

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7018435号
(P7018435)

(45)発行日 令和4年2月10日(2022.2.10)

(24)登録日 令和4年2月2日(2022.2.2)

(51)国際特許分類	F I
G 0 6 F 16/11 (2019.01)	G 0 6 F 16/11
G 0 6 F 16/903 (2019.01)	G 0 6 F 16/903

請求項の数 15 (全47頁)

(21)出願番号	特願2019-512634(P2019-512634)	(73)特許権者	502303739
(86)(22)出願日	平成29年9月15日(2017.9.15)		オラクル・インターナショナル・コーポレーション
(65)公表番号	特表2019-535058(P2019-535058 A)		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、オラクル・パークウェイ500
(43)公表日	令和1年12月5日(2019.12.5)	(74)代理人	110001195
(86)国際出願番号	PCT/US2017/051897		特許業務法人深見特許事務所
(87)国際公開番号	WO2018/053343	(72)発明者	パーク, ホヨン
(87)国際公開日	平成30年3月22日(2018.3.22)		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
審査請求日	令和2年6月22日(2020.6.22)	(72)発明者	ビシュノイ, サンディーブ
(31)優先権主張番号	201641031479		アメリカ合衆国、94941 カリフォルニア州、ミル・バレー、セナリ、
(32)優先日	平成28年9月15日(2016.9.15)		最終頁に続く
(33)優先権主張国・地域又は機関	インド(IN)		

(54)【発明の名称】 スナップショットおよび状態のマイクロバッチ方式による管理

(57)【特許請求の範囲】

【請求項1】

連続クエリ言語（CQL）エンジンから作成されるスナップショットを管理するための方法であって、

コンピューティングデバイスが、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することと、

前記コンピューティングデバイスが、前記CQLエンジンを用いて、前記入力イベントを処理して、前記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と、

前記コンピューティングデバイスが、前記CQLエンジンによって実現されるスナップショット管理アルゴリズムを用いて、前記アプリケーションに関連する前記出力イベントのセットに少なくとも部分的に基づいて、システムの現在の状態のスナップショットを生成することと、

前記コンピューティングデバイスが、前記システムの前記現在の状態の前記スナップショットに関連付けられるスナップショット情報にアクセスするために第1のディレクトリ構造を生成することと、

前記コンピューティングデバイスが、前記システムの前記現在の状態に関連付けられるスナップショットのリストを生成するために第2のディレクトリ構造を生成することと、

前記コンピューティングデバイスが、前記システムの前記現在の状態に関連付けられる前記スナップショットのリストを取得、追加、またはクリーンにするために前記スナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスを決定することと、

前記コンピューティングデバイスが、前記アプリケーションに関連する前記出力イベントのセットを出力待ち行列に格納することと、

前記入力イベントのすべてが処理されたときに、前記コンピューティングデバイスが前記出力待ち行列内の前記出力イベントを送信することとを備える、方法。

【請求項 2】

前記マイクロバッチストリームは、1 秒未満のマイクロバッチに離散化されたデータの連続ストリームである、請求項 1 に記載の方法。

【請求項 3】

前記入力イベントを処理することは、変換されたクエリプランに少なくとも部分的に基づいて、前記入力イベントを処理することを含む、請求項 2 に記載の方法。

【請求項 4】

前記マイクロバッチストリームは、マイクロバッチのデータまたはレジリエント分散データセット (RDD) を含む、請求項 1 ～ 3 のいずれかに記載の方法。

【請求項 5】

前記入力イベントの各々を処理することは、前記変換されたクエリプランに少なくとも部分的に基づいて、前記入力イベントの各々に対して計算を実行することを含む、請求項 4 に記載の方法。

【請求項 6】

前記コンピューティングデバイスが、連続クエリを受信することと、前記連続クエリに変換を適用して前記連続クエリについてクエリプランを生成することと、変換アルゴリズムを用いて前記クエリプランを変換して前記変換されたクエリプランを生成することとをさらに備え、前記連続クエリはパターンマッチングを含む、請求項 5 に記載の方法。

【請求項 7】

システムであって、

コンピュータ実行可能命令を格納するように構成されたメモリと、

前記メモリにアクセスし、前記コンピュータ実行可能命令を実行して、

アプリケーションに関連する入力イベントのマイクロバッチストリームを受信し、

CQL エンジンを用いて、前記入力イベントを処理して、前記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し、

前記 CQL エンジンによって実現されるスナップショット管理アルゴリズムを用いて、前記アプリケーションに関連する前記出力イベントのセットに少なくとも部分的に基づいて、システムの現在の状態のスナップショットを生成し、

前記システムの前記現在の状態の前記スナップショットに関連付けられるスナップショット情報にアクセスするために第 1 のディレクトリ構造を生成し、

前記システムの前記現在の状態に関連付けられるスナップショットのリストを生成するために第 2 のディレクトリ構造を生成し、

前記システムの前記現在の状態に関連付けられる前記スナップショットのリストを取得、追加、またはクリーンにするために前記スナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスを決定し、

前記アプリケーションに関連する前記出力イベントのセットを出力待ち行列に格納し、前記入力イベントのすべてが処理されたときに、前記出力待ち行列内の前記出力イベントを送信するよう構成されるプロセッサとを備える、システム。

【請求項 8】

完全にステートフルなクエリ処理をサポートするようマイクロバッチ方式ストリームを処理するための方法であって、

コンピューティングデバイスが、連続クエリを受信することと、

前記コンピューティングデバイスが、前記連続クエリに対して変換を適用して、前記連続クエリについてクエリプランを生成することと、

前記コンピューティングデバイスが、モニタリング変換プロセスを用いて前記連続クエリを監視することと、

10

20

30

40

50

前記コンピューティングデバイスが、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することと、

前記コンピューティングデバイスが、前記モニタリング変換プロセスに少なくとも部分的に基づいて、前記マイクロバッチストリームの前記入力イベントを処理して、前記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と、

前記コンピューティングデバイスが、前記アプリケーションに関連する前記出力イベントのセットを出力待ち行列に格納することと、

前記入力イベントのすべてが処理されたときに、前記コンピューティングデバイスが前記出力待ち行列内の前記出力イベントを送信することを備える、方法。

【請求項 9】

前記処理は連続クエリ処理エンジンを用いて実行され、前記処理は前記出力イベントを生成するために前記入力イベントの各々を増分的に処理することを含む、請求項 8 に記載の方法。

【請求項 10】

前記変換は有向非巡回グラフ (DAG) 変換である、請求項 9 に記載の方法。

【請求項 11】

前記マイクロバッチストリームはマイクロバッチのデータまたはレジリエント分散データセット (RDD) を含み、前記 DAG 変換は頂点と辺とのセットであり、前記頂点は前記 RDD を表し、前記辺は前記 RDD に適用されるべき操作を表す、請求項 10 に記載の方法。

【請求項 12】

前記入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、前記入力イベントの各々に対して計算を実行することを含む、請求項 11 に記載の方法。

【請求項 13】

前記連続クエリはパターンマッチングを含む、請求項 12 に記載の方法。

【請求項 14】

システムであって、

コンピュータ実行可能命令を格納するように構成されたメモリと、

前記メモリにアクセスし、前記コンピュータ実行可能命令を実行して、

連続クエリを受信し、

前記連続クエリに対して変換を適用して、前記連続クエリについてクエリプランを生成し、モニタリング変換プロセスを用いて前記連続クエリを監視し、

アプリケーションに関連する入力イベントのマイクロバッチストリームを受信し、

前記モニタリング変換プロセスに少なくとも部分的に基づいて、前記マイクロバッチストリームの前記入力イベントを処理して、前記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し、

前記アプリケーションに関連する前記出力イベントのセットを出力待ち行列に格納し、

前記入力イベントのすべてが処理されたときに、前記出力待ち行列内の前記出力イベントを送信するよう構成されるプロセッサとを備える、システム。

【請求項 15】

請求項 1 ～ 5 および 8 ～ 13 のいずれか 1 項に記載の方法をプロセッサに実行させるためのコンピュータ可読プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

関連出願への相互参照

本出願は、「スナップショットおよび状態のマイクロバッチ方式による管理 (MANAGING SNAPSHOTS AND STATE WITH MICRO-BATCHING)」と題される、2016 年 9 月 15 日に提出されたインド仮出願番号第 201641031479 号の優先権および利益

10

20

30

40

50

を主張し、その全内容を、すべての目的のためにここに引用により援用する。

【0002】

本出願は、これとともに同日に出願された、「マイクロバッチストリーミングのための複合イベント処理 (COMPLEX EVENT PROCESSING FOR MICRO-BATCH STREAMING)」と題される、代理人番号第088325-1052956号である、出願連続番号第_____号に関連し、その全内容を、ここに完全に記載されるがごとくここ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。

10

【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。

20

【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イ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。

30

【発明の概要】

【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 (例えば方法、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。一実施形態では、イベント処理システムが開示される。1つ以上のコンピュータからなるシステムは、システムにインストールされ、動作中にシステムにアクションを実行させるソフトウェア、ファームウェア、ハードウェア、またはそれらの組合せを有することによって、特定の動作またはアクションを実行するように構成することができる。1つ以上のコンピュータプログラムは、データ処理装置によって実行されるとデータ処理装置にアクションを実行させる命令を含むことによって、特定の動作またはアクションを実行するように構成することができる。1つの一般的な態様は、連続クエリ言語 (CQL) エンジンから作成されるスナップショットを管理するための方法であって、コンピューティングデバイスが、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを備える。この方法はさらに、コンピューティングデバ

40

50

イスが、CQLエンジンを用いて、入力イベントを処理して、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を備える。この方法はさらに、コンピューティングデバイスが、CQLエンジンによって実現されるスナップショット管理アルゴリズムを用いて、アプリケーションに関連する出力イベントのセットに少なくとも部分的に基づいて、システムの現在の状態のスナップショットを生成することを備える。この方法はさらに、コンピューティングデバイスが、システムの現在の状態のスナップショットに関連付けられるスナップショット情報にアクセスするために第1のディレクトリ構造を生成することを含む。この方法はさらに、コンピューティングデバイスが、システムの現在の状態に関連付けられるスナップショットのリストを生成するために第2のディレクトリ構造を生成することを含む。この方法はさらに、コンピューティングデバイスが、システムの現在の状態に関連付けられるスナップショットのリストを取得、追加、またはクリーンにするためにスナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスを決定することを備える。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

【0007】

実現例は以下の特徴の1つ以上を含んでもよい。マイクロバッチストリームは、1秒未満のマイクロバッチに離散化されたデータの連続ストリームである、方法。入力イベントを処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力イベントを処理することを含む、方法。この方法はさらに、コンピューティングデバイスが、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納することと、入力イベントのすべてが処理されたときに、コンピューティングデバイスが出力待ち行列内の出力イベントを送信することとを備える。マイクロバッチストリームは、マイクロバッチのデータまたはレジリエント分散データセット(RDD)を含む、方法。入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力各々に対して計算を実行することを含む、方法。この方法はさらに、コンピューティングデバイスが、連続クエリを受信することと、連続クエリに変換を適用して連続クエリについてクエリプランを生成することと、変換アルゴリズムを用いてクエリプランを変換して、変換されたクエリプランを生成することとをさらに備え、連続クエリはパターンマッチングを含む。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

【0008】

1つの一般的な態様は、コンピュータ実行可能命令を格納するように構成されたメモリと、メモリにアクセスし、コンピュータ実行可能命令を実行するように構成されたプロセッサとを含むシステムを含む。システムはさらに、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを含む。システムはさらに、CQLエンジンを用いて、入力イベントを処理して、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を含む。システムはさらに、CQLエンジンによって実現されるスナップショット管理アルゴリズムを用いて、アプリケーションに関連する出力イベントのセットに少なくとも部分的に基づいて、システムの現在の状態のスナップショットを生成することを含む。システムはさらに、システムの現在の状態のスナップショットに関連付けられるスナップショット情報にアクセスするために第1のディレクトリ構造を生成することを含む。システムはさらに、システムの現在の状態に関連付けられるスナップショットのリストを生成するために第2のディレクトリ構造を生成することを含む。システムはさらに、システムの現在の状態に関連付けられるスナップショットのリストを取得、追加、またはクリーンにするためにスナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスを決定することを含む。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

【0009】

実現例は以下の特徴の1つ以上を含んでもよい。マイクロバッチストリームは、1秒未満のマイクロバッチに離散化されたデータの連続ストリームである、システム。入力イベントを処理することは、変換されたクエリプランに少なくとも部分的に基づいて入力イベントを処理することを含む、システム。コンピュータ実行可能命令は、さらに、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納し、入力イベントのすべてが処理されたときに、出力待ち行列内の出力イベントを送信するよう実行可能である、システム。マイクロバッチストリームは、マイクロバッチのデータまたはレジリエント分散データセット(RDD)を含む、システム。入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力の各々に対して計算を実行することを含む、システム。コンピュータ実行可能命令は、さらに、連続クエリを受信し、連続クエリに変換を適用して連続クエリについてクエリプランを生成し、変換アルゴリズムを用いてクエリプランを変換して、変換されたクエリプランを生成するよう実行可能であり、連続クエリはパターンマッチングを含む、システム。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

10

【0010】

1つの一般的な態様は、プロセッサによって実行されると、プロセッサに動作を実行させるコンピュータ実行可能コードを格納するコンピュータ可読媒体を含み、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを含む。コンピュータ可読媒体はさらに、CQLエンジンを用いて、入力イベントを処理して、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を含む。コンピュータ可読媒体はさらに、CQLエンジンによって実現されるスナップショット管理アルゴリズムを用いて、アプリケーションに関連する出力イベントのセットに少なくとも部分的に基づいて、システムの現在の状態のスナップショットを生成することを含む。コンピュータ可読媒体はさらに、システムの現在の状態のスナップショットに関連付けられるスナップショット情報にアクセスするために第1のディレクトリ構造を生成することを含む。コンピュータ可読媒体はさらに、システムの現在の状態に関連付けられるスナップショットのリストを生成するために第2のディレクトリ構造を生成することを含む。コンピュータ可読媒体はさらに、システムの現在の状態に関連付けられるスナップショットのリストを取得、追加、またはクリーンにするためにスナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスを決定することを含む。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

20

30

【0011】

実現例は以下の特徴の1つ以上を含んでもよい。マイクロバッチストリームは、1秒未満のマイクロバッチに離散化されたデータの連続ストリームである、コンピュータ可読媒体。入力イベントを処理することは、変換されたクエリプランに少なくとも部分的に基づいて入力イベントを処理することを含む、コンピュータ可読媒体。動作は、さらに、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納することと、入力イベントのすべてが処理されたときに、出力待ち行列内の出力イベントを送信することとをさらに含む、コンピュータ可読媒体。マイクロバッチストリームは、マイクロバッチのデータまたはレジリエント分散データセット(RDD)を含む、コンピュータ可読媒体。入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力の各々に対して計算を実行することを含む、コンピュータ可読媒体。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

40

【0012】

1つ以上のコンピュータからなるシステムは、システムにインストールされ、動作中にシステムにアクションを実行させるソフトウェア、ファームウェア、ハードウェア、またはそれらの組合せを有することによって、特定の動作またはアクションを実行するように構

50

成することができる。1つ以上のコンピュータプログラムは、データ処理装置によって実行されるとデータ処理装置にアクションを実行させる命令を含むことによって、特定の動作またはアクションを実行するように構成することができる。1つの一般的な態様は完全にステートフルなクエリ処理をサポートするようマイクロバッチ方式ストリームを処理するための方法を含み、コンピューティングデバイスが連続クエリを受信することを備える。この方法はさらに、コンピューティングデバイスが、連続クエリに対して変換を適用して、連続クエリについてクエリプランを生成することを備える。この方法はさらに、コンピューティングデバイスが、モニタリング変換プロセスを用いて連続クエリを監視することを備える。この方法はさらに、コンピューティングデバイスが、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを備える。この方法はさらに、コンピューティングデバイスが、モニタリング変換プロセスに少なくとも部分的に基づいて、マイクロバッチストリームの入力イベントを処理して、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を備える。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

【0013】

実現例は以下の特徴の1つ以上を含んでもよい。処理は連続クエリ処理エンジンを用いて実行され、処理は出力イベントを生成するために入力イベントの各々を増分的に処理することを含む、方法。変換は有向非巡回グラフ（DAG）変換である、方法。この方法はさらに、コンピューティングデバイスが、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納することと、入力イベントのすべてが処理されたときに、コンピューティングデバイスが出力待ち行列内の出力イベントを送信することとを備える。マイクロバッチストリームはマイクロバッチのデータまたはレジリエント分散データセット（RDD）を含み、DAG変換は頂点と辺とのセットであり、頂点はRDDを表し、辺はRDDに適用されるべき操作を表す、方法。入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力の各々に対して計算を実行することを含む、方法。連続クエリはパターンマッチングを含む、方法。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

【0014】

1つの一般的な態様は、コンピュータ実行可能命令を格納するように構成されたメモリと、メモリにアクセスし、コンピュータ実行可能命令を実行するように構成されたプロセッサとを含むシステムを含む。システムはさらに、連続クエリを受信することを含む。システムはさらに、連続クエリに対して変換を適用して、連続クエリについてクエリプランを生成することを含む。システムはさらに、モニタリング変換プロセスを用いて連続クエリを監視することを含む。システムはさらに、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを含む。システムはさらに、モニタリング変換プロセスに少なくとも部分的に基づいて、マイクロバッチストリームの入力イベントを処理して、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を含む。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

【0015】

実現例は以下の特徴の1つ以上を含んでもよい。処理は連続クエリ処理エンジンを用いて実行され、処理は出力イベントを生成するために入力イベントの各々を増分的に処理することを含む、システム。変換は有向非巡回グラフ（DAG）変換である、システム。コンピュータ実行可能命令は、さらに、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納し、入力イベントのすべてが処理されたときに、出力待ち行列内の出力イベントを送信するよう実行可能である、システム。マイクロバッチストリームはマイクロバッチのデータまたはレジリエント分散データセット（RDD）を含み、DAG変換

10

20

30

40

50

は頂点と辺とのセットであり、頂点はRDDを表し、辺はRDDに適用されるべき操作を表す、システム。入力イベントの各々を処理することは、変換されたクエリプランに少なくとも部分的に基づいて、入力の各々に対して計算を実行することを含む、システム。連続クエリはパターンマッチングを含む、システム。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

【0016】

1つの一般的な態様は、プロセッサによって実行されると、プロセッサに動作を実行させるコンピュータ実行可能コードを格納するコンピュータ可読媒体を含み、連続クエリを受信することを含む。コンピュータ可読媒体はさらに、連続クエリに対して変換を適用して、連続クエリについてクエリプランを生成することを含む。コンピュータ可読媒体はさらに、モニタリング変換プロセスを用いて連続クエリを監視することを含む。コンピュータ可読媒体はさらに、アプリケーションに関連する入力イベントのマイクロバッチストリームを受信することを含む。コンピュータ可読媒体はさらに、モニタリング変換プロセスに少なくとも部分的に基づいて、マイクロバッチストリームの入力イベントを処理して、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を含む。この態様の他の実施形態は、各々が方法のアクションを実行するように構成される、対応するコンピュータシステム、装置、および1つ以上のコンピュータ記憶装置に記録されるコンピュータプログラムを含む。

【0017】

実現例は以下の特徴の1つ以上を含んでもよい。処理は連続クエリ処理エンジンを用いて実行され、処理は出力イベントを生成するために入力イベントの各々を増分的に処理することを含む、コンピュータ可読媒体。変換は有向非巡回グラフ(DAG)変換である、コンピュータ可読媒体。動作は、さらに、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納することと、入力イベントのすべてが処理されたときに、出力待ち行列内の出力イベントを送信することとをさらに含む、コンピュータ可読媒体。マイクロバッチストリームはマイクロバッチのデータまたはレジリエント分散データセット(RDD)を含み、DAG変換は頂点と辺とのセットであり、頂点はRDDを表し、辺はRDDに適用されるべき操作を表す、コンピュータ可読媒体。連続クエリはパターンマッチングを含む、コンピュータ可読媒体。記載される技術の実現例は、ハードウェア、方法もしくはプロセス、またはコンピュータアクセス可能媒体上のコンピュータソフトウェアを含み得る。

【0018】

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。

【図面の簡単な説明】

【0019】

【図1】本開示の実施形態によるイベント処理ネットワークを表現する図である。

【図2】本開示の実施形態によるイベント処理システムの単純化されたハイレベル図を示す。

【図3】マイクロバッチ方式ストリーム処理を用いたステートフル処理用に構成されたストリーム処理アプリケーションを本開示の実施形態に従って実施することができる例示的なシステムまたはアーキテクチャである。

【図4】本開示の実施形態によるマイクロバッチ方式ストリームの処理を示すフローチャートである。

【図5】本開示の実施形態に従ってCQLエンジントラッカーが実施される例示的なシステムまたはアーキテクチャである。

【図6A】本開示の実施形態に従って実施されるMapディレクトリ構造の例示的なデー

10

20

30

40

50

タ構造である。

【図 6 B】本開示の実施形態に従って実施される Map ディレクトリ構造の例示的なデータ構造である。

【図 7】本開示の実施形態によるマイクロバッチ方式ストリームの処理を示すフローチャートである。

【図 8】本開示の実施形態によるマイクロバッチ化の処理を説明するフローチャートを示す。

【図 9】本開示の実施形態を実施するための分散型システムの簡略図を示す。

【図 10】本開示の実施形態による、実施形態システムの 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をクラウドサービスとして提供することができるシステム環境の 1 つ以上のコンポーネントの簡略ブロック図である。

【図 11】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を示す。

【発明を実施するための形態】

【0020】

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。

【0021】

複合イベント処理（CEP）の概要

複合イベント処理（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（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。

【0022】

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。

【0023】

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す

10

20

30

40

50

るために必要な可視性を提供する。

【0024】

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。

10

【0025】

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。

20

【0026】

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。

【0027】

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。

30

40

【0028】

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。

【0029】

50

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。

【0030】

...

```
( <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。

【0031】

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。また、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。

【0032】

一部の局面において、CQLエンジンは、完全に発達したクエリ言語を含み得る。このようなことから、ユーザは、クエリに関して演算を特定し得る。加えて、CQLエンジンは、メモリの最適化、クエリ言語特性の利用、演算子の共有、リッチパターンマッチング、リッチ言語構成などのために設計され得る。加えて、一部の例において、CQLエンジンは、過去データおよびストリーミングデータの両方を処理し得る。たとえば、ユーザは、

10

20

30

40

50

カリフォルニアの販売が特定のターゲットを超えた時に警告を送るようにクエリを設定し得る。したがって、一部の例において、警告は、過去の販売データおよび受信中のライブ（すなわち、リアルタイム）販売データに少なくとも一部基づき得る。

【0033】

一部の例において、CQLエンジンまたは以下に記載されるコンセプトの他の特徴は、リアルタイムで過去のコンテキスト（すなわち、ウェアハウスデータ）と受信中のデータとを結合するように構成され得る。したがって、一部の例において、本開示は、データベースに記憶される情報とインフライト情報との間の境界を記載し得る。データベースに記憶される情報およびインフライト情報の両方は、BEデータを含み得る。このようなことから、データベースは、一部の例において、BEサーバーであり得る、または任意のタイプのデータベースであり得る。さらに、一部の例において、本開示の特徴は、ユーザがどのようにプログラムするのか、またはそれ以外にどのようにコードを書くのかを知らなくとも上記の特徴の実施を可能とし得る。言い換えると、特徴は、多機能ユーザインターフェイス（UI）において提供され得る、または過去データとリアルタイムのデータとの結合を非開発者が実施することを可能とする態様で提供され得る。

【0034】

一部の例において、上記のコンセプトは、複合イベント処理に関連付けられるリッチリアルタイム連続イベント処理能力を活用するために利用され得る。限定されないが、アーカイブされたリレーションなどのいくつかの特徴がサポートされ得る。このようなことから、このような特徴（たとえば、リッチリアルタイム連続イベント処理）を活用するために、システムは、リレーショナルデータのスタートアップ状態およびランタイム状態に透明に対処するように構成され得る。言い換えると、システムは、作成の瞬間においては空でないクエリ（すなわち、アーカイブされたリレーション）を管理するように構成され得る。

【0035】

一部の例において、アーカイブされたリレーションが利用され得る。このようなことから、アーカイブされたリレーションに基づくことを示すクエリをCQLエンジンが確認した場合、そのアーカイブされたリレーションは、たとえば、過去のコンテキストについてクエリを呼び出すことができる特定のエンティティがあることを示し得る。一部の例において、データ定義言語（DDL）は、照会をどのように行なうか、テーブル内の重要なカラムは何か、および／または残りのデータをどこへ送信するかなど、アーカイブされたリレーションについての注釈を示し得るが、これらに限定されない。一部の例において、ひとたびクエリがCQLエンジンにおいて構築されると（たとえば、グラフとして）、システムはクエリグラフを分析し得る。加えて、一部の局面において、「distinct」、「group aggr」、「pattern」、および／または「group by」のような、ステートフルな特定の演算子がある。しかしながら、ステートレスな演算子は、入力を取り、それをたとえば下流の演算子などの親に送るのみであり得る。このため、1つの手法は、このテーブル全体をここに記憶することである。しかしながら、アーカイブされたリレーションを利用することにより、システムは、クエリグラフを分析し、アーカイブを照会するために使用することができる最低のステートフル演算子がどれかを決定し得る。一部の例において、システム（または1つ以上のコンピュータ実行方法）は、グラフを横断しながら到達した最低のステートフル演算子において状態を検索し得る。たとえば、クエリグラフは、ソースからのトポロジー順序において分析され得る。この第1のステートフル演算子に少なくとも一部基づき、CQLエンジンは、アーカイブされたリレーションに対して定義されるクエリについての演算子の状態を初期化するために、取得されるデータの最適量を判定し得る。

【0036】

少なくとも1つの非限定的な例において、リレーションおよび／またはソースのようなソース演算子は、トポロジー横断線において最初に置かれ得て、クエリ出力および／またはルートは最後に置かれ得る。たとえば、CQLクエリが、select sum(c1) from R1 where c2>c25のような場合、このクエリについてのプランは、RelationSource→SELECT→GroupAggrのようになり得る。したがって、トポロジー順序に従うとともに、Relatio

10

20

30

40

50

nSourceおよびSELECTが両方ともステートレスであることから、最低のステートフル演算子はGroupAggrであり得る。この方法により、クエリのステートフル演算子（この例では、GroupAggr）は、クエリエンジンがストリーミングデータを受信する前にデータストアから過去データをクエリエンジンに集合させることを可能にし得る。これは、アーカイブされたリレーションをクエリが分析しており、そのアーカイブされたリレーションがそのように示されているという事実になくとも一部基づいて可能となり得る。

【0037】

一部の例において、所与のアーカイブされたリレーションのウィンドウサイズは、ユーザによって指定され得る。一部の局面において、ウィンドウは、アーカイブされたリレーションに関連して、受信中のストリーミングされたコンテンツを分析またはそれ以外に評価するクエリグラフにおけるノードを含み得る。言い換えると、ウィンドウは、クエリエンジンによって分析および／または処理されるストリーミングされたコンテンツの量、および／またはアーカイブされたリレーションに含まれる過去データの量を定義し得る。

【0038】

高いレベルにおいて、ひとたびウィンドウがStreamに対して適用されると、それはRelationとなり、リレーショナルデータベースと同様に、通常のリレーショナルロジックが適用され得る。タプルが到着してウィンドウを離れると、考慮中のRelationは、それに対してコンパイルされるクエリによって変化し、同時に結果を発する。CQLは、RANGE（ナノ秒粒度に至る）、ROWS、PARTITION BY、および拡張可能ウィンドウをサポートし得る。これらのウィンドウは、ストリームからリレーションへの演算子（stream-to-relation operators）の例である。他方、ISTREAM（すなわち、挿入ストリーム）、DSTREAM（すなわち、削除ストリーム）、およびRSTREAM（すなわち、リレーションストリームは、リレーションからストリームへの演算子（relation-to-stream operators）である。一部の例において、ユーザ、開発者、および／または管理者は、クエリエンジンまたはクエリエンジンを操作またはホスティングする1つ以上のコンピューティングシステムによって提供されるウィンドウサイズを（たとえば、UIを介して）設定し得る。一部の例において、ストリーム上のウィンドウは、時間ベースの範囲ウィンドウであり得る。たとえば、アーカイブされたリレーション上のコンフィギュラブル値ウィンドウは、ウィンドウサイズおよびウィンドウが計算される属性を使用して特定され得る。アーカイブされたリレーションの上部に特定されるコンフィギュラブル値ウィンドウがある場合、スナップショットクエリが演算され得て、ウィンドウ限界内のスナップショットタプルが出力され得る。加えて、状態の初期化後、値ウィンドウは、受信中のアクティブデータに対して適用され得る。一部の例において、ウィンドウ属性の値がウィンドウサイズよりも小さい現在イベント時間とは異なるウィンドウには受信中のアクティブデータのみが挿入される。

【0039】

さらに、いくつかの例では、本開示の特徴は、リアルタイムデータ分析をサポートするために、CQLエンジンおよび／またはCEPエンジンの連続クエリ処理能力を活用することもできる。いくつかの態様では、CQLエンジンおよび／またはCEPエンジンは、従来はストリーム指向型分析エンジンであったかもしれない。しかしながら、それは、耐久性のあるストア（たとえば、上記のアーカイブされたリレーション）によってバックアップされるストリーム指向のデータをサポートするように拡張することができる。たとえば、本開示は、耐久性のあるストア（データベースおよび／またはテーブル）であるデータオブジェクト（DO）の概念をサポートすることができる機能について説明する。DOに対して行われる変更は、事実上データストリームを作成する関心のあるリスナーに対して変更通知をブロードキャストさせ得る。このデータストリームは、実行中のクエリのサポートにおいてCQLエンジンおよび／またはCEPエンジンによって消費され得る。しかしながら、CQLエンジンおよび／またはCEPエンジンは、DOバックキングストア内の既存のデータを考慮に入れて設計されていなかった可能性がある。たとえば、CQLエンジンおよび／またはCEPエンジンは、CQLエンジンおよび／またはCEPエンジンで実行されているクエリの初期状態が現在DOバックキングストアにあるすべてのデータを含

10

20

30

40

50

むD Oの現在の状態を反映することを要求し得る。このクエリがそのように初期化されると、C Q Lエンジンおよび/またはC E Pエンジンは、従来のストリーム指向のスタイルにおいて、その時点から、D O変更通知のストリームに関わる必要があるだけである。

【0040】

いくつかの態様では、C Q Lエンジンおよび/またはC E Pエンジンはストリームまたはアーカイブされていないリレーションを従来の処理し得、したがって初期状態は存在しないかもしれない。たとえば、クエリがロードされ得、それは、実行および変更のリスニングを開始し得る。場合によっては、ユーザが棒グラフで州別に売上を求めた後、誰かが新たな販売を行なった場合、テーブルが更新され、ユーザはグラフの変更が売上に表示されるのを見ることを予想し得る。しかしながら、ダッシュボードを閉じて1週間後に戻って販売を改めると、ユーザは合計された売上データのテーブルに従って売上の合計を得ることを予想し得る。言い換えれば、クエリは、クエリをアーカイブの状態にしてから、アクティブな変更をリスニングする必要があるかもしれない。

10

【0041】

いくつかの態様では、たとえば、C Q Lエンジンは、アーカイブされたデータで事前に初期化されてもよい。初期化されると、C Q Lエンジンは、（たとえば、アーカイブからのデータの挿入、削除などのためにA P I呼び出しに少なくとも一部基づいて）変更通知についてJava Messaging Service (J M S) または他のメッセージングサービスを利用することができる。したがって、サービスはリスニングすることができ、J M Sがリスニングサービスがリスニングしているのと同じトピックでパブリッシュする場合、それはデータを受信し得る。サービスは、誰がパブリッシュしているのか、またはそれらがそうしているのか否かを知る必要はない。リスニングサービスはただリスニングすることができ、何かが起こった場合、リスニングサービスはそれを聞き得る。いくつかの例では、これは、持続性の、たとえばそのコンシューマからの分離方法である。さらに、いくつかの例では、アラートエンジンは、アラートエンジンが聞くもの、潜在的には、そしてさらには、リスナーに関連するプロセスクエリを聴取しているかもしれないS Q Lエンジンに基づいて、アラートを生成し得る。

20

【0042】

いくつかの例では、C Q L、S Q L、および/またはC E Pエンジンでクエリを開始し得、アーカイブデータを取得して（たとえば、ポンプをプライミングし）次いでこれらのJ M Sメッセージを聞き始めるよう、命令を構成してもよい。しかしながら、挿入、削除などが多数あると、大量の情報が含まれる可能性がある。さらに、リスナーがメッセージを聞くまでにタイムラグがあることがあり、リスニングは、いくつかの例では、飛び込んで、アーカイブに照会し、戻ってきて、リスニングを開始し得る。したがって、イベントの喪失および/または二重カウントの可能性がある。

30

【0043】

さらに、エンジンが単にクエリを実行する場合、クエリを実行している間にさまざまなものがJ M Sに入り、エンジンがリスニングしていなかったところでパブリッシュされ得る。そのため、リスナーを最初にセットアップし、アーカイブクエリを実行してから戻って実際に待ち行列から引き出して、何も見逃さないように、エンジンを構成してもよい。したがって、J M Sはさまざまなものを待ち行列に入れ得、そしてそれらがバックアップされる場合、エンジンがクエリを行なっている間は大丈夫であり、なぜならばそれはあとでキャッチアップでき、それが同期しているかどうかを心配する必要はないからである。それがここになく、聞いている場合、それはそれを見逃すことはなく、そのリスナーが確立されている限り、エンジンが戻ってくるまでただ待ち行列に入れられる。

40

【0044】

さらに、いくつかの例では、システムコラムをユーザのデータに追加することができる。このシステムコラムは、ダブルカウントおよび/または喪失操作問題を処理しようとするトランザクションIDを示すためのものであってもよい。しかしながら、他の例では、システムは、トランザクションコンテキストテーブルを提供するか、そうでなければ生成す

50

ることができる。さらに、2つの追加のコラムTRANSACTION_CIDおよびTRANSACTION_TIDが存在してもよい。コンテキストテーブルは、最後にコミットされたトランザクションIDのスレッド（コンテキスト）様式を知るために、持続性サービスによって常に維持されてもよい。トランザクションIDは、スレッド（コンテキスト）の昇順でコミットされることが保証され得る。たとえば、サーバは、起動すると、持続性サービスを実行してもよい。各々は、事前初期化された情報のデータがJMSを通過したすべてのデータを含むかどうかを判断するためのコンテキストIDとトランザクションIDとのセットを割り当てることができる。さらに、いくつかのケースでは、（JTAに準拠して、および/または高可用性（HA）を実現するために、複数の出力サーバを利用することができ、各サーバは、他のサーバによって管理される他のテーブルとは完全に別個のコンテキスト/トランザクションテーブルの単一セットを管理してもよい。

10

【0045】

いくつかの実施形態では、連続的な（たとえばCQL）クエリが作成または登録されると、それは構文および意味論解析を受けてもよく、その最後に論理クエリプランが作成される。たとえば、"alter query <queryname> start" DDLを発行してCQLクエリを開始すると、論理クエリプランは物理クエリプランに変換されてもよい。一例では、物理クエリプランは、物理演算子の有向非循環グラフ（DAG）として表されてもよい。次に、物理演算子を実行演算子に変換して、そのCQLクエリの最終クエリプランに到達してもよい。CQLエンジンへの入来イベントは、ソース演算子に到達し、最終的に、演算子のイベントでの処理を実行し、適切な出力イベントを生成する方法で、演算子とともに下流に移動する。

20

【0046】

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。

30

【0047】

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。

40

【0048】

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。

【0049】

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。

【0050】

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

50

る能力。

【0051】

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。

【0052】

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。

【0053】

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。

【0054】

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。

【0055】

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。

【0056】

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はほぼ常にその後2秒以内に発生する。

【0057】

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。

【0058】

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。

【0059】

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

10

20

30

40

50

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。

【0060】

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。

【0061】

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。Web 2.0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ン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。

【0062】

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実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。

【0063】

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。

【0064】

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。

【0065】

マイクロバッチベースのストリーム処理システムにおけるイベントごとのイベント処理のためのフレームワーク

近年、データストリーム管理システム（DSM）が開発されてきており、これは潜在的に無限のリアルタイムデータストリーム上で連続的にクエリを実行することができる。新たなDSMの中で、これらのシステムは一般に、単一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。そのようなシステムの例は、Spark（登録商標）プラットフォーム上で動作するSpark Streamingアプリケーションである。マイクロバッチ方式ストリーム処理は、ステートフル処理が複雑になる可能性があるシステムの設計の性質上、いくつかの欠点がある。そのような欠点の1つは、「パターンマッチング」操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ることが望ましい重要な機能であり、パターンマッチングでは、ステートマシンを実行してアンバウンド（unbounded）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が必要である。

【0066】

前述のOracle Stream Analytics Platformを使用することで、提案されたソリューションはステートフル処理をマイクロバッチ方式ストリーム処理と組み合わせる。基本的に、このソリューションは複合イベント処理（CEP）とマイクロバッチ方式ストリーム処理とを組み合わせる。ステートフル処理は、連続クエリ言語（Continuous Query Language）（CQL）で書かれた連続クエリ処理エンジンであるCQLエンジンによって処理される。完全にステートフルなクエリ処理をサポートするために、一実施形態では、CQLクエリエンジンはマイクロバッチ方式ストリーム処理内に追加される。

【0067】

一実施形態では、有向非巡回グラフ（DAG）変換に追加することができるCQL変換アルゴリズムが開示される。特定の实施形態において、変換アルゴリズムは以下のように実施されてもよい。（i）ストリーム処理アプリケーションからのドライバは、決して戻ることのない長期実行タスクとしてエグゼキュータの1つ以上にCQLエンジンを起動する。（ii）CQLエンジンは動作し続け、クエリ状態を維持する。（iii）各マイクロバッチジョブでは、CQL変換はマイクロバッチジョブの一部として実行される。（iv）CQL変換が実行されると、マイクロバッチの入力イベントはCQLエンジンに送信される。CQLエンジンは、マイクロバッチ内の各イベントをイベントごとに処理し、クエリに対してインクリメンタルコンピュテーション（incremental computation）を実行し、出力イベントを作成する。（v）マイクロバッチ内のイベントが処理されている間に出力イベントは待ち行列内に捕捉される。（vi）マイクロバッチ内の各すべてのイベントがCQLエンジンで完了した後、結果待ち行列内の出力イベントはCQL変換の結果として返される。（vii）CQL変換の次の変換は、追加の変換なしで出力イベントを消費できる。

【0068】

開示されたCQL変換アルゴリズム／プロセスは、一般的なストリーム処理システムにお

10

20

30

40

50

いてCQLを処理するためにCQL変換を追加する能力を提供する。加えて、CQLエンジンを使用することで、機能処理とステートフル処理とを組み合わせることができる。開示されたプロセスは、複合イベント処理を追加することによってマイクロバッチ方式ベースのストリーム処理のいくつかの欠点を解決する。また、CEP技術のインクリメンタルコンピューテーションを用いることで、分析の一部をより効率的に実行できる。

【0069】

図1は、本開示の一実施形態を組み込んでもよいイベント処理ネットワーク（EPN）を表現する図である。図1に示すように、EPN100は、各々がイベントストリーム内のイベントの処理において異なる役割を果たすいくつかのステージから構成されてもよい。イベントは定義上時間ベースであるため、ストリームはその意味においてイベントの自然な状態である。それは、イベントデータがOracle Event Processingアプリケーションにどのように到着するかである。Oracle Event Processingでイベントを処理するために、EPN100などのEPNをコアとするアプリケーションが構築される。EPN100は、イベントデータの受信から、データの照会、イベントに関して発見されたことに基づいたロジックの実行まで、各々がイベントの処理において異なる役割を果たすステージで構成されている。アプリケーションは生のイベントデータを受信し、そのデータをイベントタイプにバインドしてから、処理のためにステージ間でイベントをルーティングする。EPNにおける接続されたステージは、EPNを通過するイベントに対してさまざまな種類のコードを実行する方法を提供する。ステージの種類は、アダプタ、プロセッサ、およびビーンを含むことができる。より具体的には、様々な実施形態において、EPN100は、イベントを受信するイベントソース105と、ステージを接続するチャンネル110と、連続クエリ言語（CQL）のクエリコードを含むCQLプロセッサなどのプロセッサ115と、一般的な処理ロジックを実行するビーン120、コード125、および／またはシンク130とを含む。本明細書に記載されているように、イベントのストリームは、次々と、時間による連続した順序にある。

【0070】

いくつかの実施形態では、イベントソース105は、アダプタ（たとえば、JMS、HTTPおよびファイル）、チャンネル、プロセッサ、テーブル、キャッシュなどが挙げられるが、それらに限定されるものではない。たとえば、イベントソース105は1つ以上のアダプタを含んでもよい。1つ以上のアダプタは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してもよい。1つ以上のアダプタは、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てもよく、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照会可能な正規化された形態に変換する役割を担う。例えば、アダプタは、イベントデータを受信し、それをイベントタイプインスタンスにバインドし、次いでそのイベントをプロセッサ115に渡すことができる。1つ以上の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。チャンネル110はイベント処理終点の役割を果たす。とりわけ、チャンネル110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。プロセッサ115は、イベントデータに対するクエリの実行など、イベントデータに対するアクションを実行するように構成されたイベント処理エージェントであり得る。特定の実施形態では、プロセッサ115は、入力チャンネル（例えば、チャンネル110）によって提供されるイベントに作用する1つ以上のCQLクエリと関連付けることができるCQLプロセッサを備える。例えば、プロセッサのCQLコードは、（SQLコードがデータベース行に照会するように）イベントに照会することができ、データがEPN100を流れるときにデータ内の特定のパターンを探す。CQLプロセッサは、クエリ結果が書き込まれる出力チャンネル（例えば、チャンネル110）に接続されてもよい。例えば、パターン基準を満たすイベントは、（例えば、Javaで書かれた）ビーン120またはコード125に渡すことができ、そこでデータは、外部ソースから検索されたデータと共に計算に使用することができる。さらに下流のビーン120またはコード125は、計算結果を用い、外部コンポーネントを用いてプロセスを実行するこ

10

20

30

40

50

とができる。ビーン120またはコード125は、出力チャネルをリスニングするように登録され得、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る。いくつかの実施形態では、ビーン120の処理ロジックは、Java（登録商標）またはplain-old-java-object（POJO）などのプログラミング言語で書くことができる。いくつかの実施形態では、処理ロジックは、ビーンがオラクルCEPによって管理できるようにオラクルCEPイベントビーンAPIを用い得る。EPN100内でイベントを受信または送信するように設計された任意のコンポーネント（EPNステージなど）は、そうするために特に実装されるビーンであってもよい。イベントを受信することができるコンポーネントはイベントシンク130として知られ、イベントを送信するコンポーネントはイベントソース105として知られている。1つのコンポーネントは、イベントソースとシンクとの両方になることができる。CQLプロセッサの基盤となるアダプタやコンポーネントなど、Oracle Event Processingに含まれる記載のステージコンポーネントは、必要な機能をすでにサポートしている。開発者は、OEP APIからインターフェイスを実装することで、イベントシンクおよびソースサポートを、彼らが書くビーン、新たなアダプタ、およびその他のコードに追加できる。

【0071】

図2は、本開示の一実施形態を組み込むことができるイベント処理システム200の単純化されたハイレベル図を示す。イベント処理システム200は、1つ以上のイベントソース（204，206，208）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）202（CQサービス202とも称される）と、1つ以上のイベントシンク（210，212）とを備え得る。イベントソースは、EPS202によって受信されるイベントストリームを生成する。EPS202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し得る。たとえば、図2に示されるように、EPS202は、イベントソース204から第1の入力イベントストリーム214を受信し、イベントソース206から第2の入力イベントストリーム216を受信し、イベントソース208から第3のイベントストリーム218を受信する。1つ以上のイベント処理アプリケーション（220，222および224）がEPS202上にデプロイされてEPS202によって実行され得る。EPS202によって実行され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スニングし、注目イベントとして入力イベントストリームから1つ以上のイベントを選択する処理論理に基づいて、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され得る。次いで、当該注目イベントは、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（210，212）に送信され得る。たとえば、図2では、EPS202は、第1の出力イベントストリーム226をイベントシンク210に出力し、第2の出力イベントストリーム228をイベントシンク212に出力する。特定の实施形態では、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、他のコンポーネントに対する変更を生じさせることなくこれらのコンポーネントのうちのいずれかを追加または除去することができるように互いに分離される。

【0072】

一実施形態では、EPS202は、共有サービスを有する、Equinox OSGiに基づくものなどの軽量Java（登録商標）アプリケーションコンテナを備えるJavaサーバとして実現され得る。いくつかの実施形態では、EPS202は、たとえばJRockit Real Timeを使用してイベントを処理するための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。また、EPS202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、オラクルCEPビジュアライザおよびオラクル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のエンドツーエンドJavaイベント駆動アーキテクチャ（Event-Driven Architecture：EDA）開発プラットフォーム）を提供し得る。

【0073】

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スニングして

、1つ以上の入力イベントストリームから1つ以上の注目イベントを選択するための論理（たとえば、クエリ）を実行し、選択された注目イベントを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2は、1つのこのようなイベント処理アプリケーション220のためのドリルダウンを提供する。図2に示されるように、イベント処理アプリケーション220は、入力イベントストリーム218をリスニングして、入力イベントストリーム218から1つ以上の注目イベントを選択するための論理を備える連続クエリ230を実行し、選択された注目イベントを出力イベントストリーム228を介してイベントシンク212に出力するように構成される。イベントソースの例としては、アダプタ（たとえば、JMS、HTTPおよびファイル）、チャンネル、プロセッサ、テーブル、キャッシュなどが挙げられるが、それらに限定されるものではない。イベントシンクの例としては、アダプタ（たとえば、JMS、HTTPおよびファイル）、チャンネル、プロセッサ、キャッシュなどが挙げられるが、それらに限定されるものではない。

10

【0074】

図2におけるイベント処理アプリケーション220は、1つの入力ストリームをリスニングして、選択されたイベントを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、これは限定的であるよう意図されるものではない。代替的な実施形態では、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スニングして、モニタリングされたストリームからイベントを選択し、選択されたイベントを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され得る。同一のクエリを2つ以上のイベントシンクおよび異なるタイプのイベントシンクに関連付けることができる。

20

【0075】

無限の性質のために、イベントストリームを介して受信されるデータの量は、一般に非常に多い。その結果、照会目的で全てのデータを格納またはアーカイブすることは、一般に非現実的であり、望ましくない。イベントストリームの処理は、受信した全てのイベントデータを格納する必要なしにイベントがEPS202によって受信されるときにリアルタイムでイベントの処理を必要とする。したがって、EPS202は、受信した全てのイベントを格納する必要なしにイベントがEPS202によって受信されるときにイベントの処理を実行することを可能にする特別な照会機構を提供する。

30

【0076】

イベント駆動アプリケーションは、ルール駆動であり、これらのルールは、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され得る。連続クエリは、クエリ処理の結果として、どのイベントを注目イベントとして選択して出力すべきかを含む、受信したイベントに対して実行される処理を識別する命令（たとえば、ビジネス論理）を備え得る。連続クエリは、データストアに留まって、イベントの入力ストリームの処理およびイベントの出力ストリームの生成に使用され得る。連続クエリは、一般に、フィルタリングおよび集計機能を実行して、入力イベントストリームから注目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送信イベントの数は、一般に、イベントが選択され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はるかに少なくなる。

40

【0077】

有限のデータセットに対して一度実行されるSQLクエリとは異なって、EPS202を有するアプリケーションによって特定のイベントストリームのために登録された連続クエリは、当該イベントストリームにおいてイベントが受信されるたびに実行され得る。連続クエリの実行の一部として、EPS202は、連続クエリによって指定された命令に基づいて、受信したイベントを評価して、連続クエリの実行の結果として、1つ以上のイベントを注目イベントとして選択すべきか否かを判断して出力する。

【0078】

連続クエリは、さまざまな言語を使用してプログラミングされ得る。特定の実施形態では

50

、連続クエリは、オラクル社によって提供されるCQLを使用して構成され、オラクルの複合イベント処理（CEP）製品提供によって使用され得る。オラクルのCQLは、イベントストリームに対して実行可能なクエリ（CQLクエリと称される）をプログラミングするために使用できる宣言型言語である。特定の実施形態では、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追加の構造を有するSQLに基づく。

【0079】

一実施形態では、イベント処理アプリケーションは、以下のコンポーネントタイプで構成され得る。

【0080】

（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。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照会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ン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。

10

【0081】

（2）イベント処理終点の役割を果たす1つ以上のチャンネル。とりわけ、チャンネルは、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。

【0082】

20

（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）は、正規化されたイベントデータをチャンネルから消費し、クエリを使用してそれを処理して注目イベントを選択し、選択された注目イベントを出力チャンネルに転送（または、コピー）するように構成される。

【0083】

（4）1つ以上のビーンは、出力チャンネルをリスニングするように構成され、出力チャンネル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（plain-old-Java-object：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など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。

30

【0084】

（5）イベントビーンは、出力チャンネルをリスニングするように登録され得て、出力チャンネル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、ビーンがオラクルCEPによって管理できるようにオラクルCEPイベントビーンAPIを使用し得る。

【0085】

一実施形態では、イベントアダプタは、イベントデータを入力チャンネルに提供する。入力チャンネルは、入力チャンネルによって提供されるイベント上で動作する1つ以上のCQLクエリに関連付けられたCQLプロセッサに接続される。CQLプロセッサは、クエリ結果が書込まれる出力チャンネルに接続される。

40

【0086】

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接続されるか、アプリケーションによって処理されるイベントタイプを記述するアセンブリファイルがイベント処理アプリケーションに対して提供され得る。イベントの選択のために連続クエリまたはビジネス論理を指定するために別々のファイルが提供されてもよい。

【0087】

図2に示されるシステム200は、図2に示されているコンポーネント以外の他のコンポーネントを有していてもよいということが理解されるべきである。さらに、図2に示され

50

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 200 は、図 2 に示される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 200 は、図 1 に記載されるサービスプロバイダコンピュータ 106、パーソナルコンピュータ、ポータブルデバイス（たとえば、携帯電話またはデバイス）、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。いくつかの他の実施形態では、システム 200 は、システム 200 の 1 つ以上のコンポーネントがクラウド内の 1 つ以上のネットワークにわたって分散される分散型システムとして構成されてもよい。

10

【0088】

図 2 に示されるコンポーネントのうちの 1 つ以上は、ソフトウェア、ハードウェア、またはそれらの組み合わせで実現されてもよい。いくつかの実施形態では、ソフトウェアは、メモリ内に格納されてもよく（たとえば、非一時的なコンピュータ可読媒体）、メモリデバイス上に格納されてもよく、または何らかの他の物理メモリ上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。

【0089】

図 3 は、マイクロバッチ方式ストリーム処理を用いたステートフル処理のために構成されたストリーム処理アプリケーション 300 を実装することができる例示的なシステムまたはアーキテクチャである。様々な実施形態において、ストリーム処理アプリケーション 300 は 1 つ以上のデータストリーム 305 を含む。データストリーム 305 は、しばしばもっぱら新たな要素の挿入を通して、絶えず変化しているデータを表す。多くのタイプのアプリケーションは、センサデータアプリケーション、財務ティッカー、ネットワーク性能測定ツール、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、ならびにクリックストリーム分析ツールを含む、データセットとは対照的なデータストリーム 305 を生成する。いくつかの実施形態では、Spark Streaming は、Spark 用の増分的マイクロバッチ方式ストリーム処理フレームワーク 310 であり、Spark Streaming は、連続データストリームを扱う複雑さを隠し、プログラマのためにそれを一度に 1 つの RDD を使用することと同じくらい容易にする離散化されたストリーム (Dstream) 315 と呼ばれるデータ抽象化を提供する。Dstream は基本的に、バッチのために入力ストリームから受け取ったデータを要素とするレジリエント分散データセット (RDD) のストリームである（おそらくはウィンドウ化されるかまたはステートフルな演算子によって範囲が拡張される）。RDD は Spark の基本的なデータ構造である。それはオブジェクトの不変の分散コレクションである。RDD における各データセットは論理パーティションに分割され、それらはクラスタの異なるノードで計算されてもよい。RDD には、ユーザー定義クラスを含む、Python、Java、または Scala オブジェクトのうちの任意のタイプを含むことができる。ストリーム処理フレームワーク 310 によって実行されるマイクロバッチ化では、バッチは本質的に一度に 1 つの RDD である。したがって、データストリーム 305 を一度に 1 レコードずつ処理する代わりに、Spark Streaming のレシーバは、データを並列で受け取り、それを Spark のワーカーノードのメモリにバッファする。その後、レイテンシ最適化された Spark エンジン短いタスク（数十ミリ秒）を実行してバッチを処理して結果を他のシステムに出力する。

20

30

40

【0090】

ストリーム処理アプリケーション 300 は、CQL エンジン 320 をさらに含む。離散化されたストリーム 315 内などのアンバウンドのイベントストリームからパターンを検出するために必要なステートフル処理は、CQL で書かれた連続クエリ処理エンジンである CQL エンジン 320 によって処理される。完全にステートフルなクエリ処理をサポートするために、一実施形態では、CQL エンジン 320 は、離散化されたストリーム 315

50

内のイベントまたはデータに対するマイクロバッチ方式ストリーム処理に加えられる。CQLエンジン320は、DAG変換に追加され得るCQL変換アルゴリズムを用いて、CQLプロセッサなどのプロセッサ内でステートフルクエリ処理の実行を最適化する。変換アルゴリズムは、離散化されたストリーム315を入力として受け取り、CQLプロセッサと組んで結果325を生成するのを助ける。理解されるように、結果を得るためにRDD上で実行されることができ2つのタイプの操作がある：(i) マップ、フィルタのような、別のRDD325をもたらす変換、および(ii) カウントのような、出力をもたらすアクション。スパークジョブは通常、RDD上で変換およびアクションを実行するタスクのDAGから構成される。CQLプロセッサコードは、データがEPN（例えば、図1に関して説明したようにEPN100）を通して流れるときにデータ内の特定のパターンを探して、離散化されたストリーム315内のイベントに照会することができる。パターン基準を満たすイベントは、結果325としてビーン、コード、またはシンクに渡すことができ、そこで最終的に結果325は用いられ、外部ソースから検索されたデータを用いた計算において入力330として渡すことができる。

【0091】

図4、図7および図8は、いくつかの実施形態による完全にステートフルなクエリ処理をサポートするためにマイクロバッチ方式ストリームを処理するための技法を示す図である。個々の実施形態は、フローチャート、フロー図、データフロー図、構造図またはブロック図として示されるプロセスとして説明されてもよい。フローチャートは動作をシーケンシャルなプロセスとして説明することができるが、動作の多くは並列または同時に実行されてもよい。さらに、動作の順序は並べ替えられてもよい。プロセスは、その動作が完了すると終了するが、図に含まれていないさらなるステップを有し得る。プロセスは、メソッド、関数、プロシージャ、サブルーチン、サブプログラムなどに対応し得る。プロセスが関数に対応している場合、その終了は、その関数を呼び出し関数またはmain関数へと戻すことに対応し得る。

【0092】

図4、図7、および図8において示されるプロセスおよび／または動作は、1つ以上の処理ユニット（たとえば、プロセッサコア）によって実行されるソフトウェア（たとえば、コード、命令、プログラム）、ハードウェア、またはそれらの組み合わせで実装され得る。ソフトウェアは、メモリ（例えば、メモリデバイス上、非一時的コンピュータ可読記憶媒体上）に格納することができる。図4、図7、および図8における特定の一連の処理ステップは、限定的であることを意図しない。代替の実施形態に従って、他の一連のステップも実行することができる。例えば、代替の実施形態では、上で概説したステップは異なる順序で実行されてもよい。さらに、図4、図7、および図8に示される個々のステップは、個々のステップに適切であるように様々なシーケンスで実行され得る複数のサブステップを含み得る。さらに、特定の用途に応じて追加のステップを追加または削除することができる。当業者は、多くの変形形態、修正形態、および代替形態を認識するであろう。

【0093】

図4は、本開示の実施形態によって実施される完全にステートフルなクエリ処理をサポートするためにマイクロバッチ方式ストリームの処理を示すフローチャート400を示す。いくつかの実施形態では、フローチャート400に示されているプロセスは、図1、図2および図3のイベント処理システムによって実施することができる。ステップ405で、連続クエリ言語で書かれた連続クエリ処理エンジンが起動される。いくつかの実施形態では、ストリーム処理アプリケーションからのドライバが、決して戻ることのない長期実行タスクとして1つ以上のエグゼキュータにCQLエンジンを起動する。CQLエンジンは動作し続け、マイクロバッチストリームに対するクエリ状態を維持する。ステップ410で、連続クエリが受信される。いくつかの実施形態では、クエリはパターン認識を含む。たとえば、CQLのMATCH_RECOGNIZE句およびその副次句を呼び出して、CQLクエリでパターン認識を実行できる。ステップ415で、連続クエリに対してクエリプランを生成するために、連続クエリに操作が適用される。クエリプラン（またはクエリ実行プラ

10

20

30

40

50

ン) は、クエリまたは連続クエリの処理のために、たとえばSQLリレーショナルデータベース管理システムにおいてデータにアクセスするために使用される順序付けられたステップのセットである。いくつかの実施形態では、操作はDAG変換であり、クエリプランはDAGクエリプランである。DAG変換は頂点と辺との集合であり、頂点はRDDを表し、辺はRDDに適用されるべき操作を表す。ステップ420において、クエリプランは、変換されたクエリプランを生成するために変換アルゴリズムを用いて変換される。様々な実施形態において、変換アルゴリズムはCQL変換である。たとえば、クエリプランを生成するためにマイクロバッチまたはRDDが操作される各インスタンスで、CQL変換が実行される。いくつかの実施形態では、変換されたクエリプランを生成するために、CQL変換がDAG変換に追加される。ステップ425で、アプリケーションに関連する入力イベントのマイクロバッチストリームが受信される。いくつかの実施形態では、Spark Streamingは、連続するデータストリームを1秒未満の小さいマイクロバッチつまりマイクロバッチストリームに離散させることができる。ステップ430で、入力イベントは、少なくとも部分的に、変換されたクエリプランに基づいて、処理され、アプリケーションに関連する出力イベントのセットが生成される。いくつかの実施形態では、処理は連続クエリ処理エンジンを用いて実行され、処理は出力イベントを生成するために入力イベントの各々を増分的に処理することを含む。たとえば、CQL変換などの変換アルゴリズムが実行されると、マイクロバッチの入力イベントはCQLエンジンに送信される。CQLエンジンは、マイクロバッチの各入力イベントをイベントごとに処理し、変換されたクエリプランに少なくとも部分的に基づいて、クエリについてマイクロバッチの各入力イベント上でインクリメンタルコンピューションを実行し、マイクロバッチにおける各入力イベントに対する出力イベントを作成する。したがって、ステートフル処理はCQLエンジンによって実行される。ステップ435で、アプリケーションに関連する出力イベントのセットが出力待ち行列に格納される。いくつかの実施形態では、マイクロバッチ内の残りのイベントがCQLエンジンによって処理されている間に、出力イベントが出力待ち行列に捕捉される。ステップ440で、マイクロバッチ内の各イベントが処理された後、出力待ち行列内の出力イベントが連続クエリの結果として返されおよび／または送信される。

【0094】

マイクロバッチベースのイベント処理におけるスナップショットおよびアプリケーション状態の管理

近年、データストリーム管理システム(DSM)が開発されてきており、これは潜在的に無限のリアルタイムデータストリーム上で連続的にクエリを実行することができる。新たなDSMの中で、これらのシステムは一般に、単一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。そのようなシステムの例は、Spark(登録商標)プラットフォーム上で動作するSpark Streamingアプリケーションである。マイクロバッチ方式ストリーム処理は、ステートフル処理が複雑になる可能性があるシステムの設計の性質上、いくつかの欠点がある。そのような欠点の1つは、「パターンマッチング」操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ることが望ましい重要な機能であり、パターンマッチングでは、ステートマシンを実行してアンバウンド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が必要である。

【0095】

完全にステートフルなクエリ処理をサポートするために、一実施形態では、ここに記載されるように、CQLクエリエンジンはマイクロバッチ方式ストリーム処理内に追加される。本質的に、このソリューションは複合イベント処理(CEP)とマイクロバッチ方式ストリーム処理とを組み合わせる。ステートフル処理は、連続クエリ言語(CQL)で書かれた連続クエリ処理エンジンであるCQLエンジンによって処理される。

【0096】

特定の状況では、クラスタ内に複数のCQLエンジンが存在する可能性があり、各エンジンは個別にチェックポイントを設定するために状態スナップショットを作成する必要がある

る。そのため、マイクロバッチ処理が完了した後にスナップショットの生成を調整し、スナップショットの保持など、スナップショットを管理する必要がある。

【0097】

図5は、本開示の実施形態による、スナップショット生成を調整し、スナップショットを管理するために、CQLエンジントラッカー605を実装することができる例示的なシステムまたはアーキテクチャ500である。様々な実施形態において、システムまたはアーキテクチャ500はリスナー510と通信するCQLエンジントラッカー505を含む。CQLエンジントラッカー505およびリスナー510は、ドライバ515内に配置することができる。CQLエンジントラッカー505は、CQLエンジン520から作成されたスナップショットを管理するように構成され、CQLエンジン520は、クラスタ内の1つ以上のエグゼキュータ525上にあり得る。特定の実施形態では、CQLエンジントラッカー505は、スナップショットを管理するために2つのディレクトリ構造、つまりSnapshot Map (スナップショットマップ) およびMap (マップ) を使用することができる。Snapshot Mapディレクトリ構造を用いて、特定のqueryid (クエリid)、パーティションおよび時間ならびにマップからスナップショット情報に直接アクセスできる。Snapshot Mapを用いて、回復またはクリーンアップすべきスナップショットを見つけることができる。Snapshot Mapディレクトリ構造の例示的なデータ構造を図6Aに示す。Mapディレクトリ構造: (queryid, partitionId, time) -> mark。Map: (queryid, partitionId) -> 逆順序でList of Snapshot(time, mark, fullFlag)。Mapディレクトリ構造の例示的なデータ構造を図6Bに示す。CQLエンジン520からのスナップショットおよびCQLエンジントラッカー505からのメタデータは、スナップショットストレージ530に書き込まれてもよい。

【0098】

一実施形態では、CQLエンジントラッカーはCQLエンジンと連携してスナップショット管理アルゴリズムを実装することができる。いくつかの実施形態では、スナップショット管理アルゴリズムは、スナップショットを追加するプロセス、スナップショットを取得するプロセス、およびスナップショットをクリーンにするプロセスを含み得る。いくつかの実施形態では、AddSnapshot Processは以下の操作を含む。(i) スナップショットを管理するために、一次構造は逆順序でlist of Snapshot(time, mark, full_flag)に対するPartitionKey(queryId, partitionId)のマップを用いる。(ii) CQLエンジンは、計算が終了した後にCQLエンジントラッカーに対するaddSnapshot RPCを呼び出し、queryId、partitionId、時間およびスナップショットマーク情報、ならびにfull_flagでスナップショットを作成する。(iii) AddSnapshotが呼び出される。(iv) PartitionKeyオブジェクトがqueryIdおよびpartitionIdで作成される。(v) マップ内にpartitionKeyに対するリストがない場合は、新しいリストが作成され、それ以外の場合は、既存のリストを使用する。(vi) Snapshotオブジェクトが時間、マーク、およびfull_flagで作成される。

【0099】

いくつかの実施形態では、GetSnapshots Processは、以下の操作を含む。(i) 状態を復元するためにスナップショットのリストを取得するために計算を開始する前に、CQL RDD (Resilient Distributed Dataset (レジリエント分散データセット)) がCQLエンジントラッカーに対してgetSnapshot RPC (Remote Procedure Call (リモートプロシージャコール)) をqueryId, partitionIdおよび時間で呼び出す。(ii) GetSnapshots PartitionKeyがqueryIdおよびpartitionIdで作成される。(iii) スナップショットが、partitionKeyを用いてスナップショットマップから検索される。(iv) 保存されたスナップショットマップがない場合は、empty[Snapshot]を返す。(v) 逆順リストの各スナップショットに対して、Stack[Snapshot]が作成される。(vi) スナップショット時間が(time - batchDuration) (時間 - バッチ継続期間) より小さい場合は、それをスタックに追加し、スタックをリストに変換して返す。

【0100】

いくつかの実施形態では、CleanSnapshots Processは、以下の操作を含む。(i) バッチが完了すると、スナップショットを削除しても安全であり得る。(ii) バッチが完了すると、onEndBatch (バッチの終わり) がジョブスケジューラから呼び出され、それは、バッチ時間でEndOfBatch RPC呼び出しを呼び出す。(iii) アルゴリズムは、フルスナップショットを除き、所与のバッチ時間前にすべてのスナップショットを削除する。(iv) スナップショットマップの各エントリおよびスナップショットリストの各スナップショットに対するCleanSnapshots。(v) (スナップショット時間がバッチ時間より短い場合) それをマップから削除し、スナップショットストレージからも削除する。

【0101】

図7は、本開示の実施形態によって実施される完全にステートフルなクエリ処理をサポートするためにマイクロバッチ方式ストリームの処理を示すフローチャート700を示す。いくつかの実施形態では、フローチャート700に示されているプロセスは、図1、図2、図3および図5のイベント処理システムによって実施することができる。ステップ705で、アプリケーションに関連する入力イベントのマイクロバッチストリームが受信される。いくつかの実施形態では、Spark Streamingは、連続するデータストリームを1秒未満の小さいマイクロバッチつまりマイクロバッチストリームに離散させることができる。ステップ710において、入力イベントは、アプリケーションに関連した出力イベントのセットを生成するために連続クエリ処理エンジンを使用して処理される。いくつかの実施形態では、処理は、入力イベントの各々を増分的に処理して出力イベントを生成することを含む。たとえば、CQL変換などの変換アルゴリズムが実行されると、マイクロバッチの入力イベントはCQLエンジンに送信される。CQLエンジンは、マイクロバッチの各入力イベントをイベントごとに処理し、変換されたクエリプランに少なくとも部分的に基づいて、クエリについてマイクロバッチの各入力イベント上でインクリメンタルコンピューテーションを実行し、マイクロバッチにおける各入力イベントに対する出力イベントを作成する。したがって、ステートフル処理はCQLエンジンによって実行される。ステップ715で、システムに関連するイベントの出力セットに少なくとも部分的に基づいてシステムの現在の状態のスナップショットが生成される。いくつかの実施形態では、スナップショットは、CQLエンジンによって実現されるスナップショット管理アルゴリズムを用いて生成される。特定の実施形態では、スナップショット管理アルゴリズムは、スナップショットを追加するプロセス、スナップショットを取得するプロセス、およびスナップショットをクリーンにするプロセスを含み得る。ステップ720で、システムの現在の状態のスナップショットに関連付けられるスナップショット情報にアクセスするために第1のディレクトリ構造が生成される。いくつかの実施形態では、第1のディレクトリ構造はSnapshot Mapディレクトリ構造である。ステップ725で、システムの現在の状態に関連付けられるスナップショットのリストを生成するために第2のディレクトリ構造が生成される。いくつかの実施形態では、第2のディレクトリ構造はMapディレクトリ構造である。ステップ730で、システムの現在の状態に関するスナップショットのリストを生成、追加、またはクリーンにするためにスナップショット管理アルゴリズムに少なくとも部分的に基づいてプロセスが決定される。いくつかの実施形態では、スナップショット管理アルゴリズムがスナップショットを追加するためのプロセスを含むとき、そのプロセスはシステムの現在の状態に関するスナップショットのリストを追加するよう決定される。いくつかの実施形態では、スナップショット管理アルゴリズムがスナップショットを取得するためのプロセスを含むとき、そのプロセスはシステムの現在の状態に関するスナップショットのリストを取得するよう決定される。いくつかの実施形態では、スナップショット管理アルゴリズムがスナップショットをクリーンにするためのプロセスを含むとき、そのプロセスはシステムの現在の状態に関するスナップショットのリストをクリーンにするよう決定される。理解されるように、プロセスは、図4に関して説明されたステップをさらに含み得、例えば、連続クエリ処理エンジンを起動し、連続クエリに操作を適用して連続クエリに対するクエリプランを生成し、クエリプランを変換して、変換されたクエリプランを生成し、変換されたクエリプランに少なくとも部分的に基づいて入力イベントを処理し

10

20

30

40

50

て出力イベントのセットを生成し、アプリケーションに関連する出力イベントのセットを出力待ち行列に格納し、マイクロバッチ内の各イベントが処理された後に、出力待ち行列内の出力イベントが連続クエリの結果として返されるおよび／または送信されてもよい。

【0102】

本開示の実施形態は、Spark Streamingシステムにおける実行状態を維持し、マイクロバッチ方式ストリーム処理内で完全にステートフルなCQLエンジンを提供し、分散されたCQLエンジンから作成されたスナップショットを管理し、増分スナップショットを処理するための保持アルゴリズムを提供する。開示された技術は、イベントごとのCEP処理をマイクロバッチ方式ベースのストリーム処理に追加した後も高可用性を可能にする。

【0103】

Spark Streamingにおけるステージの出力の非侵襲的モニタリング

近年、データストリーム管理システム(DSM)が開発されてきており、これは潜在的に無限のリアルタイムデータストリーム上で連続的にクエリを実行することができる。新たなDSMの中で、これらのシステムは一般に、単一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。そのようなシステムの例は、Spark(登録商標)プラットフォーム上で動作するSpark Streamingアプリケーションである。

【0104】

DSMSにおける典型的なアプリケーションは、操作または変換のDAG(有向非巡回グラフ)の形態での「トポロジ」として設計される。トポロジはデータ変換パイプラインとして機能する。ほとんどのストリーム処理システム(例えばSpark Streamingシステム)は、アプリケーションのトポロジをマシンのクラスタに迅速に展開する方法を提供し、結果を即座に見ることができる。このような展開の迅速なターンアラウンドサイクルは、アプリケーションに変更を加えるために重要である。ターンアラウンドサイクルが十分速い場合、ユーザは展開の遅延を待たずに結果を確認できる。これは「ストリーム探索」と呼ばれる。

【0105】

ストリーム探索モードでは、カスタマは、通常、既存のトポロジまたはデータ変換パイプラインに新しいコンポーネントを追加することによって、ビジネスアプリケーションを増分的に開発する。このような探索モードでは、変更からの即時出力、およびパイプラインの各ステージからの中間出力も確認することが重要である。

【0106】

Spark(登録商標) StreamingやApache(登録商標) Flinkなどの現在のDSMSでは、トポロジはJava, Scala, Closureなどのプログラミング言語を用いて記述されている。その結果、アプリケーション開発者が1つの変換からの中間出力をモニタリングしたい場合、開発者はプログラムを変更して出力操作を追加する必要がある。これは面倒だけでなく侵襲的でもあり、なぜならば、Spark Streamingのようないくつかのシステムでは、通常、すべての出力操作が追加のジョブになるからである。現在、アプリケーションの実行中に出力モニタリングがアプリケーションに入れられた後にそれをオンにするメカニズムはないため、状況をより複雑にしている。

【0107】

一実施形態では、以下の特徴を有するモニタリング変換プロセスが開示される。(i) 指定された宛先に出力を送信しながらいかなる変換も追加することなく次のパイプラインに出力を生成するパススルー変換、(ii) モニタリング出力がアプリケーションに構成される。(iii) アプリケーションの実行中にモニタリング出力をオフにするか変更することができる。

【0108】

一実施形態では、上記の機能は、次の例を用いて実装することができる。

```
val s1 = cc.cql(inputs, "select * from stream")  
val producerConfig = KafkaMonitorConfig(outputTopic, brokerList)
```

10

20

30

40

50

```
val s1output = s1.monitor(KafkaMonitorOutput(producerConfig));
val s2output = cc.cql(s1output, "select * from s1")
```

上記の例のフローは、以下のように説明され得る。(i) 's1' DStreamに対して'monitor'を呼び出すことによって、MonitorDStreamが、's1'のCQLDStreamの後でDAGに追加される。MonitorDStreamは、KafkaMonitorOutputに関する情報を、構成とともに担持する。(ii) ジョブ生成ステップは、MonitorDStreamからMonitorRDDを作成する。(iii) ジョブが実行されると、MonitorRDD.computeが呼び出される。(iv) タプルを次のパイプラインに返す一方で、pathThroughIteratorが、構成されたモニタ出力に出力を書き込む。

【0109】

モニタリング出力をオフにするかまたは構成を更新するフローは、以下のように実施することができる。(i) RESTサービスがアプリケーションから実行されて、更新を取得する。(ii) アプリケーションおよびステージに対する生成されたMonitorDStreamインスタンスはアプリケーションに保存され、それは、appname (アプリケーション名) および stagename (ステージ名) をキーとして、見つけ出されることができる。(iii) '/monitoroutput/ <appname> / <stagename> /off'に対するPUT操作または新たな構成を伴う'/monitoroutput/ <appname> / <stagename> /configure'に対するPOST操作などのREST要求は'MonitorOutputManager'コンポーネントにデリゲートされる。(iv) MonitorOutputManagerは、MonitorDStreamオブジェクトインスタンスの設定または構成を変更する。(v) ジョブランナーによって実行される次のジョブは、変更の影響を受ける。

【0110】

図8は、本開示の実施形態によって実施される完全にステートフルなクエリ処理をサポートするためにマイクロバッチ方式ストリームの処理を示すフローチャート800を示す。いくつかの実施形態では、フローチャート800に示されているプロセスは、図1、図2および図3のイベント処理システムによって実施することができる。ステップ805で、連続クエリが受信される。いくつかの実施形態では、クエリはパターン認識を含む。たとえば、CQLのMATCH_RECOGNIZE句およびその副次句を呼び出して、CQLクエリでパターン認識を実行できる。ステップ810で、連続クエリに対してクエリプランを生成するために、連続クエリに操作が適用される。クエリプラン (またはクエリ実行プラン) は、クエリまたは連続クエリの処理のために、たとえばSQLリレーショナルデータベース管理システムにおいてデータにアクセスするために使用される順序付けられたステップのセットである。いくつかの実施形態では、操作はDAG変換であり、クエリプランはDAGクエリプランである。DAG変換は頂点と辺との集合であり、頂点はRDDを表し、辺はRDDに適用されるべき操作を表す。ステップ815で、モニタリング変換プロセスを用いて連続クエリがモニタリングされる。たとえば、モニタリング変換プロセスには、次のような特徴がある。(i) 指定された宛先に出力を送信しながらいかなる変換も追加することなく次のパイプラインに出力を生成するパススルー変換、(ii) モニタリング出力がアプリケーションに構成される。(iii) アプリケーションの実行中にモニタリング出力をオフにするか変更することができる。ステップ820で、アプリケーションに関連する入力イベントのマイクロバッチストリームが受信される。いくつかの実施形態では、Spark Streamingは、連続するデータストリームを1秒未満の小さいマイクロバッチつまりマイクロバッチストリームに離散させることができる。ステップ825で、入力イベントは、モニタリング変換プロセスに少なくとも部分的に基づいて処理されて、アプリケーションに関連する出力イベントのセットが生成される。いくつかの実施形態では、処理は連続クエリ処理エンジンを用いて実行され、処理は出力イベントを生成するために入力イベントの各々を増分的に処理することを含む。たとえば、モニタリング変換プロセスが実行されると、マイクロバッチの入力イベントはCQLエンジンに送信される。CQLエンジンは、マイクロバッチの各入力イベントをイベントごとに処理し、モニタリング変換プロセスに少なくとも部分的に基づいて、クエリについてマイクロバッチの各入力イベント

10

20

30

40

50

ント上でインクリメンタルコンピューションを実行し、マイクロバッチにおける各入力イベントに対する出力イベントを作成する。したがって、ステートフル処理はCQLエンジンによって実行される。アプリケーションに関連する出力イベントのセットは、出力待ち行列に格納される。いくつかの実施形態では、マイクロバッチ内の残りのイベントがCQLエンジンによって処理されている間に、出力イベントが出力待ち行列に捕捉される。理解されるように、プロセスは、図4に関して説明されたステップをさらに含み得、例えば、連続クエリ処理エンジンを起動し、マイクロバッチ内の各イベントが処理された後、出力待ち行列内の出力イベントが、連続クエリの結果として返されおよび／または送信されてもよい。

【0111】

本開示の実施形態は、Spark Streamingを用いた非侵入的出力モニタリング技術、および実行中のSpark Streamingアプリケーションからの中間出力をオン／オフにする技術を提供する。さらに、開示された技術は、ストリーム探索のために中間出力モニタリングを追加し、出力を変更することを可能にする。

【0112】

例示的システム

図9～図7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9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900の簡略図を示す。図示の実施形態では、分散型システム900は、1つ以上のネットワーク910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902，904，906および908を含む。サーバ912は、ネットワーク910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902，904，906および908と通信可能に結合されてもよい。

【0113】

さまざまな実施形態では、サーバ912は、アイデンティティ管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912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902，904，906および／または908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902，904，906および／または908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912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。

【0114】

図9に示される構成では、システム900のソフトウェアコンポーネント918，920および922は、サーバ912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900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902，904，906および／または908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900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9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。

【0115】

クライアントコンピューティングデバイス 902, 904, 906、および／または 908 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、BlackBerry 10、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 910 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。

【0116】

図 9 の分散型システム 900 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 912 と対話してもよい。

【0117】

分散型システム 900 におけるネットワーク 910 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 910 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）1002.11 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。

【0118】

サーバ 912 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 912 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

10

20

30

40

50

レンジデバイスの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912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912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912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。

【0119】

サーバ912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912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。

【0120】

いくつかの実現例では、サーバ912は、クライアントコンピューティングデバイス902、904、906および908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912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902、904、906および908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。

【0121】

分散型システム900は、1つ以上のデータベース914および916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914および916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914および916のうちの1つ以上は、サーバ912に局在する（および／またはサーバ912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914および916は、サーバ912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912と通信してもよい。一組の実施形態では、データベース914および916は、記憶域ネットワーク（SAN）にあってもよい。同様に、サーバ912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912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914および916は、SQLフォーマットされたコマンドに回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。

【0122】

図10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1000を示す。いくつかの実施形態では、コンピュータシステム1000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10に示すように、コンピュータシステム1000は、バスサブシステム100

10

20

30

40

50

2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1004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1006、I/Oサブシステム1008、ストレージサブシステム1018および通信サブシステム1024を含んでもよい。ストレージサブシステム1018は、有形のコンピュータ可読記憶媒体1022およびシステムメモリ11010を含む。

【0123】

バスサブシステム1002は、コンピュータシステム1000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1002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实施例は、複数のバスを利用してもよい。バスサブシステム1002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、エンハンスド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IEEE P1386.1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。

【0124】

処理サブシステム1004は、コンピュータシステム1000の動作を制御し、1つ以上の処理ユニット1032、1034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1004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1004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。

【0125】

いくつかの実施形態では、処理サブシステム1004内の処理ユニットは、システムメモリ1010またはコンピュータ可読記憶媒体1022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1010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1010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1004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。

【0126】

特定の実施形態では、コンピュータシステム1000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1004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1006を設けることができる。

【0127】

I/Oサブシステム1008は、コンピュータシステム1000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1000から、もしくはコンピュータシステム1000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

10

20

30

40

50

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1000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、ダイヤル、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。ユーザインターフェイス入力デバイスは、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／またはジェスチャー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。

【0128】

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、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。

【0129】

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1000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。

【0130】

ストレージサブシステム1018は、コンピュータシステム1000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1018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1004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1018に格納されてもよい。ソフトウェアは、処理サブシステム1004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1018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。

【0131】

ストレージサブシステム1018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10に示すように、ストレージサブシステム1018は、システムメモリ1010およびコンピュータ可読記憶媒体1022を含む。システムメモリ1010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1000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1004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1010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。

【0132】

一例として、限定を伴うことなく、図10に示されるように、システムメモリ1010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1012、プログラムデータ1014およびオペレーティングシステム1016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1016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) 10 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。

【0133】

コンピュータ可読記憶媒体1022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1004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1018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1022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1022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1022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1022は、コンピュータシステム1000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。

【0134】

ある実施形態では、ストレージサブシステム1000は、コンピュータ可読記憶媒体1022にさらに接続可能なコンピュータ可読記憶媒体リーダ1020も含んでもよい。システムメモリ1010とともに、およびオプションとしてシステムメモリ1010との組み合わせで、コンピュータ可読記憶媒体1022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。

【0135】

特定の実施形態では、コンピュータシステム1000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1000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1000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1000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。

【0136】

通信サブシステム1024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1024は、他のシステムとコンピュータシステム1000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1024は、コンピュータシステム1000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1024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お知らせを伝達することができる。

【0137】

通信サブシステム1024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1024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（IEEE 802.11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1024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。

【0138】

通信サブシステム1024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1024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1026、イベントストリーム1028、イベント更新1030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1024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1026を受信（または送信）するように構成されてもよい。

【0139】

ある実施形態では、通信サブシステム1024は、連続データストリームの形式でデータ

10

20

30

40

50

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1028および／またはイベント更新情報1030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。

【0140】

また、通信サブシステム1024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1026、イベントストリーム1028、イベント更新情報1030などを、コンピュータシステム1000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。

【0141】

コンピュータシステム1000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。

【0142】

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10に示されるコンピュータシステム1000の記載は、単に具体的な例として意図される。図10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。

【0143】

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。

【0144】

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

10

20

30

40

50

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。

【0145】

図11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1100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1100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1102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1104、1106および1108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1102と対話して、クラウドインフラストラクチャシステム1102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。

10

【0146】

図に示されるクラウドインフラストラクチャシステム1102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1102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。

20

【0147】

クライアントコンピューティングデバイス1104、1106および1108は、902、904、906および908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。

【0148】

例示的なシステム環境1100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1102と対話してもよい。

30

【0149】

ネットワーク1110はクライアント1104、1106および1108とクラウドインフラストラクチャシステム1102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910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。

【0150】

クラウドインフラストラクチャシステム1102は、サーバ912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。

40

【0151】

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ル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

50

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。

【0152】

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れない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。

【0153】

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1102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。

【0154】

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1102は、クラウドインフラストラクチャシステム1102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1102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1102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1102およびクラウドインフラストラクチャシステム1102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。

【0155】

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1102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 1102 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 1102 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。

【0156】

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 1102 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。

【0157】

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。

【0158】

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。

【0159】

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。

【0160】

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1102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利用される資源を提供するためのインフラストラクチャ資源1130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1130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。

10

【0161】

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1102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1130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の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。

20

【0162】

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1102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1102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1132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。

30

【0163】

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1102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1102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。

【0164】

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1120、オーダーオーケストレーションモジュール1122、オーダープロビジョニングモジュール1124、オーダー管理およびモニタリングモジュール1126、ならびにアイデンティティ管理モジュール1128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、サーバクラスター、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。

40

【0165】

例示的な動作1134においては、クライアントデバイス1104、1106または1108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

50

システム 1102 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 1102 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 1102 と対話してもよい。特定の実施形態において、顧客は、クラウドユーザーインターフェイス (UI)、クラウド UI 1112、クラウド UI 1114 および／またはクラウド UI 1116 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 1102 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 1102 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。

10

【0166】

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 UI 1112、1114 および／または 1116 を介して受取られてもよい。

【0167】

動作 1136 において、オーダーはオーダーデータベース 1118 に格納される。オーダーデータベース 1118 は、クラウドインフラストラクチャシステム 1118 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。

【0168】

動作 1138 で、オーダー情報はオーダー管理モジュール 1120 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 1120 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。

20

【0169】

動作 1140 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 1122 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 1122 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 1122 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 1124 のサービスを用いてサポートしてもよい。

30

【0170】

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 1122 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 1142 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 1122 は、オーダープロビジョニングモジュール 1124 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 1124 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 1124 は、クラウドインフラストラクチャシステム 1100 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利用さ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 1122 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。

40

【0171】

動作 1144 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 1124 によってクライアントデバイス 1104、1106 および／または 11

50

08における顧客に送信されてもよい。動作1146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1126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1126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。

【0172】

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1100はアイデンティティ管理モジュール1128を含んでもよい。アイデンティティ管理モジュール1128は、クラウドインフラストラクチャシステム1100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1128は、クラウドインフラストラクチャシステム1102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1128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。

【0173】

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。

【0174】

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。

【0175】

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同等物は、特許請求の範囲内にある。

10

20

30

40

50

【図 5】

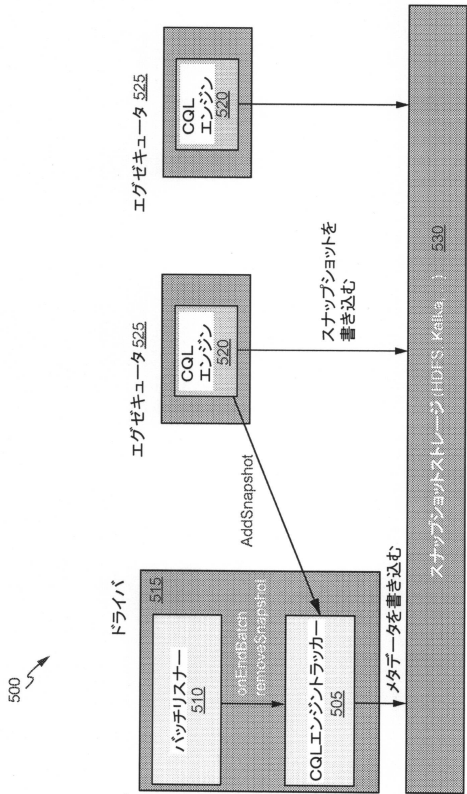


FIG. 5

【図 6 A】

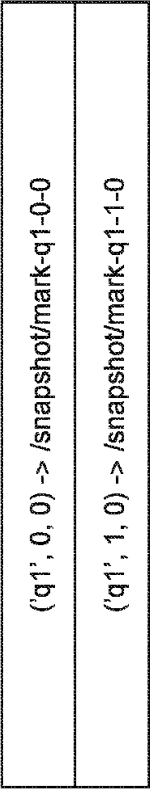


FIG. 6A

【図 6 B】

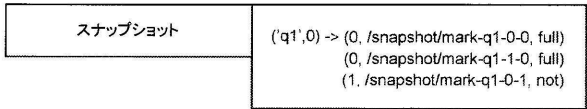


FIG. 6B

【図 7】

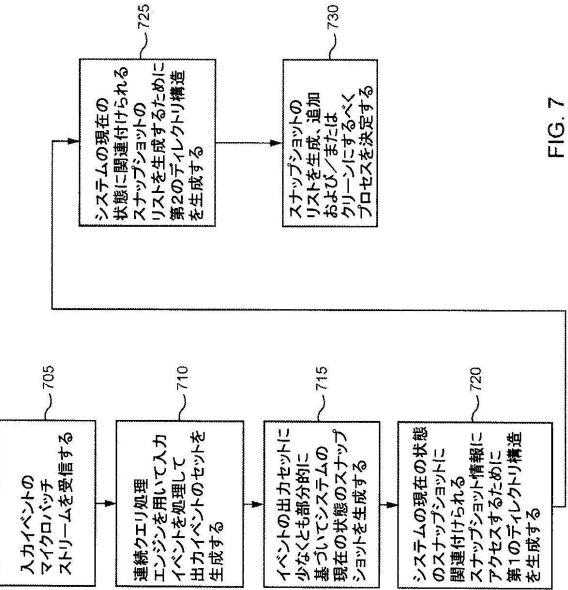
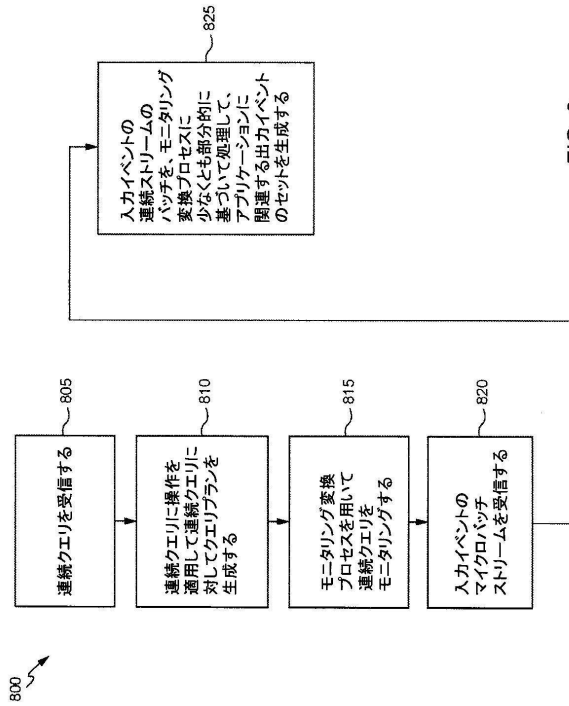


FIG. 7

【図 8】



【図 9】

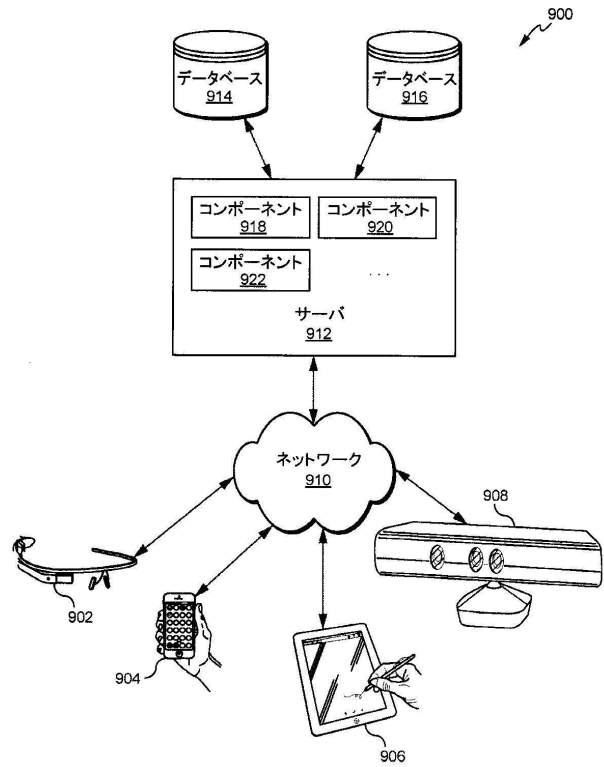
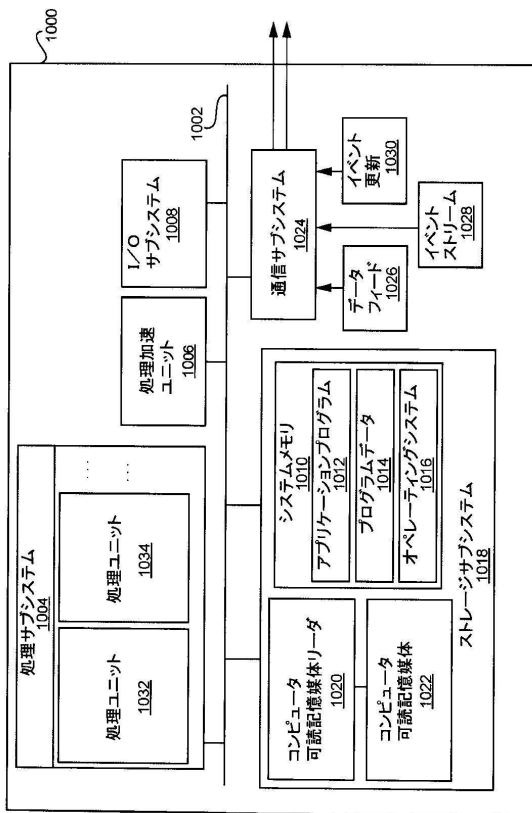


FIG. 9

【図 10】



【図 11】

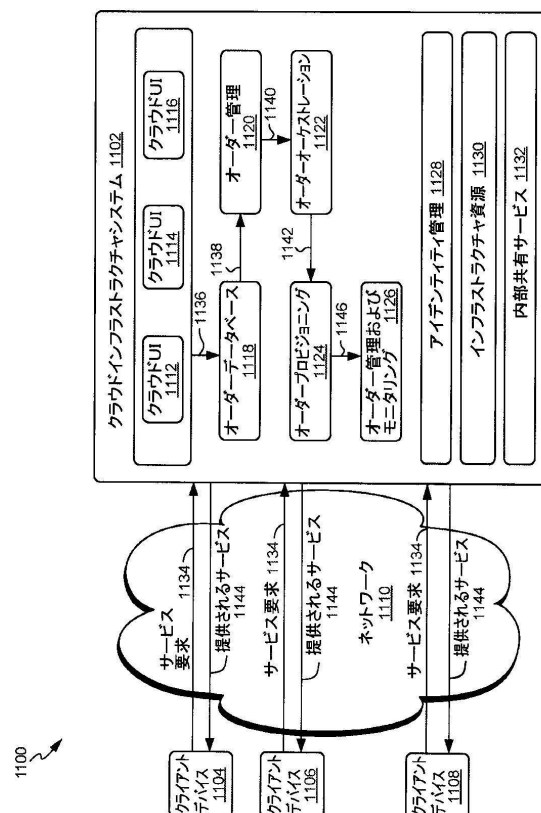


FIG. 11

フロントページの続き

- ドライブ、133、アパートメント・ケイ
(72)発明者 ツッカラム, プラブ
アメリカ合衆国、94582 カリフォルニア州、サン・ラモン、ハーロウ・レーン、2525
(72)発明者 クマール, サントシュ
インド、560090 バンガロール、イエスバントプル・ホブリ、サマシェッティハリ、ジャナ
ブリヤ・グリーンウッド・アパートメント、フラット・ナンバー・ビィ・4-326
(72)発明者 アドバニ, パバン
インド、560037 バンガロール、マラサハリ、バーサー・メイン・ロード、ローハン・バサ
ンサ・アパートメンツ、ディー1006
(72)発明者 ムレー, クナル
インド、452018 マディヤ・プラデーシュ、インドール、ティラック・ナガー・エクステン
ション、181
(72)発明者 トイリオン, ジェフリー
アメリカ合衆国、94019 カリフォルニア州、ハーフ・ムーン・ベイ、パインハースト・レー
ン、28
審査官 甲斐 哲雄
(56)参考文献 米国特許出願公開第2014/0095444 (US, A1)
特開2006-338432 (JP, A)
特開2013-058221 (JP, A)
(58)調査した分野 (Int.Cl., DB名)
G06F 16/00-16/958