると判断されなかった場合、特定の実施形態において、プロセスは１２２０に進み、属性が表すデータ値に対して第２のタイプのデータ圧縮を実行すると判断する。たとえば、上記オーダー処理ストリームの例において、１２０６および１２０８におけるプロセスは、イベントの第２の属性が「orderStatus」属性に対応すること、および、この属性のデータタイプは非数値データタイプであることとを確認してもよい。この場合、プロセスは、１２２０に進んで、属性が格納するデータ値に対して第２のタイプのデータ圧縮を実施すると判断する。ある実施形態において、第２のタイプのデータ圧縮は、第１のタイプのデータ圧縮とは異なっていてもよい。たとえば、１２２０におけるプロセスは、属性

が格納するデータ値に対し、非数値圧縮技術（たとえば値インデックス圧縮）を適用すべきであると判断してもよい。１２１４において、プロセスは、属性が格納するデータ値に対して非数値圧縮技術を適用した結果として、属性が表すシリアライズされたデータ値のセットを生成することを含む。イベントの非数値属性に対してシリアライズされたデータ値のセットを生成し得るプロセスは図１４に記載されている。１２１６で、プロセスは、属性が表すシリアライズされたデータ値のセットを格納することを含む。

【０１４０】

特定の実施形態において、プロセスは続いて１２１８に進み、特定し処理すべきイベントの他の属性があるか否かを判断する。より多くの属性がある場合、プロセスは１２０６にループバックしてイベントのバッチにおけるイベントの第３の属性を特定する。次に、このプロセスは１２０８において第３の属性のデータタイプを特定してもよく、このプロセスは、１２１０において、第３の属性のデータタイプに基づいて第３の属性が格納するデータ値に対して実行する第３のタイプのデータ圧縮を決定することを含み得る。たとえば、上記オーダー処理イベントストリームの例の場合、第３のタイプのデータ圧縮は、属性のデータタイプに基づいて「orderAmount」が格納するデータ値に対して実行してもよい。特定の例において、イベントのすべての属性が特定および処理されると、プロセスは１２２２で終了する。

【０１４１】

図１３Ａは、本開示の実施形態に係る、イベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１３００のフロー図の一例である。ある実施形態において、プロセス１３００は、図１２の１２１４におけるプロセスによって実行される動作のさらに他の詳細を説明する。特定の例において、プロセス１３００は、１３０２において、そのデータ値が処理されているカラムの現在のカラム番号（たとえば第１の属性）に対する現在のバッファオフセットを格納することで、開始される。１３０４で、プロセスは、属性のデータタイプ（カラムタイプ）を取得することを含む。たとえば、この場合の属性のデータタイプは、数値属性であると判断することができる。１３０６において、プロセスは、入力タプルのセットをスキャンすることにより、最小値、最大値、および属性によって表される一組の固有値を取得することを含む。１３０８において、プロセスは、範囲（最大～最小）からの属性によって表されるデータ値を格納するのに必要なビット数を計算することを含む。１３０９において、プロセスは、必要なビット数は属性のデータタイプのビット数の２分の１よりも大きいか否か、および、一組の固有値のサイズは入力タプルの数／値__インデックス__しきい値（value_index_threshold）よりも小さいか否かを判断することを含み、値__インデックス__しきい値は設定可能である。一例において、値__インデックス__しきい値は、デフォルト値としての１１という値に設定することができる。大きい場合、プロセスは１３５０に進み、精度低減インデックス値圧縮技術を実行する。１３１０において、プロセスは、必要なビット数はカラムの元のデータタイプのビット数よりも小さいか否かを判断する。１３１０での検査は、シリアライズされたブロックのサイズが元のブロックから増大していないことを確認するために実行される。その理由は次の通りである。もし値の範囲をカバーするのに必要なビットが元のデータに必要なビットよりも多ければ、結果として値インデックス技術を用いて生成されたブロックは元のブロックサイズよりも大きい可能性があるからである。

【０１４２】

必要なビット数がカラムの元のデータタイプのビット数よりも少ない場合、プロセスは１３１２に進み、一組の固有値のサイズが入力タプル数／２よりも小さいか否かを判断する。この判断は、圧縮レートが十分に大きいことを確実にするために実行される。特定の例において、固有値が多すぎる場合、値と値に対するインデックスとを使用する代わりに値そのものを使用する。必要なビット数がカラムの元のデータタイプよりも少なくかつ一組の固有値のサイズが入力タプル数／２よりも少ない場合、１３１４－６２６に記載のプロセスが実行される。

【０１４３】

たとえば、１３１４において、プロセスは、データ圧縮の精度低減値インデックスタイプとして属性によって表されるデータ値に対して実行すべき第１のタイプのデータ圧縮を格納することを含む。精度低減技術は、値の範囲を発見することによって使用される値からの値を表すビットを低減する。１３１６において、プロセスは、属性の最小データ値を格納することを含む。１３１８において、プロセスは、最小値ごとにビット数を格納することを含む。１３２０において、プロセスは、そのデータ値が現在処理値であるカラム（たとえば属性）の各データ値に対して１３２２および１３２４における動作を実行することを含む。たとえば、１３２２において、プロセスは、一組の固有データ値からインデックスを取得することを含む。１３２４において、プロセスは、インデックスをバッファに格納することを含む。カラムのすべてのデータ値が処理された後、プロセスは、１３２６において、タプルのセットにおける固有値ごとに（固有値－最小値）を格納することを含む。これらの値は、ステップ１３２４で格納されたインデックスからインデックスされた実際の値を表す。

【０１４４】

特定の実施形態において、プロセスが１３１２において一組の固有値のサイズが入力タプル数／２よりも小さくないと判断した場合、いくつかの実施形態において、以下で説明する図１３Ｂのプロセス１３３２－６３８が実行される。特定の実施形態において、プロセスが１３１０において必要なビット数がカラムの元のデータタイプよりも小さくないと判断した場合、一実施形態において、図１３Ｃに記載の１３４２－６４６に記載のプロセスが実行される。特定の実施形態において、プロセスは、１３２８において、属性のシリアライズされたデータ値のセット（すなわちデータのシリアライズされたブロック）を、イベントストリームを介して受信したタプルのセット（すなわちイベントのバッチ）を処理するために、ＣＱＬエンジンに返すことにより、終了する。

【０１４５】

図１３Ｂは、本開示の実施形態に係る、精度低減圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１３５０のフロー図の一例である。ある実施形態において、プロセス１３５０は、図１３Ａのプロセス１３３０によって実行される動作のその他の詳細を説明する。特定の例において、プロセス１３５０は、１３３２において、精度低減圧縮として、属性によって表されるデータ値に対して実行すべきデータ圧縮のタイプを格納することにより、開始される。１３３４において、プロセスは、属性の最小データ値を格納することを含む。１３３６において、プロセスは、属性のデータ値当たりのビット数を格納することを含む。１３３８において、プロセスは、カラムのデータ値ごとに、必要なビットについてのみ、ビットコピー（値－最小）を実行することを含む。たとえば、一組の入力値（１０，１１，１２）は、（１０－１０（最小値））の結果である値０に対するビット００、（１１－１０）の結果である値１に対するビット０１、および、（１２－１０）の結果である値２に対するビット０２とともに、格納される。ビット値００、０１、および０２のシーケンスは、バイト（８ビット）０００１０２００に格納し、１６進数値の１５４として格納することができる。

【０１４６】

図１３Ｃは、本開示の実施形態に係る、通常の圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１３６０のフロー図の一例である。ある実施形態において、プロセス１３６０は、図１３Ａの１３４０におけるプロセスによって実行される動作のその他の詳細を説明する。特定の例において、プロセス１３６０は、１３４２において、一般圧縮タイプとして、属性によって表されるデータ値に対して実行される第１のタイプのデータ圧縮を格納することにより、開始される。１３４４において、プロセスは、zipまたはgzip等の標準圧縮技術を用いてカラム値のアレイを圧縮することを含む。１３４６において、プロセスは、属性が表すデータ値の圧縮されたバイトを格納することを含む。

【０１４７】

図１３Ｄは、本開示の実施形態に係る、精度低減値インデックス圧縮技術を用いてイベントの数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１３７０のフロー図の一例である。ある実施形態において、プロセス１３７０は、図１３Ａの１３５０におけるプロセスによって実行される動作のその他の詳細を説明する。特定の例において、プロセス１３７０は、１３７２において、精度低減インデックス値圧縮として、属性によって表されるデータ値に対して実行すべきデータ圧縮のタイプを格納することにより、開始される。１３７４において、差分値セット（たとえば値－最小値）が計算される。１３７６において、プロセスは、差分値セットにおけるすべての値をスキャンすることにより列挙された値のセットを取得することを含む。１３７８において、プロセスは、差分値セットによって表される各データ値についてインデックスのセットを計算することを含む。１３８０において、プロセスは、インデックスのセットから最大および最小値を計算することを含む。１３８２において、プロセスは、属性の最小データ値を格納することを含む。１３８４において、プロセスは、インデックス値のデータ値当たりのビット数を格納することを含む。１３８６において、プロセスは、カラムの各データ値について、必要ビットに対してビットコピーのみを実行することを含む。１３８８において、プロセスは、列挙された差分値のセットを格納することを含む。

【０１４８】

図１４は、本開示の実施形態に係る、イベントの非数値属性のシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１４００のフロー図の一例である。ある実施形態において、プロセス１４００は、属性（たとえば非数値属性）によって表されるデータ値に対して第２のタイプのデータ圧縮を実行すべきと判断された場合に図１２のプロセス１２１４によって実行される動作のその他の詳細を説明する。プロセス１４００は、１４０２において、そのデータ値が処理中であるカラム（たとえば属性）の現在のカラム番号に対する現在のバッファオフセットを格納することにより、開始される。１４０４において、属性のデータタイプ（カラムタイプ）を取得する。たとえば、この場合、属性のデータタイプは非数値属性であると判断される。１４０６において、プロセスは、値インデックス圧縮として、属性によって表されるデータ値に対して実行すべきデータ圧縮のタイプを格納する。この場合、入力のバッチ内の可能なすべての値が列挙され、値を複数回コピーするのではなく、場所のインデックスを使用する。

【０１４９】

１４０８において、プロセスは、カラムの列挙された値のセットを取得するためにすべての入力タプルをスキャンすることを含む。１４１０において、プロセスは、カラムによって表される各データ値についてインデックスのセットを計算することを含む。１４１２において、プロセスは、カラムに格納されている各データ値について、以下で１４１４－７１６で説明する動作を実行することを含む。１４１４において、プロセスは、列挙された値のセットからインデックスを取得することを含む。１４１６において、プロセスは、インデックスをバッファに格納することを含む。１４１８において、プロセスは、列挙された値のセットを格納することを含む。１４２０において、プロセスは、イベントストリームを介して受信したタプルのセット（すなわちイベントのバッチ）を処理するために、ＣＱＬエンジンに、属性についてのデータ値のセット（すなわちデータのシリアライズされたブロック）を返すことにより、終了する。

【０１５０】

図１５は、本開示の実施形態に係る、イベントストリームにおけるイベントの属性のデータタイプの判断に基づいてイベントストリームデータをシリアライズできる手法の一例である。以下に示す例において、イベントストリームは、ある企業が販売する製品に関する製品関連情報を表す。イベントストリーム内の各イベントは、商品に対するオーダーを表していてもよく、商品に関する、オーダー識別子、オーダーステータス、およびオーダー量等の属性を含み得る。このようなオーダーイベントストリームに対するスキーマは、S(timestamp, 〈orderId〉, 〈orderStatus〉, 〈orderAmount〉)として表すことができる。こ

のようなストリームを介して受信した各イベントはしたがって、タイムスタンプと３つの属性とによって特定できる。一例として、オーダーイベントストリームを介して受信するイベントのバッチは、以下のイベントと、対応付けられたタイムスタンプとを含み得る。

【０１５１】

【数５】

...

(timestamp_N, 10, “open”,100)

(timestamp_N+1, 11, “open”,5000)

(timestamp_N+2, 10,”processing”,100)

(timestamp_N+3, 10,”shipped”,100)

(timestamp_N+4, 11, “processing”,5000)

(timestamp_N+5, 10,”closed”,100)

(timestamp_N+6, 11,”shipped”,5000)

(timestamp_N+7, 11,”closed”,5000)

【０１５２】

上述のように、特定の実施形態において、イベントの１つ以上の属性は、イベントのバッチを表す入力タプルのセットにおける１つ以上のカラムとして表すことができる。よって、いくつかの例において、属性は、入力タプルのセットにおけるタプル（イベント）のデータ値を格納するカラムを指す場合がある。オーダーイベントストリームを介して受信するイベントのバッチに対応する入力タプルのセットの一例を以下の表１に示す。

【０１５３】

【表１】

タプル（イベント）	属性１（カラム１） オーダーＩｄ	属性２（カラム２） オーダーステータス	属性３（カラム３） オーダー量
e1	10	Open	100
e2	11	Open	5000
e3	10	Processing	100
e4	10	Shipped	100
e5	11	Processing	5000
e6	10	Closed	100
e7	11	Shipped	5000
e8	11	Closed	5000

表１

【０１５４】
特定の実施形態において、イベントバッチにおけるイベントは、イベントの各属性のデータタイプを特定し、属性のデータタイプに基づいて各属性が表すデータ値に適用すべき特定種類の圧縮技術を判断することにより、シリアライズされる。たとえば、第１の圧縮技術を、第１の属性は数値属性であるという判断に基づいてイベントの第１の属性（たとえばオーダーｉｄ属性）に適用してもよく、第２の圧縮技術を、第２の属性が非数値属性であるという判断に基づいてイベントの第２の属性（たとえばオーダーステータス属性）に適用してもよく、第３の圧縮技術を、第３の属性は数値属性であるという判断に基づいてイベントの第３の属性（たとえばオーダー量属性）に適用してもよい。特定の例において、第１のタイプの圧縮技術、第２のタイプの圧縮技術、および第３のタイプの圧縮技術は、互いに異なっていてもよい。

【０１５５】
ある実施形態において、カラムストレージを用いて、カラム内の値が同一データタイプになるように、同一のデータタイプを有する属性（カラム）を格納することができる。特定の実施形態において、数値タイプのカラムに格納された値は、ベース値圧縮技術または精度低減圧縮技術を用いて圧縮することができる。精度低減は、値の範囲を発見することにより、使用値からの値を表すビットを低減する。必要なビットは値の範囲に応じて決まる。ベース値圧縮は、ベース値として最小値を使用し、その他の値からのベース値からの差分を格納する。たとえば、イベントバッチ（１０，１１，１０，１０，１１，１０，１１，１１）における各イベントの「オーダーｉｄ」を表す一組の入力値を、（１０，１）、２進数０１００１０１１、または１６進数０×４Ｂに圧縮できる。これは、３２ビットから２ビットに減じられたビットを用いて値（０，１，０，０，１，０，１，１）を表す。なぜなら、最小値は１０、範囲は２であるからである。別の例として、「オーダー量」を表す一組の入力値は、精度低減および値インデックス技術を用いて圧縮できる。この場合、オーダー量を表す一組の入力値（１００，５０００，１００，１００，５０００，１００，５０００，５０００）は、精度低減技術および値インデックス技術双方を用いて（１００，２，０×１０，０×４Ｆ）および（０，４９００）に圧縮できる。入力セットは、１００をベース値とする（０，４９００，０，０，４９００，０，４９００，４９００）で表すことができる。結果セットの値は、２進数で０００１００００，０１００１１１１、１６進数で１０および４Ｆである。これは、ベース値テーブル（０，４９００）に対するインデックスを有する（０，１，０，０，１，０，１，１）を表す（たとえば、０は、０を指し、さらにベース値を１００とするの１００を指し、１は、４９００を指し、さらにベース値を１００とする５０００を指す）。

【０１５６】
特定の実施形態において、「値インデックス圧縮」技術を用いて、ストリング値等の非数値を格納するカラムの値を処理してもよい。この場合において、入力のバッチ内の可能なすべての値を列挙し、値を複数回コピーする代わりに、場所のインデックスを用いる。たとえば、「オーダーステータス」属性（カラム）の値が、（open, open, processing, shipped, processing, closed, shipped, closed）である場合、対応する列挙された固有値は、（open, processing, shipped, closed）であろう。カラムの値が線形バッファに連続的に格納される場合、各値のインデックスは（０，５，１７，２５）であろう。なぜなら、このバッファは、open/0processing/0shipped/0closed/0を有するであろうからで
ある。/0は、ストリングマーカの最後を示す。値の線形バッファを用いた場合の最終圧縮結果は、（０，０，５，１７，５，２５，１７，２５）である。

【０１５７】
図１６は、本開示の実施形態に係る、イベントのバッチに含まれるデータをデシリアライズできる一組の動作を説明するプロセス１６００のフロー図の一例である。特定の実施形態において、プロセス１６００は、分散イベント処理システムにおけるバッチデシリアライザプロセス（４２０）によって実行することができる。上述のように、バッチデシリ

アライザプロセスは、特定の実施形態において、アプリケーションにおいて定められたジョブ／動作を実行することをマスタコンピューティングノード（たとえば図３に示す３１４）が要求されたときに、マスタコンピューティングノードが呼び出してもよい。上述のように、マスタコンピューティングノードは、分散コンピューティングシステム３０２におけるコンピューティングノードのクラスタ３１２のノード（たとえばエグゼキュータノード３１８）を特定し、アプリケーションにおいて定められたジョブ／動作に対してイベントのバッチを処理し、処理の結果として出力イベントのセットを生成する。特定の実施形態において、イベントのバッチを処理することは、イベントのバッチにおけるデータのシリアライズとその後のデシリアライズとを含み得る。図１６のプロセスは、イベントのバッチにおけるデータをデシリアライズできる技術を説明する。図１６に示す特定の一連の処理ステップは、限定を意図したものではない。代替の実施形態は、各種配置および組み合わせにおいて図１６に示すステップよりも多くのまたは少ないステップを有し得る。

【０１５８】

特定の実施形態において、プロセス１６００は、１６０２において、イベントのバッチ（入力タプルのセット）におけるイベントの１つ以上の属性に対応するシリアライズされたデータ値のセットを受信することで開始される。１６０４において、プロセスは、イベントのバッチにおけるイベントの１つ以上の属性に対応するシリアライズされたデータ値のセットを処理することにより、出力イベントのセットを生成する。特定の例において、１６０４におけるプロセスは、１６０６における、シリアライズされたデータ値のセットに基づいて属性に対応するデシリアライズされたデータ値のセットを生成することと、１６０８における、１つ以上の連続クエリのセットに対して属性に対応するデシリアライズされたデータ値のセットを処理することにより、出力イベントの第１のセットを生成することとを含み得る。１６１０において、プロセスは、出力イベントのセットを分散イベント処理システムのユーザに送信することを含む。

【０１５９】

イベントのバッチにおけるイベントの属性に対応するデシリアライズされたデータ値のセットを生成するプロセス１６０４を、以下で図１７～図１４との関連で詳細に説明する。具体的には、図１７は、イベントのバッチにおけるイベントの１つ以上の属性に対応するデシリアライズされたデータ値のセットを生成できるプロセスを説明している。図１８は、イベントの非数値属性に対応するデシリアライズされたデータ値のセットを生成できるプロセスを説明している。図１９～図１４は、イベントの数値属性に対応するデシリアライズされたデータ値のセットを生成できるプロセスを説明している。

【０１６０】

図１７は、本開示の実施形態に係る、イベントのバッチにおけるイベントの１つ以上の属性のデシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１７００のフロー図の一例である。ある実施形態において、プロセス１７００は、図１６の１６０４におけるプロセスの動作のその他の詳細を説明する。特定の例において、プロセス１７００は、１７０２においてタプルのアレイを生成することによって開始される。１７０４において、プロセスは、イベントの第１のカラム（第１の属性）を特定することを含む。１７０６において、プロセスは、そのデータ値が処理されているカラム（たとえば第１の属性）の現在のカラム番号のバッファオフセットを取得することを含む。１７０８において、プロセスは、属性の圧縮タイプを読み取ることを含む。これは、たとえば、属性のデータ値をシリアライズするためにバッチシリアライザプロセスによって実行されたデータ圧縮のタイプを読み取ることを含む。１７１０において、プロセスは、属性に適用されるデータ圧縮のタイプが値インデックス圧縮であるか否かを判断することを含む。属性に適用される圧縮タイプが値インデックス圧縮である場合、プロセスは、１７２４に進み、図１８に記載のプロセスを実行する。

【０１６１】

１７１２において、プロセスは、属性に適用されるデータ圧縮のタイプが精度低減圧縮であるか否かを判断することを含む。属性に適用される圧縮タイプが精度低減圧縮である

場合、プロセスは、１７２６に進み、図１９に記載のプロセスを実行する。
【０１６２】
特定の実施形態では、１７１４において、プロセスは、属性に適用されるデータ圧縮タイプのタイプが精度低減値インデックス圧縮であるか否かを判断することを含む。属性に適用される圧縮タイプが精度低減値インデックス圧縮である場合、プロセスは、１７２８に進み、図２０に記載のプロセスを実行する。
【０１６３】
特定の実施形態において、プロセスが、属性に適用される圧縮タイプは、値インデックス圧縮、精度低減圧縮、および精度低減値インデックス圧縮のうちのいずれでもないと判断した場合、プロセスは、１７１６に進み、属性に適用される圧縮タイプは一般的な圧縮であると判断する。１７１６において、プロセスは、図２１に記載のプロセスを実行することを含む。
【０１６４】
１７１８において、プロセスは、処理すべき他の属性があるか否かを判断することを含む。処理すべき他の属性がある場合、プロセスは、１７０４にループバックしてイベントの次の属性を特定し処理する。処理すべき属性がない場合、いくつかの実施形態において、プロセスは、１７２０においてさらに処理するためにタプルのアレイをＣＱＬエンジンに返す。特定の実施形態において、プロセスは１７２２で終了する。
【０１６５】
図１８は、本開示の実施形態に係る、値インデックス圧縮を用いてイベントのバッチにおけるイベントの数値属性または非数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１８００のフロー図の一例である。ある実施形態において、プロセス１８１０は、図１７のプロセス１７２４のその他の詳細を説明し、属性に適用されるデータ圧縮のタイプが「値インデックス」圧縮であると判断された場合（たとえば図１７の１７１０）に実行される。一例として、プロセス１８００は、オーダーイベントストリームにおける「オーダーステータス」属性（非数値属性）に対して実行することができる。
【０１６６】
値インデックス圧縮技術を用いる例は、図８、カラム２に示される。圧縮された値は、２セットのデータとして、（０，０，５，１７，５，２５，１７，２５）と、（‘open’,’processing’,’shipped’,’closed’）とを有する。第２のセットは、実際の値を有するので、value_arraysと呼ぶ。第１のセットは、value_arraysに格納される実際の値に対するインデックス値を含むので、index_valuesと呼ぶ。value_indexは、index_values における個々のインデックス値各々を指す。
【０１６７】
特定の実施形態において、プロセス１８００は、１８０２において、インデックス値をindex_valuesに読み込むことによって開始される。１８０４において、プロセスは、値アレイをvalue_arrayに読み込むことを含む。１８０６において、プロセスは、入力タプルのセットにおける属性に対応する各データ値に対し、１８０８、１８１０、および１８１２の動作を実行することを含む。たとえば、１８０８において、プロセスは、index_values[value_index]からインデックスを取得することを含む。１８１０において、プロセスは、value_array[index]から値を取得することを含む。１８１２において、プロセスは、tuples[value_index]のタプルカラムに値をセットすることを含む。
【０１６８】
図１９は、本開示の実施形態に係る、精度低減圧縮技術を用いてイベントのバッチにおけるイベントの数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス１９００のフロー図の一例である。ある実施形態において、プロセス１９００は、図１７のプロセス１７２６のその他の詳細を説明し、属性に適用されるデータ圧縮のタイプが「精度低減」圧縮であると判断されたときに（たとえば図１７の１７１２）実行される。一例として、プロセス１９００は、オーダーイベント

ストリームにおける「オーダーｉｄ」または「オーダー量」属性（数値属性）に対して実行することができ、圧縮結果が図８、カラム１に示され、（１０，２，０×４Ｂ）である。一例として、下記一組の動作において、用語「最小値」は、値の範囲の最小値である値１７を有し、用語「ビット数」は、値の範囲を表すビット数である値２を有する。

【０１６９】

特定の実施形態において、プロセス１９００は、１９０２において、最小値をbase_valueに読み込むことによって開始される。１９０４において、プロセスは、ビット数を読み取ることを含む。１９０６において、プロセスは、入力タプルのセットにおける属性に対応する各データ値について１９０８および１９１０のプロセスを実行することを含む。たとえば、１９０８において、プロセスは、値ビットをvalue_bitsに読み込むことを含む。１９１０において、プロセスは、base_value+value_bitsを、tuples[value_index]のタプルカラムにセットすることを含む。

【０１７０】

図２０は、本開示の実施形態に係る、精度低減値インデックスを用いて、イベントのバッチにおけるイベントの数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス２０００のフロー図の一例である。ある実施形態において、プロセス２０００は、図１７のプロセス１７２８のその他の詳細を説明し、属性に適用されるデータ圧縮のタイプが「精度低減値インデックス」圧縮であると判断されたときに（たとえば図１７の１７１２）実行される。一例として、プロセス２０００は、オーダーイベントストリームにおける「オーダーｉｄ」または「オーダー量」属性（数値属性）に対して実行することができる。図８は、カラム３において、「オーダー量属性」についての結果の一例を示し、結果は（１００，２，０×１０，０×４Ｆ）および（０，４９００）である。

【０１７１】

下記一組の動作において、用語「ベース値」は、カラム値のベース値の値１７０を指し、用語「ビット数」は、インデックス値のビット数の値２を指し、用語「index_values」は、インデックス値の値（０×１０，０×４Ｆ）を指し、用語「value_array」は、一組
の異なる値を表す（０，４９００）を指す。

【０１７２】

特定の実施形態において、プロセス２０００は、２００２において、最小値をbase_valueに読み込むことによって開始される。２００４において、プロセスは、ビット数を読み取ることを含む。２００６において、プロセスは、インデックス値をindex_valuesに読み込むことを含む。２００８において、プロセスは、値アレイをvalue_arrayに読み込むこ
とを含む。２０１０において、プロセスは、j←0 to value_array.lengthの各値に対し、value_array[j]をvalue_array[j] + base_valueにセットすることを含む。２０１２において、プロセスは、入力タプルのセットにおける属性に対応する各データ値に対し、２０１４、２０１６、および２０１８のプロセスを実行することを含む。たとえば、２０１４において、プロセスは、index_values[value_index]からインデックスを取得することを含む。２０１６において、プロセスは、value_array[index]から値を取得することを含む。２０１８において、プロセスは、tuples[value_index]のタプルカラムに値をセットす
ることを含む。

【０１７３】

図２１は、本開示の実施形態に係る、イベントのバッチにおけるイベントの数値属性または非数値属性に対応する、デシリアライズされたデータ値のセットを生成するための一組の動作を説明するプロセス２１００のフロー図の一例である。ある実施形態において、プロセス２１００は図１７のプロセス１７１６のその他の詳細を説明し、属性に適用されるデータ圧縮のタイプが「一般的な圧縮」技術であると判断されたときに（たとえば図１

８の１７１６）実行される。一例として、プロセス２１００は、オーダーイベントストリームにおける「オーダーｉｄ」または「オーダー量」（数値属性）に対して実行することができる。下記一組の動作において、用語「値」は復元された値を指す。

【０１７４】

特定の実施形態において、プロセス２１００は、２１０２において、ブロックを値のアレイに復元することによって開始される。２１０４において、プロセスは、入力タプルのセットにおける属性に対応する各データ値について、アレイにおける値をタプルのタプルカラムにセットすることを含む。

【０１７５】

マイクロバッチベースのイベント処理システム内の複数のＣＥＰエンジンのスケジューリングおよび管理

近年、場合によっては無制限であるリアルタイムデータストリームにわたって連続的にクエリを実行できるデータストリーム管理システム（data stream management system）

（ＤＳＭ）が開発された。新たなＤＳＭのうち、これらのシステムは、１つのフレームワークからバッチ処理とストリーム処理との組み合わせを提供するために、マイクロバッチ方式ベースのストリーム処理を採用する。このようなシステムの一例は、Spark（登録商標）プラットフォーム上で実行するSpark（登録商標） Streamingアプリケーションである。

【０１７６】

マイクロバッチ方式のストリーム処理は、ステートフルな処理が一般的に複雑であるというシステム設計の性質を原因とするいくつかの欠点を有する。このような欠点の１つは、「パターンマッチング」動作を実行できないことである。パターンマッチングは、ストリーム処理システムがサポートすべき、望ましい重要な特徴であり、パターンマッチングは、ステートマシンを実行して無制限のイベントストリームからパターンを検出するために非常にステートフルな処理を必要とする。

【０１７７】

十分にステートフルなクエリ処理をサポートするために、開示される技術は、マイクロバッチ方式のストリーム処理にＣＱＬクエリエンジンを追加する。クラスタ内には２つ以上のＣＱＬエンジンが存在するので、スケジューリング、トラッキング、および局所性に関する問題に対処しなければならない。

【０１７８】

図２２は、ＣＱＬエンジントラッカーにおけるスケジューリングプロセスを実現できるシステムまたはアーキテクチャの一例である。一実施形態において、以下の図２２に示すように、ＣＱＬエンジン２２１２とＣＱＬレジリエント分散データセット（ＲＤＤ）２２１８との間において遠隔通信可能なドライバ（マスタ）２２０６内のＣＱＬエンジントラッカー（CQL Engine Tracker）２２０２というコンポーネントが開示されている。起動およびスケジューリングのために、CQLEngineTracker２２０２は、２ステップのスケジューリングポリシーを用いることにより、異なるシステム環境を区別する。一実施形態において、局所性を最大にするために、CQLEngineTracker２２０２は以下のアフィニティアルゴリズムを使用する。

１．すべてのCQLEngine２２１２、２２１４、２２１６を、ドライバ２２０６のCQLEngineTracker２２０２によって起動する。好ましい場所に対するCQLEngineの対応付けは設定されない。

２．最初のCQLRDD２２１８は好ましい場所の情報を有しない。

３．スケジューラ２２０４は、親ＲＤＤを親ＲＤＤの好ましい場所を用いて配置できるホストに、ともに位置しようと試みる。

４．CQLRDD２２１８の最初の実行は、CQLEngine２２１２を同一のホスト２２０８に対応
付ける。

５．次のCQLRDD２２２０は、ステップ４からの対応付け情報セットから得た好ましい場所情報を設定する。
６．スケジューラ２２０４は、それが設定された好ましい場所にCQLRDDを走らせようと試みる。
【０１７９】
開示される技術により、マイクロバッチ方式のストリーム処理内で十分にステートフルなCQLEngine２２１２、２２１４、２２１６を有することができ、CQLEngineとCQLRDDとの間で局所性を保つことができ、CQLEngineを起動および再起動するためのマルチステップ
スケジューリングアルゴリズムが可能である。加えて、開示するローカルアフィニティアルゴリズムは、その他のイベントごとのストリーム処理システムと比較して最大の性能を提供する。
【０１８０】
特定の実施形態において、開示するＣＱＬエンジントラッカー２２０２は、クラスタ内のCQLEngineをスケジューリングし、トラッキングし、再始動する役割を果たす。ＣＱＬエンジンは、クラスタ内において長時間実行タスクとして実行され、通常のストリーミングジョブとして起動することができる。ＣＱＬエンジントラッカー２２０２は、障害という事態に遭遇したときを除いてリターンしない。
【０１８１】
いくつかの実施形態において、ＣＱＬエンジントラッカー２２０２による以下のトラッキング情報を維持することができる。
・state: CQLEngineState - INACTIVE, SCHEDULED, ACTIVE
－これは、CQLEngine２２１２、２２１４、２２１６のライフサイクルにわたり、INACTIVE -〉 SCHEDULED -〉 ACTIVE -〉 INACTIVEのように変化する。
・scheduleLocation：TaskLocation
－スケジュールされた最初の場所
・runningExecutor：ExecutorCacheTaskLocation
－CQLEngineが実際に実行されるエグゼキュータ２２０８の場所
・name：String
－CQLEngine２２１２、２２１４、２２１６の名称
・endpoint：RpcEndpointRef
－遠隔アクセスするためのCQLEngine２２１２、２２１４、２２１６の遠隔プロセスコール（Remote Process Call）（ＲＰＣ）
・errorInfo：CQLEngineErrorInfo
－最後に知られたエラー情報
ある実施形態において、ＣＱＬエンジン２２１２、２２１４、２２１６の起動フローは次のように説明できる。
・起動するCQLEngine２２１２、２２１４、２２１６の数を判断
・エグゼキュータ２２０８、２２１０、２２１２のリストを取得
・エグゼキュータ２２０８、２２１０、２２１２のリストにCQLEngine２２１２、２２１４、２２１６をスケジューリングするためにラウンドロビンスケジューラを実行
・TaskScheduler２２０４が実際の長時間実行タスクを起動
・新たに起動したCQLEngineがCQLEngineTracker（たとえばＣＱＬエンジントラッカー２
２０２）に対する「登録」ＲＰＣコールを呼び出す
特定の実施形態において、CQLEngine局所性アフィニティアルゴリズムは以下のプロセスによって説明することができる。
１．すべてのCQLEngineは、ドライバのCQLEngineTracker２２０２によって起動される。
好ましい場所に対するCQLEngineの対応付けは設定されない。

２．第１のCQLRDDは好ましい場所情報を有しない。
３．スケジューラは、親ＲＤＤを親ＲＤＤの好ましい場所を用いて配置できるホストに、ともに位置しようと試みる。
４．CQLRDDの最初の実行は、CQLEngineを同一のホストに対応付ける。
５．次のCQLRDDは、ステップ４からの対応付け情報セットから得た好ましい場所情報を設定する。
６．スケジューラは、それが設定された好ましい場所にCQLRDDを走らせようと試みる。
【０１８２】
ある実施形態において、CQLEngine再始動スケジューリングプロセスは次のように説明することができる。
・２つのケース（リジェクト（Rejected）、クラッシュ（Crashed））を扱う
・リジェクト－スケジュール場所と実際の場所とが異なる場合（スケジュール通りに始動していない）
・schedulePolicy.rescheduleCQLEngineで、古いスケジューリングされたエグゼキュータ
（アクティブでないものを除く）または新しいスケジューリングされたエグゼキュータのいずれかを用いて、スケジューリングされたエグゼキュータを得る
－スケジューリングされた場所のリストにおいてまだ生きているエグゼキュータを選択
・スケジューリングされたエグゼキュータでCQLEngineを始動
以下のフローは、上記アーキテクデャのデータフローを示す。
１．ドライバ２２０６におけるCQLEngineTracker２２０２は、CQLEngineごとに、長時間
実行タスクを起動する。CQLEngineTracker２２０２は、そのRPCEndpointを、長時間実行
タスクにエクスポーズする。
２．TaskScheduler２２０４は、クラスタ内のエグゼキュータ２２０８、２２１０、２２１２に対する長時間実行タスクを実行する。
３．長時間実行タスクの一部として、CQLEngineはエグゼキュータ２２０８、２２１０、２２１２から実行する。
４．CQLEngineは自身をCQLEngineのRPCEndpointとともにドライバ２２０６におけるCQLEngineTracker２２０２に登録する。
５．ストリーミングＤＡＧの一部として、ＣＥＰ処理を担当するＣＱＬＲＤＤがある。ＣＱＬＲＤＤは、ローカルCQLEngineまたは遠隔CQLEngineいずれかにより、CQLEngine Trackerにコンサルティングすることによって処理される。遠隔CQLEngineはＲＰＣを通して呼
び出される。
【０１８３】
CQLEngineの始動
CQLEngineは、クラスタ内の長時間実行タスクとして実行される。CQLEngineは、CQLEngineTrackerによって通常ジョブとして始動されるが、リターンすることはなく、障害またはクラッシュの場合を除いて実行を続ける。CQLEngineTrackerは、以下に記載のアルゴリズムに従い、クラスタ内のCQLEngineを起動する。
１．起動するCQLEngineの数を判断する
２．エグゼキュータのリストを取得する
３．ラウンドロビンスケジューラを実行し、エグゼキュータのリストに対してCQLEngineをスケジューリングする
４．TaskSchedulerが実際の長時間実行タスクを起動する
５．新たに起動されたCQLEngineは、CQLEngineTrackerに対するＲＰＣコール「登録」を
呼び出す

６．長時間実行タスクは、CQLEngineがクラッシュしたときまたはその他の障害の場合にのみ、リターンする

ステップ１において、クラスタ内の起動するCQLEngineの数を判断する。クラスタ内のCQLEngineのデフォルト数は、クラスタ内のエグゼキュータの数と同一である。結果として、１つのCQLEngineが各エグゼキュータから実行される。CQLEngineの最大数を設定することができる。

【０１８４】

ステップ２において、エグゼキュータリスト情報（エグゼキュータホストおよびエグゼキュータｉｄ）をクラスタから取り出す。

【０１８５】

ステップ３において、ラウンドロビンスケジューラはCQLEngineにエグゼキュータを割り当てる。

【０１８６】

ステップ４において、CQLEngineごとに長時間実行タスクを起動する。TaskSchedulerは、スケジューリングされたエグゼキュータ情報（エグゼキュータホストおよびｉｄ）を用いてスケジューリングされたエグゼキュータにおいてCQLEngineを始動する。

【０１８７】

ステップ５において、新たに起動されたCQLEngineは、CQLEngineTrackerに対するＲＰＣコール「登録」を呼び出す。このステップは、以下に示すCQLEngineTrackerからトラッキングプロセスを開始する。

【０１８８】

ステップ６において、CQLEngineの故障またはクラッシュが、以下に示すCQLEngineTrackerからリカバリプロセスをトリガする。

【０１８９】

CQLEngineのトラッキング

いくつかの実施形態において、ＣＱＬエンジントラッカーによる以下のトラッキング情報は、CQLEngineごとに維持することができる。

```
state: CQLEngineState
scheduleLocation : TaskLocation
runningExecutor : ExecutorCacheTaskLocation
name : String
endpoint : RpcEndpointRef
errorInfo : CQLEngineErrorInfo
```

「state」は、CQLEngineの状態を保つ。これは、CQLEngineのライフサイクルを通して、INACTIVE -〉 SCHEDULED -〉 ACTIVE -〉 INACTIVEのように変化する。INACTIVEは、CQLEngineTrackerによってCQLEngineがトラッキングされる前の初期状態である。SCHEDULEDは、CQLEngineをエグゼキュータにおいて実行することがスケジューリングされたときの状態である。ACTIVEは、CQLEngineが実際にエグゼキュータから実行されているときの状態である。

【０１９０】

「scheduledLocation」は、CQLEngineを実行するとスケジューリングされた場所をキープする。

【０１９１】

「runningExecutor」は、CQLEngineが実際に実行されるエグゼキュータの場所をキープする。

【０１９２】
「name」は、識別子としてのCQLEngineの名称である。
「endpoint」は、通信対象のRPCEndpointである。
【０１９３】
「errorInfo」は、最後に知られたCQLEngineのエラー情報である。
CQLEngineのリカバリ
長時間実行タスクは、CQLEngineがクラッシュしたときまたはその他の障害の場合にのみ、コントロールをCQLEngineTrackerにリターンする。CQLEngineTrackerは、CQLEngine
を再始動するために、以下のCQLEngine再始動スケジューリングプロセスを用いる。再始動スケジューリングプロセスは、リジェクトおよびクラッシュという２つのケースから呼び出される。
【０１９４】
クラッシュは、実行されているCQLEngineがクラッシュしたときまたは長時間実行タスクが何らかの故障でリターンしたときのケースである。リジェクトは、スケジューリングされた場所と実際の場所とが異なっているときのケースである（たとえば、スケジューリングされたエグゼキュータから始動されずTaskSchedulerによって異なるエグゼキュータから始動される）。これは、クラスタからのリソースの問題を原因として起こり得る。
【０１９５】
ある実施形態において、CQLEngine再始動スケジューリングプロセスは以下のように説明できる。
１．古いスケジューリングされたエグゼキュータ（アクティブでないものを除く）とクラスタ内の新たなエグゼキュータを用いて候補エグゼキュータのリストを取得
２．候補エグゼキュータのリストのにおいてまだ生きているエグゼキュータを選択
３．スケジューリングされたエグゼキュータを用いてCQLEngineを始動する長時間実行タスクを起動
局所性アフィニティアルゴリズム
水平スケーラビリティをサポートするために、入力データセットはパーティションされパラレル化分散データ処理によって処理される。CQLEngineは、CQLEngineに対する（queryId、partitionId）の対応付けまたはアフィニティを用いて複数のパーティションを処理することができる。エグゼキュータ間でデータを送信するためにネットワークトラフィックを最小にして変換を最適化するために、このアフィニティを局所性を最大にして生成する必要がある。局所性を最大にするために、一実施形態において、CQLEngineTrackerは以下のアフィニティアルゴリズムを使用する。
１．すべてのCQLEngineを、ドライバからのCQLEngineTrackerによって起動する。好ましい場所に対するCQLEngineの対応付けは設定されない。
２．最初のCQLRDDは好ましい場所の情報を有しない。
３．Spark（登録商標）スケジューラは、親ＲＤＤを親ＲＤＤの好ましい場所を用いて配置できるエグゼキュータに、ともに位置しようと試みる。
４．CQLRDDは、CQLEngineTrackerに対して「getCQLEngine」ＲＰＣを呼び出す。
５．CQLRDDのパーティションの最初の計算は、（partitionId, queryId）をCQLRDDの同一のエグゼキュータに対するCQLEngineに対応付ける。
６．CQLEngineに対する（PartitionId, queryId）の好ましい場所のマッピングをCQLEngineTrackerにおいて維持する。
７．対応付けからのCQLEngineはCQLRDDにリターンし、ＲＤＤをCQLEngineによって処理する。
８．次のCQLRDDは、ステップ５で設定された対応付け情報から好ましい位置情報を設定する。
９．Spark（登録商標）スケジューラは、設定した好ましい場所にCQLRDDを走らせよう

と
試みる。
１０．CQLRDDはCQLEngineTrackerに対し「getCQLEngine」ＲＰＣを呼び出し、（partitionId, queryId）は既に同一のエグゼキュータであるはずである。
【０１９６】
図２３は、本開示の実施形態に係る、マイクロバッチベースのイベント処理システムにおいて複数のＣＥＰエンジンをスケジューリングし管理するための一組の動作を説明するプロセス２３００のフロー図の一例である。ある実施形態において、プロセス２３００は、図２２に記載の動作のその他の詳細を説明する。特定の例において、プロセス２３００は、ＣＱＬエンジンのクラスタにおける第１のＣＱＬエンジンを起動することによって開始される。第１のＣＱＬエンジンおよびその他のＣＱＬエンジンは、ＣＱＬエンジントラッキングエンジンを用いて起動することができる。２３０４において、ＣＱＬエンジントラッキングエンジンはまた、アプリケーションに関連する入力イベントの連続ストリームのバッチを処理するために第１のＣＱＬエンジンをスケジューリングすることができる。２３０６において、ＣＱＬエンジントラッキングエンジンはまた、実行のためにスケジューリングすべき第１のＣＱＬエンジンをトラッキングすることができる。２３０８において、ＣＱＬエンジントラッキングエンジンはまた、第１のＣＱＬエンジンを実行して入力イベントの連続ストリームのバッチを処理することにより、アプリケーションに関連する出力イベントのセットを生成することができる。
【０１９７】
<u>Group Byおよびオブジェクト ＩＤフィールドを用いた自動データパーティショニングおよびパラレル処理</u>
近年、場合によっては無制限であるリアルタイムデータストリームにわたって連続的にクエリを実行できるデータストリーム管理システム（ＤＳＭ）が開発された。たとえば、典型的なＤＳＭは、１つ以上のデータストリームを受信し、データストリームに対するクエリを登録し、ストリーム内に新たなデータが現れるとクエリを連続的に実行することができる。この種の連続クエリは実行時間が長いので、ＤＳＭは、更新された結果の連続ストリームをクライアントに提供することができる。
【０１９８】
ＤＳＭにおける典型的なアプリケーションは、動作または変換の有向非巡回グラフ（ＤＡＧ）の形態の「トポロジ」として設計される。このトポロジはデータ変換パイプラインとして作用する。
【０１９９】
Apache Storm、Spark Streaming、およびFlinkを含む多くのストリーム処理システムは、アプリケーション開発者が、Java、Scala、またはClojure等の異なるプログラミング言語を用いてトポロジを構築するための、アプリケーションプログラミングインターフェイス（Application Programming Interface）（ＡＰＩ）を提供する。
【０２００】
ＡＰＩは、プログラマーがストリーム処理アプリケーションを構築するのに公的であるが、コード生成レイヤが複雑なので、Stream Analytics等のユーザのためのストリーム処理アプリケーションを生成するコード生成システムにとっては比較的複雑である。
【０２０１】
図２４は、一例としてのアーキテクチャ２４００であり、このアーキテクチャにおいて、データ変換の入力パイプラインを、パイプラインアナライザ２４０２に入力し段分類モジュール２４０４によって分類することができる。いくつかの例において、Stream Analyticsのコード生成レイヤは、パイプライン段を解析することによって自動的にデータ変換パイプラインにおけるパラレル処理を判断するという役割を担う。データ変換パイプラインは各種段で構成され、各段は段の定義に従って特定の変換を実行する。アグリゲータ段は、入来するストリームデータに対するリアルタイムアグリゲートを計算する。データ変換パイプライン処理は、パイプラインの段をノードのクラスタ上で実行できる場合、最適

化することができる。
【０２０２】
ノードのクラスタ上の段を計算するためには、自動的に段の動作のパラレル処理の特徴を判断しその後変換のＤＡＧを生成することが望ましい。変換の計算は、一組のクラスタノードについて、パラレル処理を最大にすることによって完了することができる。
【０２０３】
一実施形態において、ユーザが設計したデータ変換パイプラインを解析するデータストリーム管理システム（Data Stream Management System）（ＤＳＭＳ）が構築され、各種
段のパーティショニング基準を取り出し、変換の最適化されたＤＡＧを生成する。各段は一組のクラスタノード上で実行することができる。
【０２０４】
特定の実施形態において、以下の段を、Stream Analyticsプラットフォームによって設計されたパイプラインに含めることができる。
１）クエリ
２）ビジネスルール
３）空間
４）パターン
パイプラインは、上記タイプの１つ以上の段で構成することができる。
【０２０５】
サンプルパイプラインの一例は以下の通りである。
入力 -〉 クエリ -〉 クエリ-〉 空間 -〉 出力
いくつかの例において、ユーザはパイプラインを作製することにより所望のビジネスロジックを得る。パイプラインを設計している間、ユーザは、パイプラインに対してすべての段を選択しパイプラインの段属性を構成する場合がある。いくつかの例において、段の構成属性は段のメタデータとなる。
【０２０６】
開示する技術は、段の種類がクエリである場合、段の自動データパーティショニングを判断する。ユーザが作成したパイプラインをネイティブランタイム変換のＤＡＧに変換するために、Stream Analyticsプラットフォームは以下のステップを実行することができる。
１）ソースからシンクまでパイプラインをトラバース
２）各段について、
ｉ）段の種類を判断
ｉｉ）段の種類が「クエリ」であれば、プラットフォームは、この段について変換を分散方式で計算できるか否かをマークする
ａ）クエリ段に対応付けられたＣＱＬクエリを判断
ｂ）ＣＱＬクエリをトークンにパース
ｃ）パースされたクエリのＣＱＬクエリの意味解析を実行
ｄ）各種ルールを用いてクエリ分類を判断
これらのルールは、連続クエリを、ステートレス、セミステートフル、フルステートフルというカテゴリに、分類
ｅ）クエリがステートレスであれば、パーティショニング属性（基準）なしで、パーティショニングすべき段をマーク２４０６。このようにして、段は親の段のパーティショニング基準に依存する。
【０２０７】
ｆ）クエリがステートフルであれば、パーティションしない段をマーク２４０８。このようにして、段は、単一のノードクラスタ上でのみ、実行できる。
【０２０８】
ｇ）クエリがセミステートフルであれば、パーティショニング属性を用いてパーティシ

ョンすべき段をマーク２４１０。パーティショニング属性は、ステップ２．ｉｉ．ｄの結果から取得する。このようにして、段の計算を、自動的に判断したパーティショニング属性でパーティションすることができる。
３）各段についげ、データ変換パイプラインのＤＡＧにおける変換を生成する。
【０２０９】
ｉ）段が、パーティショニング属性なしでパーティションされたとマークされた場合２４０６、ＤＡＧにおけるこの段に対する変換を、再パーティショニング変換なしで生成する。段のパーティションの数は、前の段のパーティションの数から求められる。
【０２１０】
ｉｉ）段が、パーティショニング属性でパーティションされたト判断された場合２４１０、再パーティショニング変換を用いてＤＡＧにおけるこの段の変換を生成する。再パーティショニング変換に対する入力は、パーティショニング属性およびパーティションの数であろう。再パーティショニング変換は、入来するイベントストリームを、新たなパーティショニング基準で再度パーティションする。
【０２１１】
ｉｉｉ）段がパーティションされていないとマークされる場合２４０８、段の変換が後に続く再パーティショニング変換でＤＡＧ内の段に対する変換を生成する。再パーティショニング変換に対する入力は、パーティショニング属性およびパーティションの数であり、これは１であろう。再パーティショニング変換は、既にパーティションされている／パーティションされていないストリームを１つのパーティションに再度パーティションする。
【０２１２】
特定のパイプラインにおいて、システムが十分なメタデータを持っていない場合、または、クエリ解析によってパーティショニングを判断できない場合、システムは、オブジェクトｉｄをパーティショニング属性としてマークする２４１２。
【０２１３】
段の種類が空間でありＤＳＭＳが移動するオブジェクトのジオロケーション（geo-location）イベントを処理しており各オブジェクトが固有のアイデンティティを有する場合、Stream Analyticsプラットフォームは、オブジェクトｉｄを段のパーティショニング属性としてマークし、ＤＡＧのこの段の変換を、パーティショニング変換、次に段変換によって、マークする。
【０２１４】
開示される技術の実施形態は以下の特徴を提供する。
－パイプライン段のメタデータスキャンを実行しＣＱＬクエリ分類に基づいて段を分類する。
－連続クエリ言語に基づいてクエリの意味解析を実行することによりパーティショニング属性を自動的に判断する。
－パーティショニングを用いて変換のＤＡＧを生成する。
【０２１５】
以前の技術は、ユーザがパイプライン段のパラレル処理の特徴を明確に基底するために、パイプラインデータ変換を使用することを含んでいた。指定されなければ、システムは、計算リソース全体を使用することなくパイプライン段を処理できる。
【０２１６】
開示される技術は、パイプライン化されたストリーム処理システムの段を解析することにより、データパーティショニングの基準を判断する。これにより、Stream Analyticsプラットフォームのデータ処理パイプラインの設計の複雑度が大幅に減じられる。
【０２１７】
図２５は、分散イベント処理システムにおけるデータパーティショニングおよびパラレル処理のための一組の動作を説明するプロセス２５００のフロー図の一例である。ある実施形態において、プロセス２５００は、図２４に記載の動作のその他の詳細を説明する。特定の例において、プロセス２５００は、２５０２において、イベント処理システムによ

って処理されている連続クエリ言語（ＣＱＬ）クエリの段を判断することによって開始される。２５０４に進んで、システムは、段に対応付けられた段の種類をハンドなするように構成し得る。いくつかの例において、プロセス２５００は、２５０６において、段の種類に少なくとも一部基づいて段に対して計算すべき変換を判断する。プロセス２５００はまた、２５０８において、複数のルールに少なくとも一部基づいて、ＣＱＬクエリの分類を判断することを含み得る。いくつかの例において、プロセス２５００は、２５１０において、パーティショニング基準を段に適用することにより、パーティションされた段またはパーティションされていない段として段をマークすることを含み得る。加えて、いくつかの例において、プロセス２５００は、２５１２において、段のパーティショニング基準に少なくとも一部基づいて、段のデータ変換パイプラインの有向非巡回グラフ（ＤＡＧ）における変換を生成してもよい。２５１４において、プロセス２５００は、変換に少なくとも一部基づいて段のパーティショニングを判断してもよい。プロセス２５００はまた、判断したパーティショニングに少なくとも一部基づいてＣＱＬクエリを処理してもよい。

【０２１８】

例示的システム

図２６～図１２は、さまざまな実施形態による本開示の態様を実施するための例示的環境の態様を示す。図２６は、本開示の一実施形態を実施するための分散型システム２６００の簡略図を示す。図示の実施形態では、分散型システム２６００は、１つ以上のネットワーク２６１０でウェブブラウザ、所有権を主張できるクライアント（たとえばOracle Forms）などのクライアントアプリケーションを実行し動作させるように構成された１つ以上のクライアントコンピューティングデバイス２６０２，２６０４，２６０６および２６０８を含む。サーバ２６１２は、ネットワーク２６１０を介してリモートクライアントコンピューティングデバイス２６０２，２６０４，２６０６および２６０８と通信可能に結合されてもよい。

【０２１９】

さまざまな実施形態では、サーバ２６１２は、イベント処理サービスを提供するサービスおよびアプリケーションなどの１つ以上のサービスまたはソフトウェアアプリケーションを実行するように適合されてもよい。特定の実施形態では、サーバ２６１２は、非仮想および仮想環境を含むことができる他のサービスまたはソフトウェアアプリケーションも提供できる。いくつかの実施形態では、これらのサービスは、ウェブベースのサービスもしくはクラウドサービスとして、またはソフトウェア・アズ・ア・サービス（ＳａａＳ）モデルの下で、クライアントコンピューティングデバイス２６０２，２６０４，２６０６および/または２６０８のユーザに対して提供されてもよい。クライアントコンピューティングデバイス２６０２，２６０４，２６０６および/または２６０８を動作させるユーザは、次いで、１つ以上のクライアントアプリケーションを利用してサーバ２６１２と対話して、これらのコンポーネントによって提供されるサービスを利用してもよい。

【０２２０】

図２６に示される構成では、システム２６００のソフトウェアコンポーネント２６１８，２６２０および２６２２は、サーバ２６１２上で実現されるものとして示されている。他の実施形態では、システム２６００のコンポーネントのうちの１つ以上および／またはこれらのコンポーネントによって提供されるサービスは、クライアントコンピューティングデバイス２６０２，２６０４，２６０６および／または２６０８のうちの１つ以上によって実現されてもよい。クライアントコンピューティングデバイスを動作させるユーザは、次いで、１つ以上のクライアントアプリケーションを利用して、これらのコンポーネントによって提供されるサービスを用いてもよい。これらのコンポーネントはハードウェア、ファームウェア、ソフトウェア、またはそれらの組合せにおいて実現されてもよい。分散型システム２６００とは異なってもよいさまざまな異なるシステム構成が可能であることが理解されるべきである。図２６に示される実施形態は、したがって、実施形態のシステムを実現するための分散型システムの一例であり、限定的であるよう意図されるものではない。

【０２２１】
クライアントコンピューティングデバイス２６０２，２６０４，２６０６、および／または２６０８は、さまざまなタイプのコンピューティングシステムを含むことができる。たとえば、クライアントデバイスは、携帯可能な手持ち式のデバイス（たとえば、ｉＰｈｏｎｅ（登録商標）、セルラー電話、ｉＰａｄ（登録商標）、コンピューティングタブレット、携帯情報端末（ＰＤＡ））またはウェアラブルデバイス（たとえばＧｏｏｇｌｅ Ｇｌａｓｓ（登録商標）頭部装着型ディスプレイ）であってもよく、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ（登録商標） Ｍｏｂｉｌｅ（登録商標）などのソフトウェア、および／もしくは、ｉＯＳ、Ｗｉｎｄｏｗｓ Ｐｈｏｎｅ、Ａｎｄｒｏｉｄ、ＢｌａｃｋＢｅｒｒｙ２６、Ｐａｌｍ ＯＳなどのさまざまなモバイルオペレーティングシステムを動作させる。クライアントデバイスは、インターネット関連アプリケーション、電子メール、ショートメッセージサービス（ＳＭＳ）アプリケーションのようなさまざまなアプリケーションをサポートしてもよく、さまざまな他の通信プロトコルを使用してもよい。クライアントコンピューティングデバイスは、汎用パーソナルコンピュータであってもよく、一例として、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ（登録商標）、Ａｐｐｌｅ Ｍａｃｉｎｔｏｓｈ（登録商標）および／またはＬｉｎｕｘ（登録商標）オペレーティングシステムのさまざまなバージョンを実行するパーソナルコンピュータおよび／またはラップトップコンピュータも含んでもよい。クライアントコンピューティングデバイスは、たとえばＧｏｏｇｌｅ Ｃｈｒｏｍｅ ＯＳなどのさまざまなＧＮＵ／Ｌｉｎｕｘオペレーティングシステムを限定を伴うことなく含む、さまざまな市場で入手可能なＵＮＩＸ（登録商標）またはＵＮＩＸのようなオペレーティングシステムのいずれかを実行するワークステーションコンピュータであり得る。クライアントコンピューティングデバイスは、また、ネットワーク２６１０を介して通信することができる、シンクライアントコンピュータ、インターネットにより可能化されるゲームシステム（たとえば、Ｋｉｎｅｃｔ（登録商標）ジェスチャ入力デバイスを伴うかまたは伴わないＭｉｃｒｏｓｏｆｔ Ｘｂｏｘゲームコンソールおよび／または個人メッセージ伝達デバイスなどの電子デバイスを含んでもよい。

【０２２２】
図２６の分散型システム２６００は、４つのクライアントコンピューティングデバイスとともに示されているが、任意の数のクライアントコンピューティングデバイスがサポートされてもよい。センサを伴うデバイスなど、他のデバイスがサーバ２６１２と対話してもよい。

【０２２３】
分散型システム２６００におけるネットワーク２６１０は、ＴＣＰ／ＩＰ（伝送制御プロトコル／インターネットプロトコル）、ＳＮＡ（システムネットワークアーキテクチャ）、ＩＰＸ（インターネットパケット交換）、ＡｐｐｌｅＴａｌｋなどを限定を伴うことなく含む、さまざまな入手可能なプロトコルのうちのいずれかを用いてデータ通信をサポートすることができる、当業者が精通している任意のタイプのネットワークであってもよい。単なる例として、ネットワーク２６１０は、ローカルエリアネットワーク（ＬＡＮ）、イーサネット（登録商標）に基づくネットワーク、トークンリング、ワイドエリアネットワーク、インターネット（登録商標）、仮想ネットワーク、仮想プライベートネットワーク（ＶＰＮ）、イントラネット、エクストラネット、公衆交換電話網（ＰＳＴＮ）、赤外線ネットワーク、無線ネットワーク（たとえば電気電子学会（ＩＥＥＥ）２６０２．１１プロトコルのいずれかの下で動作するネットワーク、ブルートゥース（登録商標）および／もしくは任意の他の無線プロトコル）、ならびに／またはこれらおよび／もしくは他のネットワークの任意の組み合わせを含むことができる。

【０２２４】
サーバ２６１２は、１つ以上の汎用コンピュータ、専用のサーバコンピュータ（一例としてＰＣ（パーソナルコンピュータ）サーバ、ＵＮＩＸ（登録商標）サーバ、ミッドレンジサーバ、メインフレームコンピュータ、ラックマウント型サーバなどを含む）、サーバファーム、サーバクラスタ、またはその他の適切な構成および／または組み合わせで構成

されてもよい。サーバ２６１２は、仮想オペレーティングシステムを実行する１つ以上の仮想マシン、または仮想化を伴う他のコンピューティングアーキテクチャを含み得る。論理ストレージデバイスの１つ以上の柔軟なプールを仮想化してサーバのために仮想ストレージデバイスを維持することができる。仮想ネットワークを、サーバ２６１２によって、ソフトウェア定義のネットワーク接続を用いて制御することができる。さまざまな実施形態において、サーバ２６１２は、前述の開示に記載される１つ以上のサービスまたはソフトウェアアプリケーションを実行するように適合されてもよい。たとえば、サーバ２６１２は、本開示の実施形態に従って上記の処理を実行するためのサーバに対応してもよい。

【０２２５】

サーバ２６１２は、上記のもののうちのいずれかを含むオペレーティングシステム、および任意の市場で入手可能なサーバオペレーティングシステムを実行してもよい。サーバ２６１２は、ＨＴＴＰ（ハイパーテキスト転送プロトコル）サーバ、ＦＴＰ（ファイル転送プロトコル）サーバ、ＣＧＩ（コモンゲートウェイインターフェイス）サーバ、ＪＡＶＡ（登録商標）サーバ、データベースサーバなどを含むさまざまなさらに他のサーバアプリケーションおよび／または中間層アプリケーションのうちのいずれかも実行してもよい。例示的なデータベースサーバは、オラクル、マイクロソフト、サイベース、ＩＢＭ（インターナショナルビジネスマシンズ）などから市場で入手可能なものを含むが、それらに限定されるものではない。

【０２２６】

いくつかの実現例では、サーバ２６１２は、クライアントコンピューティングデバイス２６０２，２６０４，２６０６および２６０８のユーザから受信されるデータフィードおよび／またはイベント更新情報を解析および整理統合するための１つ以上のアプリケーションを含んでもよい。一例として、データフィードおよび／またはイベント更新情報は、センサデータアプリケーション、金融株式相場表示板、ネットワーク性能測定ツール（たとえば、ネットワークモニタリングおよびトラフィック管理アプリケーション）、クリックストリーム解析ツール、自動車交通モニタリングなどに関連するリアルタイムのイベントを含んでもよい、１つ以上の第三者情報源および連続データストリームから受信される、Ｔｗｉｔｔｅｒ（登録商標）フィード、Ｆａｃｅｂｏｏｋ（登録商標）更新情報またはリアルタイムの更新情報を含んでもよいが、それらに限定されるものではない。サーバ２６１２は、データフィードおよび／またはリアルタイムのイベントをクライアントコンピューティングデバイス２６０２，２６０４，２６０６および２６０８の１つ以上の表示デバイスを介して表示するための１つ以上のアプリケーションも含んでもよい。

【０２２７】

分散型システム２６００は、１つ以上のデータベース２６１４および２６１６も含んでもよい。これらのデータベースは、ユーザ識別情報、および本開示の実施形態によって使用される他の情報などの情報を格納するためのメカニズムを提供することができる。データベース２６１４および２６１６は、さまざまな位置にあってもよい。一例として、データベース２６１４および２６１６のうちの１つ以上は、サーバ２６１２に局在する（および／またはサーバ２６１２に常駐する）非一時的な記憶媒体にあってもよい。代替的に、データベース２６１４および２６１６は、サーバ２６１２から遠隔にあり、ネットワークベースまたは専用の接続を介してサーバ２６１２と通信してもよい。一組の実施形態では、データベース２６１４および２６１６は、記憶域ネットワーク（ＳＡＮ）にあってもよい。同様に、サーバ２６１２に帰する機能を実行するための任意の必要なファイルが、適宜、サーバ２６１２上においてローカルに、および／または遠隔で格納されてもよい。一組の実施形態では、データベース２６１４および２６１６は、ＳＱＬフォーマットされたコマンドに応答してデータを格納、更新および検索取得するように適合される、オラクルによって提供されるデータベースなどのリレーショナルデータベースを含んでもよい。

【０２２８】

いくつかの図面に示されたシステムは、さまざまな構成で提供されてもよい。いくつかの実施形態では、システムは、システムの１つ以上のコンポーネントが１つ以上のクラウ

ドインフラストラクチャシステムの１つ以上のネットワークに分散された分散型システムとして構成することができる。

【０２２９】

クラウドインフラストラクチャシステムは、１つ以上のサーバコンピューティングデバイス、ネットワークデバイス、および／またはストレージデバイスの集合である。これらのリソースは、クラウドサービスプロバイダによって分割され、何らかの方法で顧客に割り当てられてもよい。たとえば、カリフォルニア州レッドウッドショアーズのオラクルコーポレーションのようなクラウドサービスプロバイダは、さまざまなタイプのクラウドサービスを提供し得、それらは、サービスとしてのソフトウェア（ＳａａＳ）カテゴリで提供される１つ以上のサービス、サービスとしてのプラットフォーム（ＰａａＳ）カテゴリで提供されるサービス、サービスとしてのインフラストラクチャ（ＩａａＳ）カテゴリで提供されるサービス、またはハイブリッドサービスを含む他のカテゴリのサービスを含むが、それらに限定はされない。ＳａａＳサービスの例としては、Oracle Fusionアプリケーションなどのオンデマンドアプリケーションの一式を構築して提供する機能があるが、これに限定されない。ＳａａＳサービスは、顧客が、クラウドインフラストラクチャシステム上で実行されているアプリケーションのためのソフトウェアを購入する必要なく、そのようなアプリケーションを利用することを可能にする。ＰａａＳサービスの例としては、組織（オラクルなど）が共有された共通アーキテクチャー上で既存のアプリケーションを統合できるようにするサービスや、Oracle Java Cloud Service (JCS), Oracle Database Cloud Service (DBCS)などのプラットフォームによって提供される共有サービスを活
用する新たなアプリケーションを構築する能力を含むが、それらに限定はされない。ＩａａＳサービスは、通常、ＳａａＳプラットフォームおよびＰａａＳプラットフォームによって提供されるサービスを利用する顧客のためのストレージ、ネットワーク、およびその他の基本的な計算資源などの基盤となる計算資源の管理および制御を容易にする。

【０２３０】

図２７は、本開示の実施形態に従って、実施形態のシステムの１つ以上のコンポーネントによって提供されるサービスがクラウドサービスとして提供されてもよいシステム環境２７００の１つ以上のコンポーネントの簡略ブロック図である。示される実施形態において、システム環境２７００は、クラウドサービスを提供するクラウドインフラストラクチャシステム２７０２と対話するようユーザによって用いられてもよい１つ以上のクライアントコンピューティングデバイス２７０４，２７０６および２７０８を含む。クライアントコンピューティングデバイスは、クラウドインフラストラクチャシステム２７０２と対話して、クラウドインフラストラクチャシステム２７０２によって提供されるサービスを用いるよう、クライアントコンピューティングデバイスのユーザによって用いられてもよい、ウェブブラウザ、所有権付きクライアントアプリケーション（たとえばOracle Forms）または何らかの他のアプリケーションなどのようなクライアントアプリケーションを動作させるよう構成されてもよい。

【０２３１】

図に示されるクラウドインフラストラクチャシステム２７０２は図示されるもの以外のコンポーネントを有してもよいことが理解されるべきである。さらに、図に示される実施形態は、本開示の実施形態を組み込んでもよいクラウドインフラストラクチャシステムの一例に過ぎない。いくつかの他の実施形態において、クラウドインフラストラクチャシステム２７０２は、図に示されるよりも多いかもしくは少ないコンポーネントを有してもよく、２つ以上のコンポーネントを組み合わせてもよく、またはコンポーネントの異なる構成もしくは配置を有してもよい。

【０２３２】

クライアントコンピューティングデバイス２７０４，２７０６および２７０８は、５０２，５０４，５０６および５０８に対して上記されたものと同様のデバイスであってもよい。

【０２３３】
例示的なシステム環境２７００は３つのクライアントコンピューティングデバイスとともに示されているが、任意の数のクライアントコンピューティングデバイスがサポートされてもよい。センサを伴うデバイスなどのような他のデバイスがクラウドインフラストラクチャシステム２７０２と対話してもよい。
【０２３４】
ネットワーク２７１０はクライアント２７０４，２７０６および２７０８とクラウドインフラストラクチャシステム２７０２との間におけるデータの通信および交換を容易にし得る。各ネットワークは、ネットワーク２７１０に対して上記されたものを含む、さまざまな市場で入手可能なプロトコルの任意のものを用いてデータ通信をサポートすることができる、当業者が精通している任意のタイプのネットワークであってもよい。
【０２３５】
クラウドインフラストラクチャシステム２７０２は、サーバ２７１２に対して上記されたものを含んでもよい１つ以上のコンピュータおよび／またはサーバを備えてもよい。
【０２３６】
特定の実施形態において、クラウドインフラストラクチャシステムによって提供されるサービスは、オンラインデータストレージおよびバックアップソリューション、ウェブに基づくｅメールサービス、運営管理されるオフィススイートおよびドキュメントコラボレーションサービス、データベース処理、管理される技術サポートサービスなどのような、オンデマンドでクラウドインフラストラクチャシステムのユーザに利用可能にされるサービスのホストを含んでもよい。クラウドインフラストラクチャシステムによって提供されるサービスは動的にスケーリングしてそのユーザのニーズを満たすことができる。クラウドインフラストラクチャシステムによって提供されるあるサービスのある具体的なインスタンス化は本明細書では「サービスインスタンス」と呼ばれる。一般的に、クラウドサービスプロバイダのシステムからインターネットなどのような通信ネットワークを介してユーザに利用可能にされる任意のサービスは「クラウドサービス」と呼ばれる。典型的には、パブリックなクラウド環境においては、クラウドサービスプロバイダのシステムを形成するサーバおよびシステムは顧客自身のオンプレミスのサーバおよびシステムとは異なる。たとえば、クラウドサービスプロバイダのシステムはアプリケーションを運営管理してもよく、ユーザは、インターネットなどのような通信ネットワークを介して、オンデマンドで、アプリケーションをオーダーし使用してもよい。
【０２３７】
いくつかの例では、コンピュータネットワーククラウドインフラストラクチャにおけるサービスは、ストレージ、運営管理されるデータベース、運営管理されるウェブサーバ、ソフトウェアアプリケーション、またはクラウドベンダーによってユーザに提供されるかもしくは他の態様で当該技術分野において公知であるような他のサービスに対する保護されたコンピュータネットワークアクセスを含んでもよい。たとえば、サービスは、クラウド上の遠隔ストレージに対するインターネットを介してのパスワード保護されたアクセスを含むことができる。別の例として、サービスは、ネットワーク接続された開発者による個人的な使用のために、ウェブサービスに基づく運営管理されたリレーショナルデータベースおよびスクリプト言語ミドルウェアエンジンを含むことができる。別の例として、サービスは、クラウドベンダーのウェブサイトにおいて運営管理されるｅメールソフトウェアアプリケーションに対するアクセスを含むことができる。
【０２３８】
特定の実施形態において、クラウドインフラストラクチャシステム２７０２は、セルフサービスの、サブスクリプションに基づく、順応性を持ってスケーラブルで、信頼性があり、非常に利用可能性があり、およびセキュリティ上安全な態様で顧客に対して配送されるアプリケーションの組、ミドルウェア、データベースサービス提供を含んでもよい。そのようなクラウドインフラストラクチャシステムの一例は本譲受人によって提供されるＯｒａｃｌｅ　Ｐｕｂｌｉｃ　Ｃｌｏｕｄである。

【０２３９】
さまざまな実施形態において、クラウドインフラストラクチャシステム２７０２は、クラウドインフラストラクチャシステム２７０２によって提供されるサービスに対する顧客のサブスクリプションを自動的にプロビジョニング、管理、およびトラッキングするように構成されてもよい。クラウドインフラストラクチャシステム２７０２はクラウドサービスを異なる開発モデルを介して提供してもよい。たとえば、サービスは、クラウドインフラストラクチャシステム２７０２が（たとえばＯｒａｃｌｅによって所有される）クラウドサービスを販売する組織によって所有され、サービスが一般大衆または異なる業界エンタープライズに対して利用可能にされるパブリッククラウドモデルの下で提供されてもよい。別の例として、サービスは、クラウドインフラストラクチャシステム１１０２が単一の組織に対してのみ動作され、その組織内における１つ以上のエンティティに対してサービスを提供してもよい、プライベートクラウドモデルの下で提供されてもよい。クラウドサービスは、さらに、クラウドインフラストラクチャシステム２７０２およびクラウドインフラストラクチャシステム２７０２によって提供されるサービスが関係付けられるコミュニティにおけるいくつかの組織によって共有されるコミュニティクラウドモデルの下で提供されてもよい。クラウドサービスは、さらに、２つ以上の異なるモデルの組合せであるハイブリッドクラウドモデルの下で提供されてもよい。

【０２４０】
いくつかの実施形態において、クラウドインフラストラクチャシステム２７０２によって提供されるサービスは、Ｓｏｆｔｗａｒｅ ａｓ ａ Ｓｅｒｖｉｃｅ（ＳａａＳ）カテゴリ、Ｐｌａｔｆｏｒｍ ａｓ ａ Ｓｅｒｖｉｃｅ（ＰａａＳ）カテゴリ、Ｉｎｆｒａｓｔｒｕｃｔｕｒｅ ａｓ ａ Ｓｅｒｖｉｃｅ（ＩａａＳ）カテゴリ、またはハイブリッドサービスを含む他のサービスのカテゴリの下で提供される、１つ以上のサービスを含んでもよい。顧客は、サブスクリプションオーダーを介して、クラウドインフラストラクチャシステム２７０２によって提供される１つ以上のサービスをオーダーしてもよい。クラウドインフラストラクチャシステム２７０２は、次いで、処理を実行して、顧客のサブスクリプションオーダーにおけるサービスを提供する。

【０２４１】
いくつかの実施形態において、クラウドインフラストラクチャシステム２７０２によって提供されるサービスは、限定を伴わずに、アプリケーションサービス、プラットフォームサービスおよびインフラストラクチャサービスを含んでもよい。いくつかの例では、アプリケーションサービスはクラウドインフラストラクチャシステムによってＳａａＳプラットフォームを介して提供されてもよい。ＳａａＳプラットフォームは、ＳａａＳカテゴリに入るクラウドサービスを提供するよう構成されてもよい。たとえば、ＳａａＳプラットフォームは、一式のオンデマンドアプリケーションを統合された開発およびデプロイプラットフォーム上で構築し配送する能力を提供してもよい。ＳａａＳプラットフォームは、ＳａａＳサービスを提供するための基底のソフトウェアおよびインフラストラクチャを管理および制御してもよい。ＳａａＳプラットフォームによって提供されるサービスを利用することによって、顧客はクラウドインフラストラクチャシステムにおいて実行されるアプリケーションを利用することができる。顧客は、別のライセンスおよびサポートを購入する必要なくアプリケーションサービスを獲得することができる。さまざまな異なるＳａａＳサービスが提供されてもよい。その例は、限定を伴うことなく、大きな組織に対する売上実績管理、エンタープライズ統合、および事業柔軟性に対するソリューションを提供するサービスを含む。

【０２４２】
いくつかの実施形態において、プラットフォームサービスはクラウドインフラストラクチャシステムによってＰａａＳプラットフォームを介して提供されてもよい。ＰａａＳプラットフォームはＰａａＳカテゴリの下におけるクラウドサービスを提供するよう構成されてもよい。プラットフォームサービスの例は、限定を伴わずに、（Ｏｒａｃｌｅなどのような）組織が既存のアプリケーションを共有される共通のアーキテクチャにおいて整理

統合することができるサービス、およびプラットフォームによって提供される共有されるサービスをてこ入れする新たなアプリケーションを構築する能力を含んでもよい。ＰａａＳプラットフォームはＰａａＳサービスを提供するための基底のソフトウェアおよびインフラストラクチャを管理および制御してもよい。顧客は、クラウドインフラストラクチャシステムによって提供されるＰａａＳサービスを、別のライセンスおよびサポートを購入する必要なく獲得することができる。プラットフォームサービスの例は、限定を伴わずに、Ｏｒａｃｌｅ Ｊａｖａ Ｃｌｏｕｄ Ｓｅｒｖｉｃｅ（ＪＣＳ）、Ｏｒａｃｌｅ Ｄａｔａｂａｓｅ Ｃｌｏｕｄ Ｓｅｒｖｉｃｅ（ＤＢＣＳ）などを含む。

【０２４３】

ＰａａＳプラットフォームによって提供されるサービスを利用することによって、顧客は、クラウドインフラストラクチャシステムによってサポートされるプログラミング言語およびツールを使用することができ、デプロイされたサービスを制御することもできる。いくつかの実施形態において、クラウドインフラストラクチャシステムによって提供されるプラットフォームサービスは、データベースクラウドサービス、ミドルウェアクラウドサービス（たとえばＯｒａｃｌｅ Ｆｕｓｉｏｎ（登録商標） Ｍｉｄｄｌｅｗａｒｅサービス）、およびＪａｖａクラウドサービスを含んでもよい。一実施形態において、データベースクラウドサービスは、組織がデータベース資源をプールし、顧客にＤａｔａｂａｓｅ ａｓ ａ Ｓｅｒｖｉｃｅをデータベースクラウドの形式で提供することを可能にする、共有されるサービスデプロイモデルをサポートしてもよい。ミドルウェアクラウドサービスは、顧客のためのプラットフォームを提供してさまざまなビジネスアプリケーションを開発およびデプロイしてもよく、Ｊａｖａクラウドサービスは顧客のためのプラットフォームを提供してＪａｖａアプリケーションをクラウドインフラストラクチャシステムにおいてデプロイしてもよい。

【０２４４】

さまざまな異なるインフラストラクチャサービスがＩａａＳプラットフォームによってクラウドインフラストラクチャシステムにおいて提供されてもよい。インフラストラクチャサービスは、ＳａａＳプラットフォームおよびＰａａＳプラットフォームによって提供されるサービスを利用する顧客に対するストレージ、ネットワーク、他の基礎的計算資源などのような基底の計算資源の管理および制御を容易にする。

【０２４５】

特定の実施形態において、クラウドインフラストラクチャシステム２７０２は、さらに、クラウドインフラストラクチャシステムの顧客に対してさまざまなサービスを提供するよう用いられる資源を提供するためのインフラストラクチャ資源２７３０を含んでもよい。一実施形態において、インフラストラクチャ資源２７３０は、ＰａａＳプラットフォームおよびＳａａＳプラットフォームによって提供されるサービスを実行するよう、サーバ、ストレージおよびネットワーク接続資源などのような、ハードウェアの予め統合され最適化された組合せを含んでもよい。

【０２４６】

いくつかの実施形態において、クラウドインフラストラクチャシステム２７０２における資源は、複数のユーザによって共有され、要望に付き動的に再割当てされてもよい。加えて、資源はユーザに対して異なる時間ゾーンで割当てられてもよい。たとえば、クラウドインフラストラクチャシステム２７３０は、第１の時間ゾーンにおける第１の組のユーザがクラウドインフラストラクチャシステムの資源をある特定化された時間数の間利用することを可能にし、次いで、異なる時間ゾーンに位置する別の組のユーザに対する同じ資源の再割当てを可能にし、それによって、資源の利用を最大限にしてもよい。

【０２４７】

特定の実施形態において、クラウドインフラストラクチャシステム２７０２の異なるコンポーネントまたはモジュール、およびクラウドインフラストラクチャシステム２７０２によって提供されるサービスによって共有される、ある数の内部の共有されるサービス２７３２が提供されてもよい。これらの内部の共有されるサービスは、限定を伴うことなく

、セキュリティおよびアイデンティティサービス、統合サービス、エンタープライズリポジトリサービス、エンタープライズマネージャサービス、ウイルススキャンおよびホワイトリストサービス、高可用性、バックアップおよびリカバリサービス、クラウドサポートを可能にするためのサービス、ｅメールサービス、通知サービス、ファイル転送サービスなどを含んでもよい。

【０２４８】

特定の実施形態において、クラウドインフラストラクチャシステム２７０２は、クラウドインフラストラクチャシステムにおいてクラウドサービス（たとえばＳａａＳ、ＰａａＳ、およびＩａａＳサービス）の包括的な管理を提供してもよい。一実施形態において、クラウド管理機能は、クラウドインフラストラクチャシステム２７０２によって受信される顧客のサブスクリプションをプロビジョニングし、管理し、およびトラッキングする能力などを含んでもよい。

【０２４９】

一実施形態において、図に示されるように、クラウド管理機能は、オーダー管理モジュール２７２０、オーダーオーケストレーションモジュール２７２２、オーダープロビジョニングモジュール２７２４、オーダー管理およびモニタリングモジュール２７２６、ならびにアイデンティティ管理モジュール２７２８などのような１つ以上のモジュールによって提供されてもよい。これらのモジュールは、１つ以上のコンピュータおよび／またはサーバを含むかもしくはそれらを用いて提供されてもよく、それらは汎用コンピュータ、特殊化されたサーバコンピュータ、サーバファーム、サーバクラスタ、または任意の他の適切な構成および／もしくは組合せであってもよい。

【０２５０】

例示的な動作２７３４においては、クライアントデバイス２７０４、２７０６または２７０８などのようなクライアントデバイスを用いる顧客は、クラウドインフラストラクチャシステム２７０２によって提供される１つ以上のサービスを要求すること、およびクラウドインフラストラクチャシステム２７０２によって提供される１つ以上のサービスに対するサブスクリプションに対するオーダーを行うことによって、クラウドインフラストラクチャシステム２７０２と対話してもよい。特定の実施形態において、顧客は、クラウドユーザインターフェイス（ＵＩ）、クラウドＵＩ２７１２、クラウドＵＩ２７１４および／またはクラウドＵＩ２７１６にアクセスし、サブスクリプションオーダーをこれらのＵＩを介して行ってもよい。顧客がオーダーを行うことに応答してクラウドインフラストラクチャシステム２７０２によって受取られるオーダー情報は、顧客を識別する情報、およびクラウドインフラストラクチャシステム２７０２によって提供される、その顧客が利用することを意図する１つ以上のサービスを含んでもよい。

【０２５１】

オーダーが顧客によってなされた後、オーダー情報はクラウドＵＩ２７１２、２７１４および／または２７１６を介して受取られてもよい。

【０２５２】

動作２７３６において、オーダーはオーダーデータベース２７１８に格納される。オーダーデータベース２７１８は、クラウドインフラストラクチャシステム２７１８によって動作されるいくつかのデータベースの１つであり得、他のシステム要素との関連において動作され得る。

【０２５３】

動作２７３８で、オーダー情報はオーダー管理モジュール２７２０に転送される。いくつかの例では、オーダー管理モジュール２７２０は、オーダーを検証すること、および検証でそのオーダーを予約することなど、オーダーに関係付けられる請求および課金機能を実行するよう構成されてもよい。

【０２５４】

動作２７４０で、オーダーに関する情報がオーダーオーケストレーションモジュール２７２２に通信される。オーダーオーケストレーションモジュール２７２２は、オーダー情

報を利用して、顧客によってなされたオーダーに対してサービスおよび資源のプロビジョニングをオーケストレーションしてもよい。いくつかの例では、オーダーオーケストレーションモジュール２７２２は、資源のプロビジョニングをオーケストレーションして、利用されるサービスを、オーダープロビジョニングモジュール２７２４のサービスを用いてサポートしてもよい。

【０２５５】

特定の実施形態において、オーダーオーケストレーションモジュール２７２２は、各オーダーに関連付けられるビジネスプロセスの管理を可能にし、ビジネスロジックを適用して、オーダーがプロビジョニングに進むべきかどうかを判断する。動作２７４２で、新たなサブスクリプションに対するオーダーが受取られると、オーダーオーケストレーションモジュール２７２２は、オーダープロビジョニングモジュール２７２４に対して、資源を割当て、サブスクリプションオーダーを満たすよう必要とされる資源を構成するよう、要求を送る。オーダープロビジョニングモジュール２７２４は、顧客によってオーダーされたサービスに対する資源の割当てを可能にする。オーダープロビジョニングモジュール２７２４は、クラウドインフラストラクチャシステム２７００によって提供されるクラウドサービスと、要求されたサービスを提供するための資源をプロビジョニングするよう用いられる物理的インプリメンテーション層との間におけるある抽象レベルを与える。オーダーオーケストレーションモジュール２７２２は、したがって、サービスおよび資源がオンザフライで実際にプロビジョニングされるか、または予めプロビジョニングされ、要求でのみ割当て／分配されるかどうかなど、インプリメンテーション詳細から隔離されてもよい。

【０２５６】

動作２７４４で、サービスおよび資源がプロビジョニングされると、提供されるサービスの通知が、クラウドインフラストラクチャシステム３０２のオーダープロビジョニングモジュール２７２４によってクライアントデバイス２７０４、２７０６および／または２７０８における顧客に送信されてもよい。動作２７４６で、顧客のサブスクリプションオーダーはオーダー管理およびモニタリングモジュール２７２６によって管理およびトラッキングされてもよい。いくつかの例では、オーダー管理およびモニタリングモジュール２７２６は、用いられるストレージの量、転送されるデータ量、ユーザの数、システムアップ時間およびシステムダウン時間の量などのような、サブスクリプションオーダーにおけるサービスに対する使用統計を収集するよう構成されてもよい。

【０２５７】

特定の実施形態において、クラウドインフラストラクチャシステム２７００はアイデンティティ管理モジュール２７２８を含んでもよい。アイデンティティ管理モジュール２７２８は、クラウドインフラストラクチャシステム２７００におけるアクセス管理および承認サービスなどのようなアイデンティティサービスを提供するよう構成されてもよい。いくつかの実施形態において、アイデンティティ管理モジュール２７２８は、クラウドインフラストラクチャシステム２７０２によって提供されるサービスを利用することを望む顧客についての情報を制御してもよい。そのような情報は、そのような顧客のアイデンティティを認証する情報、およびそれらの顧客がさまざまなシステム資源（たとえばファイル、ディレクトリ、アプリケーション、通信ポート、メモリセグメントなど）に対してどのアクションを実行するよう承認されるかを記述する情報を含むことができる。アイデンティティ管理モジュール２７２８は、さらに、各顧客についての記述的情報およびどのように誰によってその記述的情報がアクセスおよび修正され得るかの管理を含んでもよい。

【０２５８】

図２８は、本開示の実施形態を実施するために使用され得る例示的なコンピュータシステム２８００を示す。いくつかの実施形態では、コンピュータシステム２８００を使用して、上述したさまざまなサーバおよびコンピュータシステムのいずれかを実装することができる。図２８に示すように、コンピュータシステム２８００は、バスサブシステム２８０２を介して複数の周辺サブシステムと通信する処理サブシステム２８０４を含むさまざ

まなサブシステムを含む。これらの周辺サブシステムは、処理加速ユニット２８０６、Ｉ／Ｏサブシステム２８０８、ストレージサブシステム２８１８および通信サブシステム２８２４を含んでもよい。ストレージサブシステム２８１８は、有形のコンピュータ可読記憶媒体２８２２およびシステムメモリ２８１０を含む。

【０２５９】

バスサブシステム２８０２は、コンピュータシステム２８００のさまざまなコンポーネントおよびサブシステムに意図されるように互いに通信させるための機構を提供する。バスサブシステム２８０２は単一のバスとして概略的に示されているが、バスサブシステムの代替的実施例は、複数のバスを利用してもよい。バスサブシステム２８０２は、さまざまなバスアーキテクチャのうちのいずれかを用いるメモリバスまたはメモリコントローラ、周辺バスおよびローカルバスを含むいくつかのタイプのバス構造のうちのいずれかであってもよい。たとえば、そのようなアーキテクチャは、業界標準アーキテクチャ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ：ＩＳＡ）バス、マイクロチャネルアーキテクチャ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ：ＭＣＡ）バス、エンハンストＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ：ＥＩＳＡ）バス、ビデオ・エレクトロニクス・スタンダーズ・アソシエーション（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ：ＶＥＳＡ）ローカルバス、およびＩＥＥＥＰ１３８６．１規格に従って製造される中二階バスとして実現され得る周辺コンポーネントインターコネクト（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ：ＰＣＩ）バスなどを含んでもよい。

【０２６０】

処理サブシステム２８０４は、コンピュータシステム２８００の動作を制御し、１つ以上の処理ユニット２８３２，２８３４などを含むことができる。処理ユニットは、単一コアまたはマルチコアプロセッサを含む１つ以上のプロセッサ、１つ以上のプロセッサコア、またはそれらの組み合わせであってもよい。いくつかの実施形態では、処理サブシステム２８０４は、グラフィックスプロセッサ、デジタル信号プロセッサ（ＤＳＰ）などのような１つ以上の専用コプロセッサを含むことができる。いくつかの実施形態では、処理サブシステム２８０４の処理ユニットの一部または全部は、特定用途向け集積回路（ＡＳＩＣ）またはフィールドプログラマブルゲートアレイ（ＦＰＧＡ）などのカスタマイズされた回路を使用して実装することができる。

【０２６１】

いくつかの実施形態では、処理サブシステム２８０４内の処理ユニットは、システムメモリ２８１０またはコンピュータ可読記憶媒体２８２２に格納された命令を実行することができる。さまざまな実施形態では、処理ユニットはさまざまなプログラムまたはコード命令を実行し、複数の同時に実行するプログラムまたはプロセスを維持できる。任意の所与の時点で、実行されるべきプログラムコードの一部または全部は、システムメモリ２８１０および／または潜在的に１つ以上の記憶装置を含むコンピュータ可読記憶媒体２８１０に常駐することができる。適切なプログラミングを介して、処理サブシステム２８０４は、使用パターンに応答して文書（たとえばウェブページ）を動的に修正するために上記のさまざまな機能を提供することができる。

【０２６２】

特定の実施形態では、コンピュータシステム２８００によって実行される全体的な処理を加速するよう、カスタマイズされた処理を実行するために、または処理サブシステム２８０４によって実行される処理の一部をオフロードするために、処理加速ユニット２８０６を設けることができる。

【０２６３】

Ｉ／Ｏサブシステム２８０８は、コンピュータシステム２８００に情報を入力するための、および／またはコンピュータシステム２８００から、もしくはコンピュータシステム２８００を介して、情報を出力するための、デバイスおよび機構を含むことができる。一般に、「入力デバイス」という語の使用は、コンピュータシステム２８００に情報を入力

するための全ての考えられ得るタイプのデバイスおよび機構を含むよう意図される。ユーザインターフェイス入力デバイスは、たとえば、キーボード、マウスまたはトラックボールなどのポインティングデバイス、ディスプレイに組み込まれたタッチパッドまたはタッチスクリーン、スクロールホイール、クリックホイール、ダイアル、ボタン、スイッチ、キーパッド、音声コマンド認識システムを伴う音声入力デバイス、マイクロフォン、および他のタイプの入力デバイスを含んでもよい。ユーザインタフェース入力デバイスは、ユーザが入力デバイスを制御しそれと対話することを可能にするＭｉｃｒｏｓｏｆｔ Ｋｉｎｅｃｔ（登録商標）モーションセンサ、Ｍｉｃｒｏｓｏｆｔ Ｘｂｏｘ（登録商標）３６０ゲームコントローラ、ジェスチャーおよび音声コマンドを用いる入力を受信するためのインタフェースを提供するデバイスなど、モーションセンシングおよび／またはジェスチャ認識デバイスも含んでもよい。ユーザインターフェイス入力デバイスは、ユーザから目の動き（たとえば、写真を撮っている間および／またはメニュー選択を行なっている間の「まばたき」）を検出し、アイジェスチャを入力デバイス（たとえばＧｏｏｇｌｅ Ｇｌａｓｓ（登録商標））への入力として変換するＧｏｏｇｌｅ Ｇｌａｓｓ（登録商標）瞬き検出器などのアイジェスチャ認識デバイスも含んでもよい。また、ユーザインターフェイス入力デバイスは、ユーザが音声コマンドを介して音声認識システム（たとえばＳｉｒｉ（登録商標）ナビゲータ）と対話することを可能にする音声認識感知デバイスを含んでもよい。

【０２６４】

ユーザインターフェイス入力デバイスの他の例は、三次元（３Ｄ）マウス、ジョイスティックまたはポインティングスティック、ゲームパッドおよびグラフィックタブレット、ならびにスピーカ、デジタルカメラ、デジタルカムコーダ、ポータブルメディアプレーヤ、ウェブカム、画像スキャナ、指紋スキャナ、バーコードリーダ３Ｄスキャナ、３Ｄプリンタ、レーザレンジファインダ、および視線追跡デバイスなどの聴覚／視覚デバイスも含んでもよいが、それらに限定されるものではない。また、ユーザインターフェイス入力デバイスは、たとえば、コンピュータ断層撮影、磁気共鳴撮像、ポジションエミッショントモグラフィー、医療用超音波検査デバイスなどの医療用画像化入力デバイスを含んでもよい。ユーザインターフェイス入力デバイスは、たとえば、ＭＩＤＩキーボード、デジタル楽器などの音声入力デバイスも含んでもよい。

【０２６５】

ユーザインターフェイス出力デバイスは、ディスプレイサブシステム、インジケータライト、または音声出力デバイスなどの非ビジュアルディスプレイなどを含んでもよい。ディスプレイサブシステムは、陰極線管（ＣＲＴ）、液晶ディスプレイ（ＬＣＤ）またはプラズマディスプレイを使うものなどのフラットパネルデバイス、投影デバイス、タッチスクリーンなどであってもよい。一般に、「出力デバイス」という語の使用は、コンピュータシステム２８００からユーザまたは他のコンピュータに情報を出力するための全ての考えられ得るタイプのデバイスおよび機構を含むよう意図される。たとえば、ユーザインターフェイス出力デバイスは、モニタ、プリンタ、スピーカ、ヘッドフォン、自動車ナビゲーションシステム、プロッタ、音声出力デバイスおよびモデムなどの、テキスト、グラフィックスおよび音声／映像情報を視覚的に伝えるさまざまな表示デバイスを含んでもよいが、それらに限定されるものではない。

【０２６６】

ストレージサブシステム２８１８は、コンピュータシステム２８００によって使用される情報を格納するためのリポジトリまたはデータストアを提供する。ストレージサブシステム２８１８は、いくつかの実施形態の機能を提供する基本的なプログラミングおよびデータ構成を記憶するための有形の非一時的なコンピュータ可読記憶媒体を提供する。処理サブシステム２８０４によって実行されると、上述の機能を提供するソフトウェア（プログラム、コードモジュール、命令）が、ストレージサブシステム２８１８に格納されてもよい。ソフトウェアは、処理サブシステム２８０４の１つ以上の処理ユニットによって実行されてもよい。ストレージサブシステム２８１８はまた、本開示に従って使用されるデ

ータを格納するためのリポジトリを提供してもよい。
【０２６７】
ストレージサブシステム２８１８は、揮発性および不揮発性メモリデバイスを含む１つ以上の非一時的メモリデバイスを含むことができる。図２８に示すように、ストレージサブシステム２８１８は、システムメモリ２８１０およびコンピュータ可読記憶媒体２８２２を含む。システムメモリ２８１０は、プログラム実行中に命令およびデータを記憶するための揮発性主ランダムアクセスメモリ（ＲＡＭ）と、固定命令が記憶される不揮発性読み出し専用メモリ（ＲＯＭ）またはフラッシュメモリとを含む、いくつかのメモリを含むことができる。いくつかの実現例では、起動中などにコンピュータシステム２８００内の要素間における情報の転送を助ける基本的なルーチンを含むベーシックインプット／アウトプットシステム（ｂａｓｉｃ　ｉｎｐｕｔ／ｏｕｔｐｕｔ　ｓｙｓｔｅｍ：ＢＩＯＳ）は、典型的には、ＲＯＭに記憶されてもよい。ＲＡＭは、通常、処理サブシステム２８０４によって現在動作され実行されているデータおよび／またはプログラムモジュールを含む。いくつかの実現例では、システムメモリ２８１０は、スタティックランダムアクセスメモリ（ＳＲＡＭ）またはダイナミックランダムアクセスメモリ（ＤＲＡＭ）などの複数の異なるタイプのメモリを含んでもよい。
【０２６８】
一例として、限定を伴うことなく、図２８に示されるように、システムメモリ２８１０は、クライアントアプリケーション、ウェブブラウザ、中間層アプリケーション、リレーショナルデータベース管理システム（ＲＤＢＭＳ）などを含んでもよいアプリケーションプログラム２８１２、プログラムデータ２８１４およびオペレーティングシステム２８１６を記憶してもよい。一例として、オペレーティングシステム２８１６は、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ（登録商標）、Ａｐｐｌｅ　Ｍａｃｉｎｔｏｓｈ（登録商標）および／もしくはＬｉｎｕｘオペレーティングシステム、さまざまな市場で入手可能なＵＮＩＸ（登録商標）またはＵＮＩＸのようなオペレーティングシステム（さまざまなＧＮＵ／Ｌｉｎｕｘオペレーティングシステム、Ｇｏｏｇｌｅ　Ｃｈｒｏｍｅ（登録商標）ＯＳなどを含むがそれらに限定されない）、ならびに／または、ｉＯＳ、Ｗｉｎｄｏｗｓ（登録商標）　Ｐｈｏｎｅ、Ａｎｄｒｏｉｄ（登録商標）　ＯＳ、ＢｌａｃｋＢｅｒｒｙ（登録商標）２６　ＯＳ、およびＰａｌｍ（登録商標）　ＯＳオペレーティングシステムなどのモバイルオペレーティングシステムのさまざまなバージョンを含んでもよい。
【０２６９】
コンピュータ可読記憶媒体２８２２は、いくつかの実施形態の機能性を提供するプログラミングおよびデータ構成を格納することができる。処理サブシステム２８０４によって実行されると、プロセッサが上記の機能を提供するソフトウェア（プログラム、コードモジュール、命令）は、ストレージサブシステム２８１８に格納されてもよい。一例として、コンピュータ可読記憶媒体２８２２は、ハードディスクドライブ、磁気ディスクドライブ、ＣＤ　ＲＯＭ、ＤＶＤ、Ｂｌｕ－Ｒａｙ（登録商標）ディスクなどの光ディスクドライブ、またはその他の光学媒体のような不揮発性メモリを含むことができる。コンピュータ可読記憶媒体２８２２は、Ｚｉｐ（登録商標）ドライブ、フラッシュメモリカード、ユニバーサルシリアルバス（ＵＳＢ）フラッシュドライブ、セキュアデジタル（ＳＤ）カード、ＤＶＤディスク、デジタルビデオテープなどを含んでもよいが、それらに限定されるものではない。コンピュータ可読記憶媒体２８２２は、フラッシュメモリベースのＳＳＤ、エンタープライズフラッシュドライブ、ソリッドステートＲＯＭなどの不揮発性メモリに基づくソリッドステートドライブ（ＳＳＤ）、ソリッドステートＲＡＭ、ダイナミックＲＡＭ、スタティックＲＡＭなどの揮発性メモリに基づくＳＳＤ、ＤＲＡＭベースのＳＳＤ、磁気抵抗ＲＡＭ（ＭＲＡＭ）ＳＳＤ、およびＤＲＡＭとフラッシュメモリベースのＳＳＤとの組み合わせを使用するハイブリッドＳＳＤも含んでもよい。コンピュータ可読媒体２８２２は、コンピュータシステム２８００のためのコンピュータ可読命令、データ構造、プログラムモジュール、および他のデータのストレージを提供することができる。
【０２７０】

ある実施形態では、ストレージサブシステム２８００は、コンピュータ可読記憶媒体２８２２にさらに接続可能なコンピュータ可読記憶媒体リーダ２８２０も含んでもよい。システムメモリ２８１０とともに、およびオプションとしてシステムメモリ２８１０との組み合わせで、コンピュータ可読記憶媒体２８２２は、コンピュータ可読情報を記憶するためにリモート、ローカル、固定および／またはリムーバブル記憶装置に記憶媒体を加えたものを包括的に表してもよい。

【０２７１】

特定の実施形態では、コンピュータシステム２８００は、１つ以上の仮想マシンを実行するためのサポートを提供することができる。コンピュータシステム２８００は、仮想マシンの構成および管理を容易にするためのハイパーバイザなどのプログラムを実行することができる。各仮想マシンは、メモリ、計算（たとえばプロセッサ、コア）、Ｉ／Ｏ、およびネットワーキングリソースを割り当てられてもよい。各仮想マシンは、通常、コンピュータシステム２８００によって実行される他の仮想マシンによって実行されるオペレーティングシステムと同じでも異なってもよい、それ自体のオペレーティングシステムを実行する。したがって、複数のオペレーティングシステムが潜在的にコンピュータシステム２８００によって同時に実行され得る。各仮想マシンは、通常、他の仮想マシンとは独立して動作する。

【０２７２】

通信サブシステム２８２４は、他のコンピュータシステムおよびネットワークに対するインターフェイスを提供する。通信サブシステム２８２４は、他のシステムとコンピュータシステム２８００との間のデータの送受のためのインターフェイスとして働く。たとえば、通信サブシステム２８２４は、コンピュータシステム２８００が、１つ以上のクライアントデバイスとの間で情報を送受信するために、インターネットを介して１つ以上のクライアントデバイスへの通信チャネルを確立することを可能にすることができる。さらに、通信サブシステム２８２４を使用して、成功したログインの通知、または特権アカウントマネージャから要求側ユーザへのパスワードの再入力の通知を伝達することができる。

【０２７３】

通信サブシステム２８２４は、有線および／または無線通信プロトコルの両方をサポートすることができる。たとえば、ある実施形態では、通信サブシステム２８２４は、（たとえば、セルラー電話技術、３Ｇ、４ＧもしくはＥＤＧＥ（グローバル進化のための高速データレート）などの先進データネットワーク技術、ＷｉＦｉ（ＩＥＥＥ８０２．１１ファミリー規格、もしくは他のモバイル通信技術、またはそれらのいずれかの組み合わせを用いて）無線音声および／またはデータネットワークにアクセスするための無線周波数（ＲＦ）送受信機コンポーネント、グローバルポジショニングシステム（ＧＰＳ）受信機コンポーネント、ならびに／または他のコンポーネントを含んでもよい。いくつかの実施形態では、通信サブシステム２８２４は、無線インターフェイスに加えて、またはその代わりに、有線ネットワーク接続（たとえば、イーサネット）を提供することができる。

【０２７４】

通信サブシステム２８２４は、さまざまな形式でデータを受信し、送信することができる。たとえば、いくつかの実施形態では、通信サブシステム２８２４は、構造化データフィードおよび／または非構造化データフィード２８２６、イベントストリーム２８２８、イベント更新２８３０などの形式で入力通信を受信することができる。たとえば、通信サブシステム２８２４は、ソーシャルメディアネットワークおよび／またはＴｗｉｔｔｅｒ（登録商標）フィード、Ｆａｃｅｂｏｏｋ（登録商標）更新情報、Ｒｉｃｈ Ｓｉｔｅ Ｓｕｍｍａｒｙ（ＲＳＳ）フィードなどのウェブフィード、および／もしくは１つ以上の第三者情報源からのリアルタイム更新情報などの他の通信サービスのユーザからリアルタイムでデータフィード２８２６を受信（または送信）するように構成されてもよい。

【０２７５】

ある実施形態では、通信サブシステム２８２４は、連続データストリームの形式でデータを受信するように構成されてもよく、当該連続データストリームは、明確な終端を持た

ない、本来は連続的または無限であり得るリアルタイムイベントのイベントストリーム２８２８および／またはイベント更新情報２８３０を含んでもよい。連続データを生成するアプリケーションの例としては、たとえば、センサデータアプリケーション、金融株式相場表示板、ネットワーク性能測定ツール（たとえば、ネットワークモニタリングおよびトラフィック管理アプリケーション）、クリックストリーム解析ツール、自動車交通モニタリングなどを挙げることができる。

【０２７６】

また、通信サブシステム２８２４は、構造化されたおよび／または構造化されていないデータフィード２８２６、イベントストリーム２８２８、イベント更新情報２８３０などを、コンピュータシステム２８００に結合される１つ以上のストリーミングデータソースコンピュータと通信し得る１つ以上のデータベースに出力するよう構成されてもよい。

【０２７７】

コンピュータシステム２８００は、手持ち式の携帯デバイス（たとえば、ｉＰｈｏｎｅ（登録商標）携帯電話、ｉＰａｄ（登録商標）コンピューティングタブレット、ＰＤＡ）、ウェアラブルデバイス（たとえば、Ｇｏｏｇｌｅ Ｇｌａｓｓ（登録商標）頭部装着型ディスプレイ）、パソコン、ワークステーション、メインフレーム、キオスク、サーバラック、またはその他のデータ処理システムを含む、さまざまなタイプのもののうちの１つであり得る。

【０２７８】

常に変化するコンピュータおよびネットワークの性質のため、図２８に示されるコンピュータシステム２８００の記載は、単に具体的な例として意図される。図２８に示されるシステムよりも多くのコンポーネントまたは少ないコンポーネントを有する多くの他の構成が可能である。本明細書における開示および教示に基づいて、当業者は、さまざまな実施形態を実現するための他の態様および／または方法を理解するであろう。

【０２７９】

本開示の特定の実施形態について説明したが、本開示の範囲内で、さまざまな修正、変更、代替構成、および同等物も包含される。本開示の実施形態は、ある特定のデータ処理環境内の動作に限定されず、複数のデータ処理環境内で自由に動作することができる。さらに、本開示の実施形態は特定の一連のトランザクションおよびステップを使用して記載されたが、本開示の範囲は記載された一連のトランザクションおよびステップに限定されないことは当業者には明らかである。上述した実施形態のさまざまな特徴および態様は、個別にまたはともに使用されてもよい。

【０２８０】

さらに、本開示の実施形態をハードウェアとソフトウェアとの特定の組み合わせを用いて記載したが、ハードウェアとソフトウェアとの他の組み合わせも本開示の範囲内であることを認識すべきである。本開示の実施形態は、ハードウェアでのみ、またはソフトウェアでのみ、またはそれらの組み合わせを用いて実施されてもよい。本明細書に記載されたさまざまなプロセスは、同じプロセッサまたは任意の組み合わせの異なるプロセッサ上で実現できる。したがって、構成要素またはモジュールが特定の動作を実行するように構成されるとして記載されている場合、そのような構成は、たとえば、動作を実行する電子回路を設計すること、プログラミング可能な電子回路（マイクロプロセッサなど）をプログラミングして動作を実行すること、またはそれらの任意の組み合わせによって達成され得る。プロセスは、プロセス間通信のための従来の技術を含むがこれに限定されないさまざまな技術を使用して通信することができ、異なる対のプロセスは異なる技術を使用してもよく、同じ対のプロセスは異なる時間に異なる技術を使用してもよい。

【０２８１】

したがって、明細書および図面は、限定的な意味ではなく例示的であると見なされるべきである。しかしながら、特許請求の範囲に記載されたより広範な精神および範囲から逸脱することなく、追加、削減、削除、ならびに他の修正および変更がなされ得ることは明らかであろう。このように、特定の開示の実施形態を説明したが、これらは限定すること

を意図するものではない。さまざまな修正および均等物は、特許請求の範囲内にある。修正および変形は、開示された特徴の関連の組み合わせを含む。

【図面】

【図１】

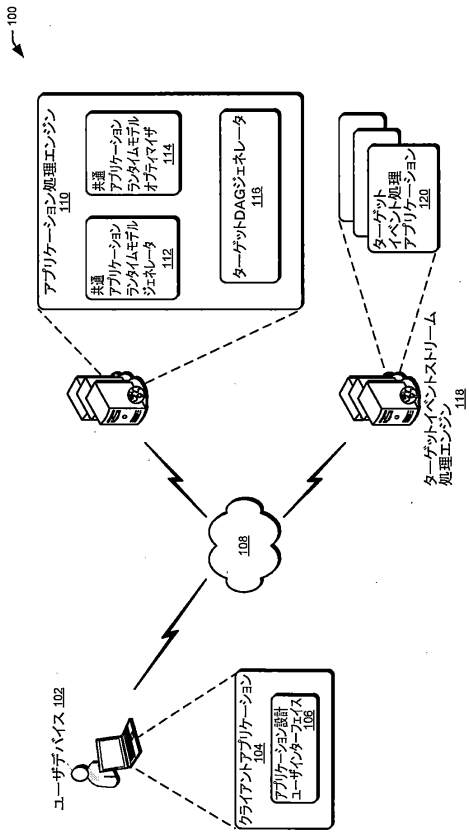

FIG. 1

【図２】

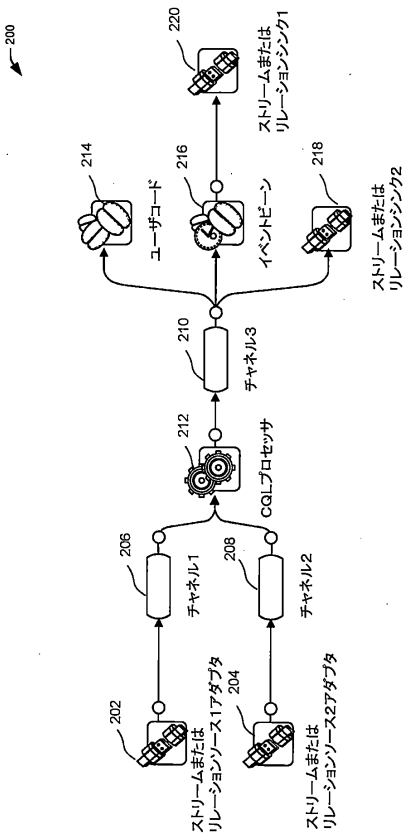

FIG. 2

【図３】

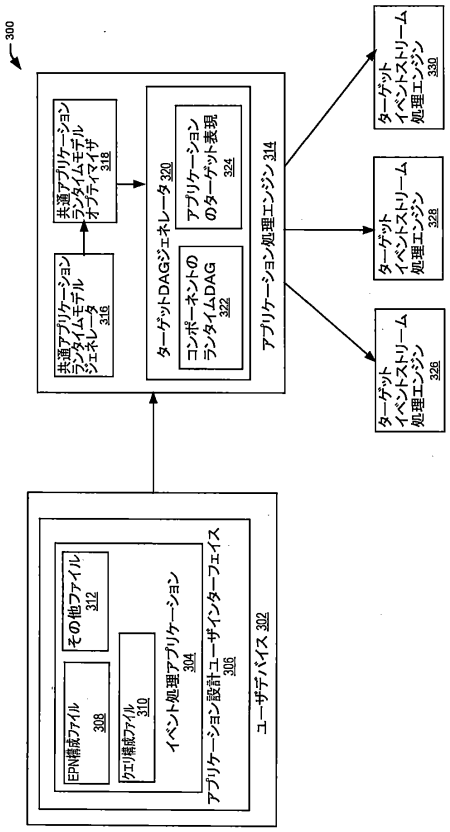

FIG. 3

【図４】

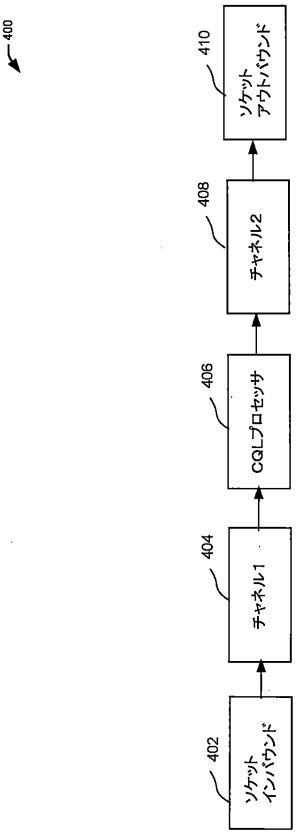

FIG. 4

【図５】

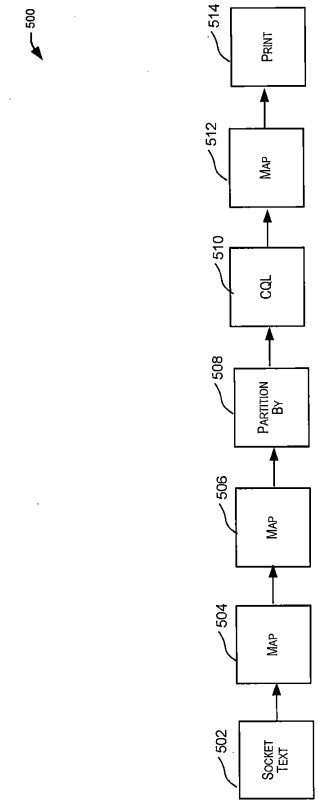

FIG. 5

【図６】

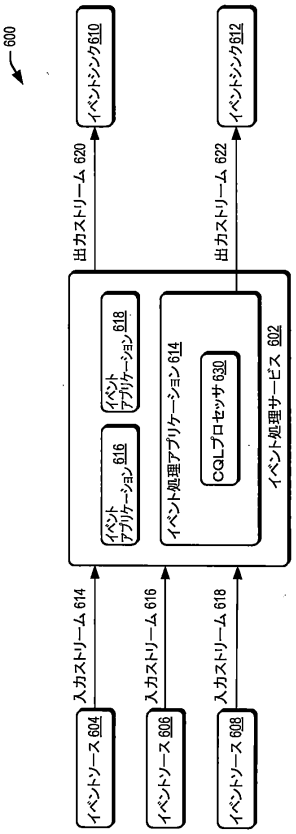

FIG. 6

【図７】

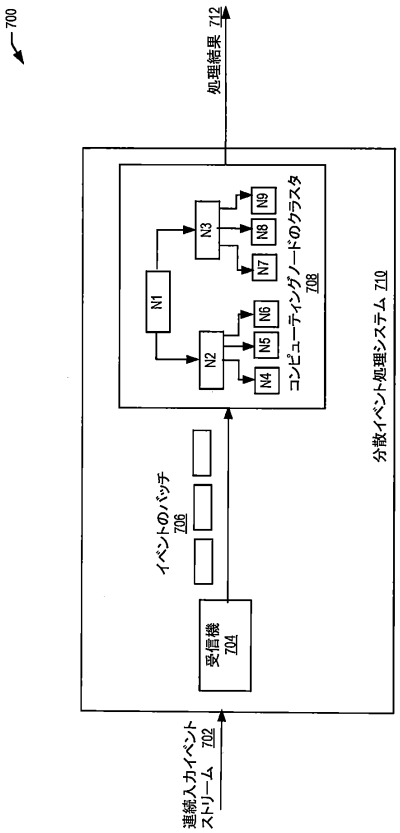

FIG. 7

【図８】

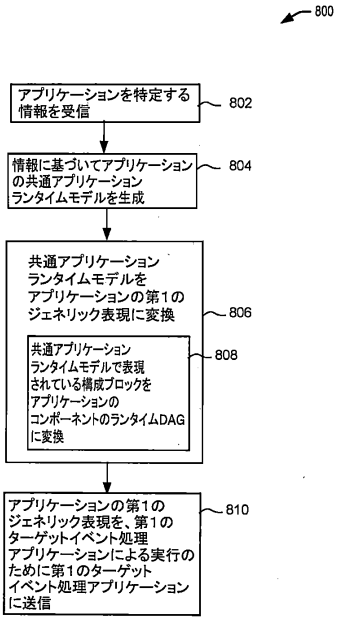

FIG. 8

【図９】

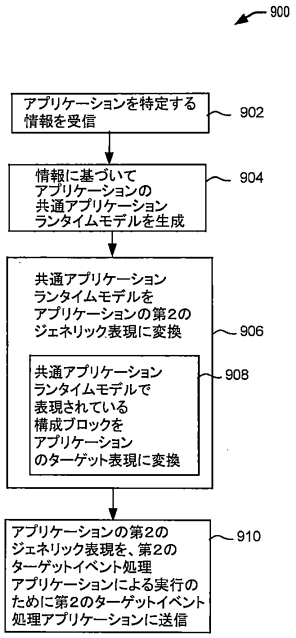

FIG. 9

【図１０】

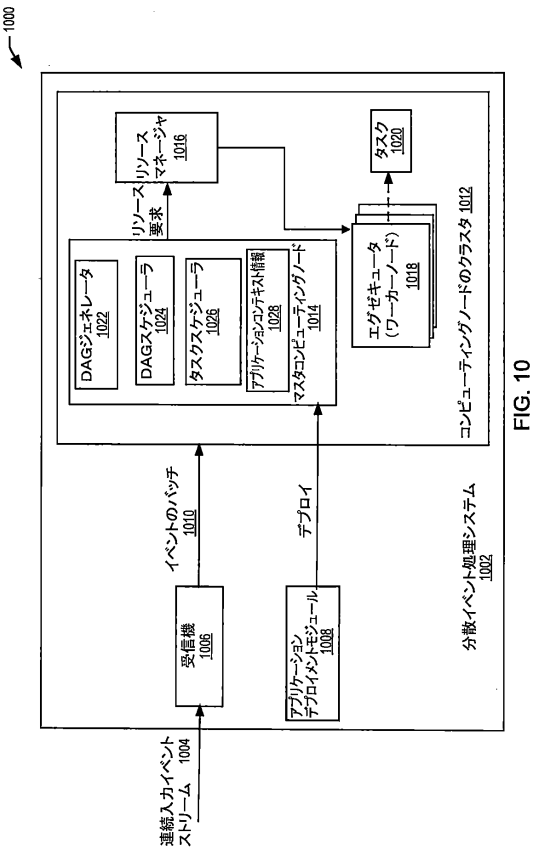

FIG. 10

【図１１】

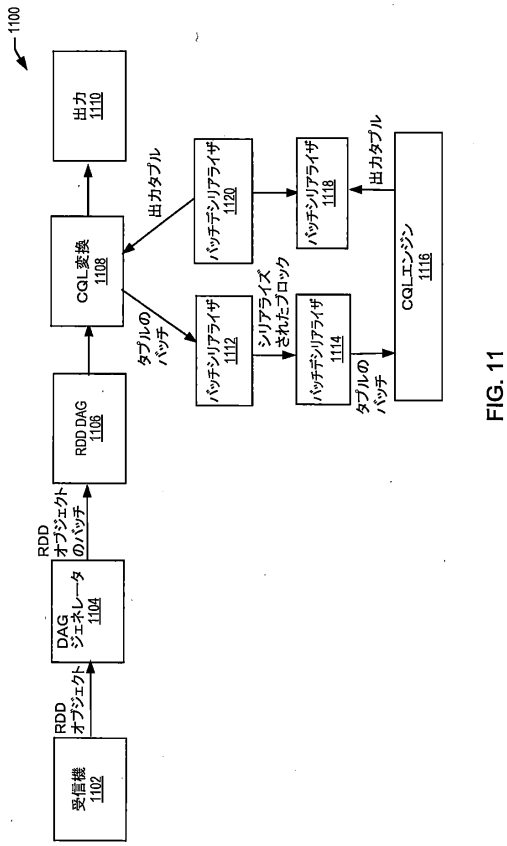

FIG. 11

【図１２】

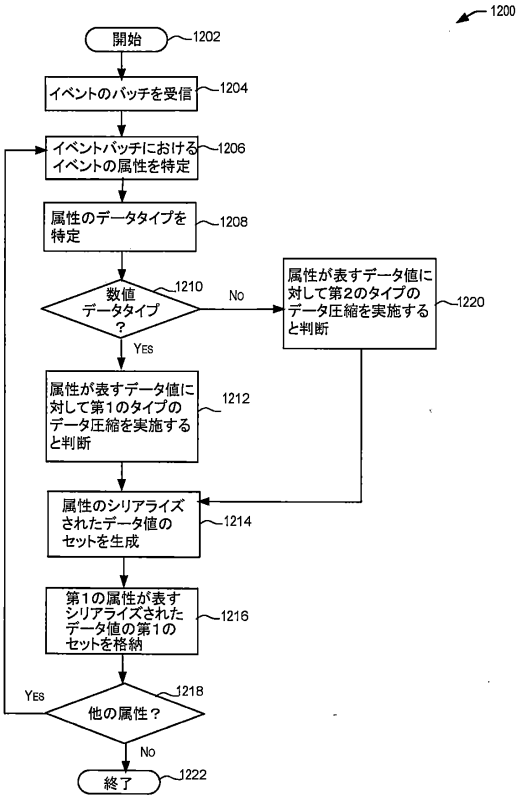

FIG. 12

【図１３Ａ】

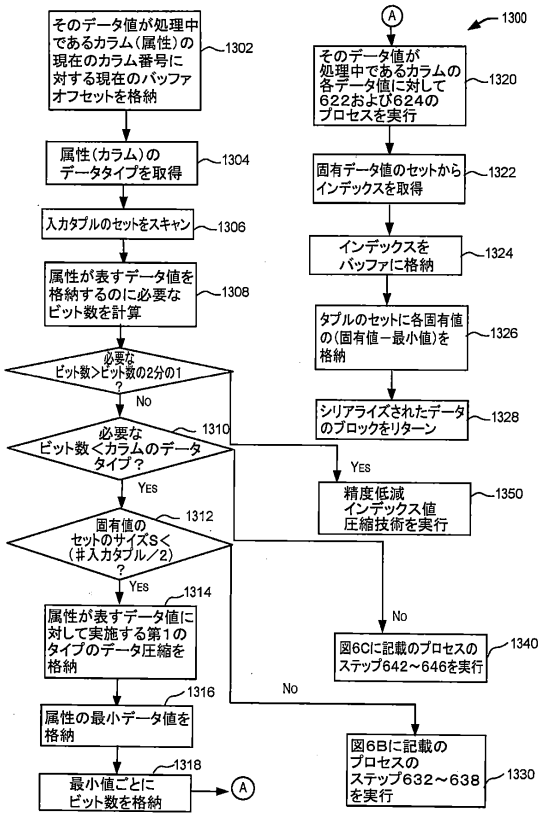

FIG. 13A

【図１３Ｂ】

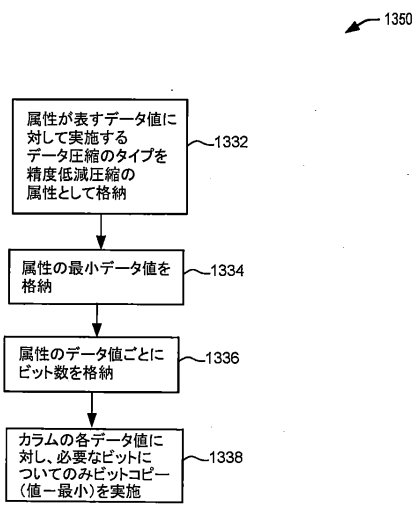

FIG. 13B

【図１３Ｃ】

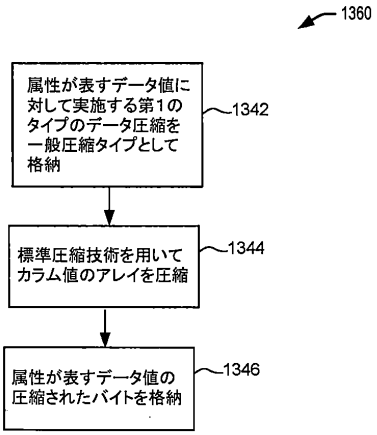

FIG. 13C

【図１３Ｄ】

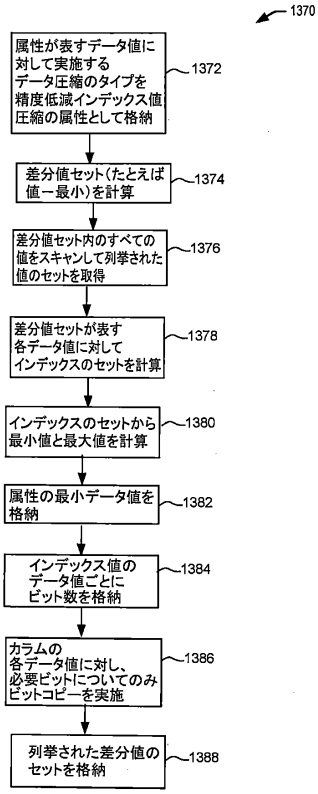

FIG. 13D

【図１４】

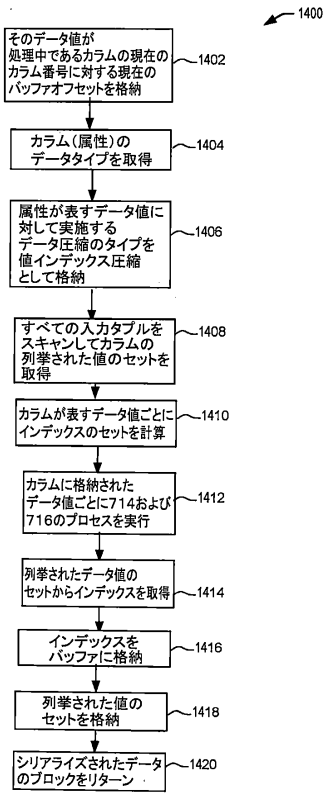

FIG. 14

【図１５】

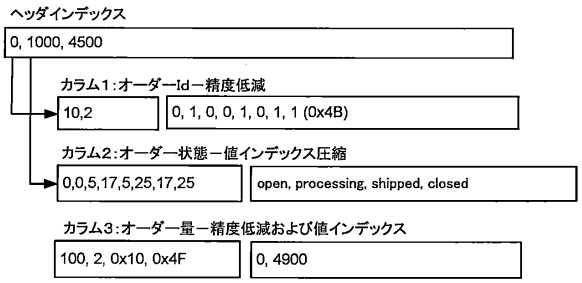

FIG. 15

【図１６】

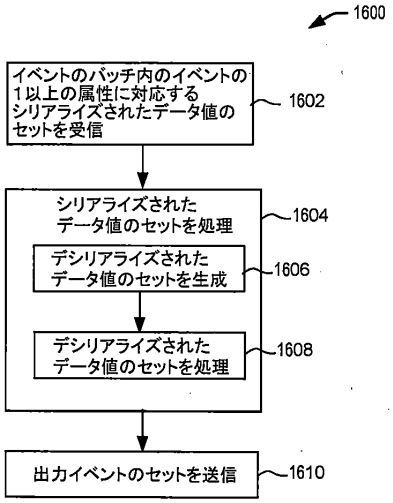

FIG. 16

【図１７】

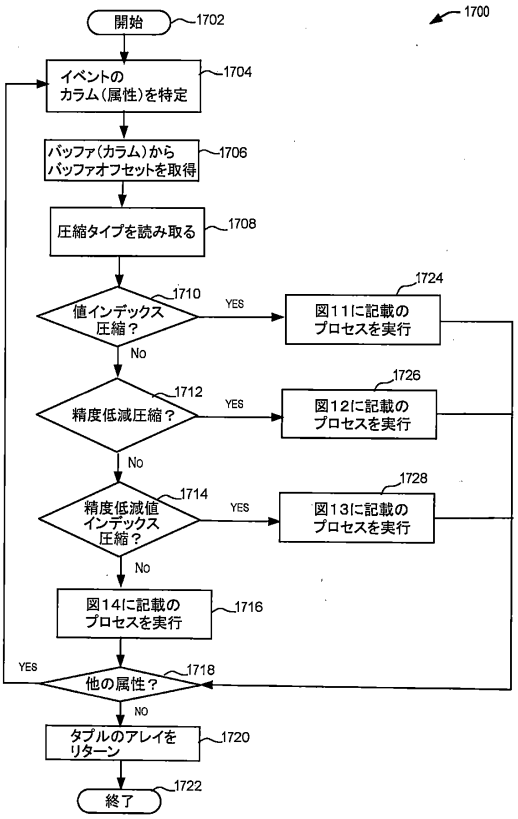

FIG. 17

【図１８】

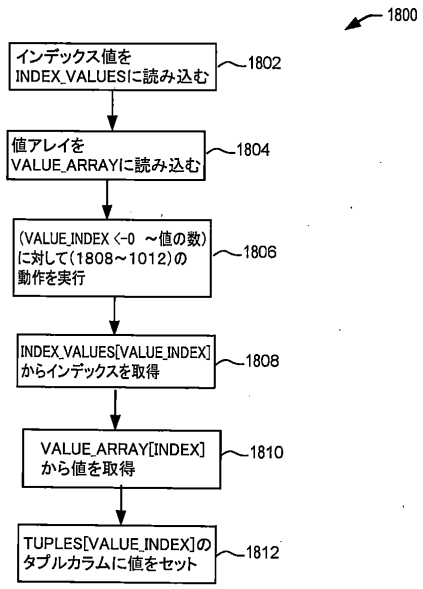

FIG. 18

【図１９】

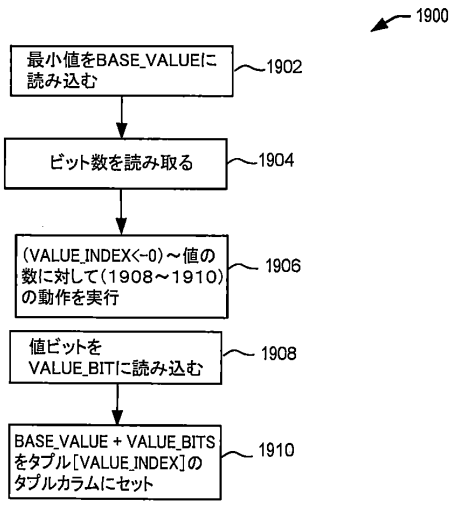

FIG. 19

【図２０】

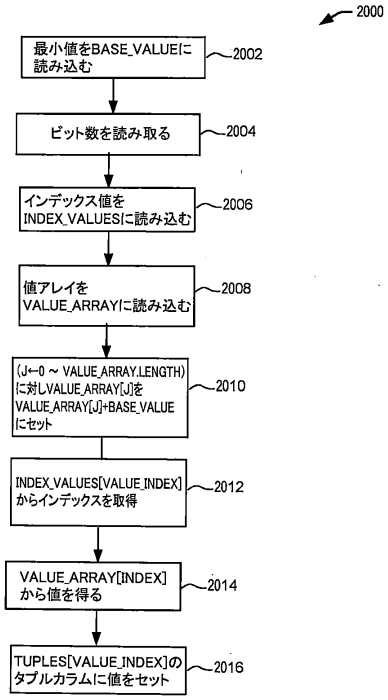

FIG. 20

【図２１】

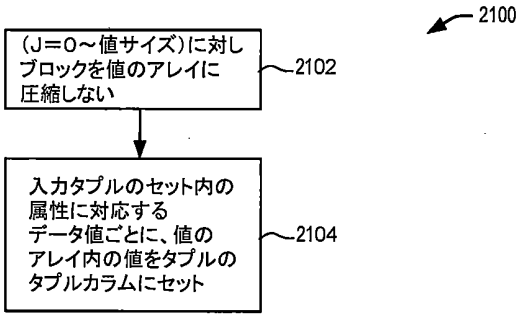

FIG. 21

【図２２】

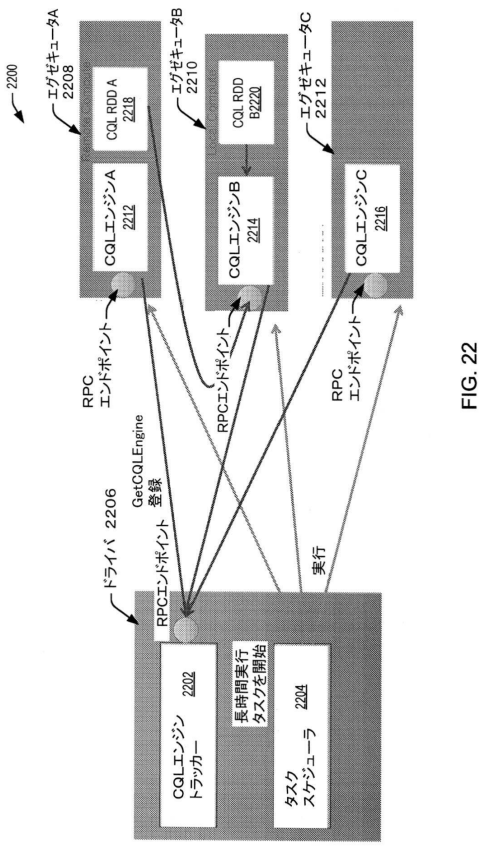

FIG. 22

【図２３】

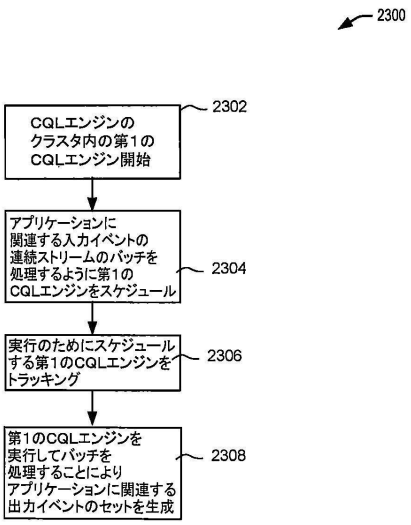

FIG. 23

【図２４】

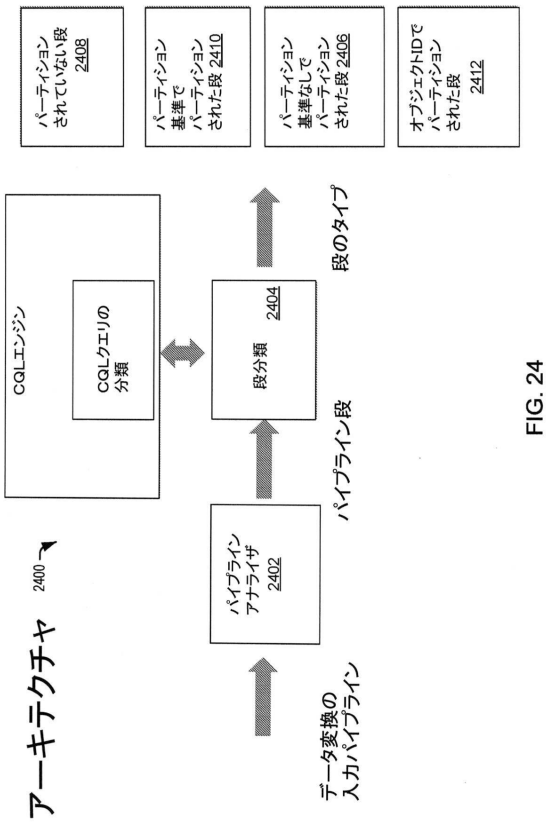

FIG. 24

【図２５】

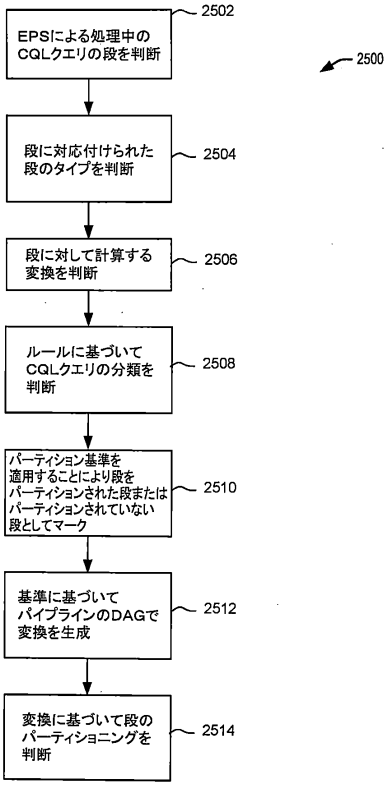

FIG. 25

【図２６】

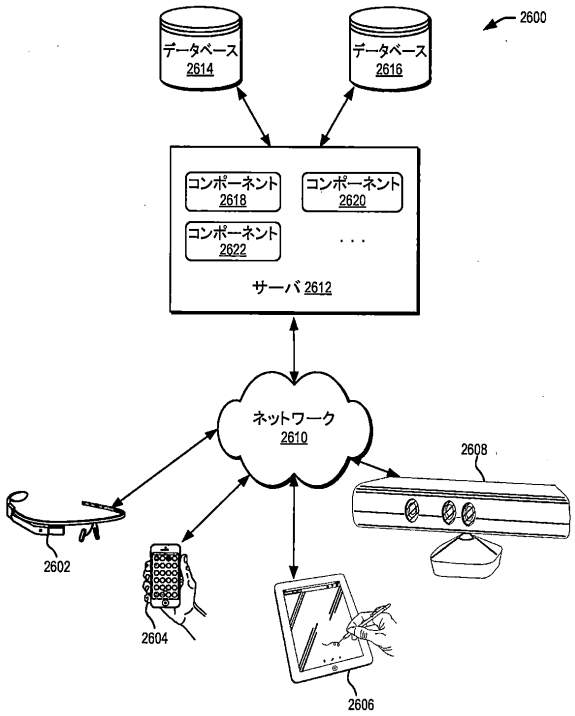

FIG. 26

【図２７】

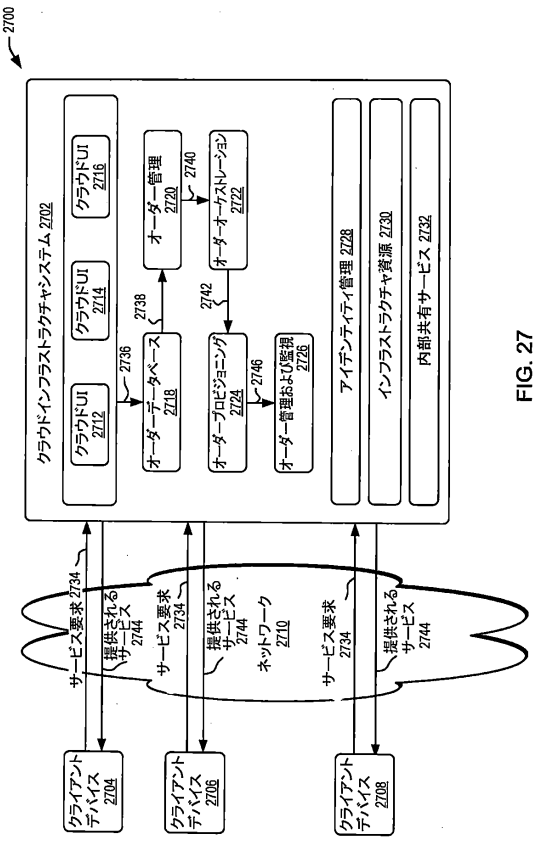

FIG. 27

【図２８】

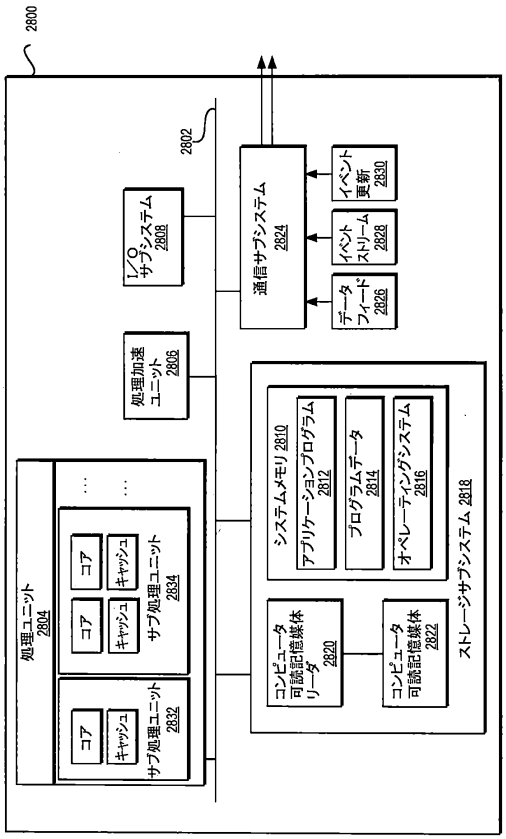
2800
2802
処理ユニット 2804
サブ処理ユニット 2832
サブ処理ユニット 2834
コア
キャッシュ1
キャッシュ2
処理加速ユニット 2806
I/Oサブシステム 2808
通信サブシステム 2824
データフィード 2826
イベントストリーム 2828
イベント更新 2830
システムメモリ 2810
アプリケーションプログラム 2812
プログラムデータ 2814
オペレーティングシステム 2816
ストレージサブシステム 2818
コンピュータ可読記憶媒体リーダ 2820
コンピュータ可読記憶媒体 2822

FIG. 28

フロントページの続き

米国(US)
審査官　三橋　竜太郎
(56)参考文献　特表２０１６－５０４６７９（ＪＰ，Ａ）
米国特許出願公開第２０１６／０１６２７８７（ＵＳ，Ａ１）
米国特許出願公開第２０１２／００５９８３９（ＵＳ，Ａ１）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１６／００－１６／９５８