(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7169973号
(P7169973)

(45)発行日　令和4年11月11日(2022.11.11)

(24)登録日　令和4年11月2日(2022.11.2)

(51)国際特許分類

		FI		
G06F	*8/65 (2018.01)*	G06F	8/65	
G06F	*9/455(2006.01)*	G06F	9/455	150

請求項の数　9　（全39頁）

(21)出願番号　特願2019-531267(P2019-531267)
(86)(22)出願日　平成30年9月21日(2018.9.21)
(65)公表番号　特表2020-534587(P2020-534587A)
(43)公表日　令和2年11月26日(2020.11.26)
(86)国際出願番号　PCT/US2018/052131
(87)国際公開番号　WO2019/060663
(87)国際公開日　平成31年3月28日(2019.3.28)
審査請求日　令和3年5月26日(2021.5.26)
(31)優先権主張番号　62/561,599
(32)優先日　平成29年9月21日(2017.9.21)
(33)優先権主張国・地域又は機関
米国(US)

(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　クリシュナッパ，ナゲンドラ
インド、５６００２９ バンガロール、エヌ・エス・パリャ、ナンバー・１２／１・アンド・２、レベル・４
(72)発明者　ナラヤナン，ビジュ
インド、６９５０１７ ケーララ、トリバンドラム、スリーカリアム、サンティ・ナガー、プロット・５４

最終頁に続く

(54)【発明の名称】　多層クラウドベースアプリケーションスタックをアップデートするためのシステムおよび方法

(57)【特許請求の範囲】

【請求項１】

コンピュータによって実施される方法であって、

アプリケーション環境が依存する１つ以上のコンポーネントに対する１つ以上の依存関係を含む前記アプリケーション環境のインスタンスを実行することと、

前記１つ以上のコンポーネントのうちの少なくとも１つのコンポーネントを修正するように構成される１つ以上のアップデートを検出することと、

前記１つ以上のアップデートを検出することに応答して、前記１つ以上のアップデートを抽出するためにアップデートサーバにアクセスすることと、

前記アプリケーション環境の新しいインスタンスを自動的に生成することと、

前記アプリケーション環境の前記新しいインスタンスに前記１つ以上のアップデートをインストールすることと、

前記インスタンスに関連付けられるメタデータを識別することと、

前記新しいインスタンスに関連付けられる新しいメタデータを識別することと、

前記インスタンスに関連付けられる前記メタデータを前記新しいインスタンスに関連付けられる前記新しいメタデータと比較することと、

前記インスタンスに関連付けられる前記メタデータと前記新しいインスタンスに関連付けられる前記新しいメタデータとの間の１つ以上の差を前記比較に基づいて決定することと、

識別された前記１つ以上の差の各々を含むアップデートデータを生成することと、

識別された前記１つ以上の差を前記インスタンスに組み込むことにより、前記アップデートデータを使用して前記インスタンスをアップデートすることとを含む、方法。
【請求項２】
前記インスタンスをアップデートすることはさらに、
前記アプリケーション環境において実行されるアプリケーションに関連付けられるすべてのトランザクションが停止されるロック期間を開始することと、
前記ロック期間中に前記アップデートデータにより前記インスタンスをアップデートすることと、
前記ロック期間を解除することとを含み、前記ロック期間の解除は、前記アプリケーションに関連付けられるトランザクションが継続することを可能にする、請求項１に記載の方法。
【請求項３】
前記メタデータは、前記インスタンスに関連付けられるバグ番号の第１の集合を含んでおり、前記新しいメタデータは、前記新しいインスタンスに関連付けられるバグ番号の第２の集合を含んでおり、バグ番号の前記第１の集合がバグ番号の前記第２の集合と異なる場合、前記第１の集合と前記第２の集合との間の差は、前記１つ以上のアップデートに含まれる１つ以上のバグフィックスに対応する、請求項１または２に記載の方法。
【請求項４】
前記アップデートデータは、前記１つ以上のアップデートに含まれる前記１つ以上のバグフィックスに関連付けられるオブジェクトを含む、請求項３に記載の方法。
【請求項５】
中央サーバ上に格納される前記１つ以上のアップデートを検出することは、規則的または不規則的な間隔で、アップデートについて前記中央サーバをポーリングすることを含む、請求項１～４のいずれか１項に記載の方法。
【請求項６】
前記メタデータは、前記１つ以上のアップデートを有していないソースコードに関連付けられており、前記新しいメタデータは、前記１つ以上のアップデートを有する前記ソースコードに関連付けられる、請求項１～５のいずれか１項に記載の方法。
【請求項７】
アップデートされた前記インスタンスをリブートすることをさらに含む、請求項１～６のいずれか１項に記載の方法。
【請求項８】
システムであって、
１つ以上のデータプロセッサと、
命令を含む一時的でないコンピュータ可読記憶媒体とを含み、前記命令は、前記１つ以上のデータプロセッサ上で実行されると、前記１つ以上のデータプロセッサに、
アプリケーション環境が依存する１つ以上のコンポーネントに対する１つ以上の依存関係を含む前記アプリケーション環境のインスタンスを実行することと、
前記１つ以上のコンポーネントのうちの少なくとも１つのコンポーネントを修正するように構成される１つ以上のアップデートを検出することと、
前記１つ以上のアップデートを検出することに応答して、前記１つ以上のアップデートを抽出するためにアップデートサーバにアクセスすることと、
前記アプリケーション環境の新しいインスタンスを自動的に生成することと、
前記アプリケーション環境の前記新しいインスタンスに前記１つ以上のアップデートをインストールすることと、
前記インスタンスに関連付けられるメタデータを識別することと、
前記新しいインスタンスに関連付けられる新しいメタデータを識別することと、
前記インスタンスに関連付けられる前記メタデータを前記新しいインスタンスに関連付けられる前記新しいメタデータと比較することと、
前記インスタンスに関連付けられる前記メタデータと前記新しいインスタンスに関連付

けられる前記新しいメタデータとの間の１つ以上の差を前記比較に基づいて決定することと、
識別された前記１つ以上の差の各々を含むアップデートデータを生成することと、
識別された前記１つ以上の差を前記インスタンスに組み込むことにより、前記アップデートデータを使用して前記インスタンスをアップデートすることとを含む動作を実行させる、システム。
【請求項９】
請求項１～７のいずれか１項に記載の方法を１つ以上のプロセッサに実行させるためのコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願
この出願は、米国特許法第１１９条（ｅ）に従って、２０１７年９月２１日に出願され、「多層クラウドベースアプリケーションスタックをアップデートするためのシステムおよび方法（SYSTEMS AND METHODS FOR UPDATING MULTI-TIER CLOUD-BASED APPLICATION STACKS）」という名称を有する米国仮出願連続番号第６２／５６１，５９９号の利益および優先権を主張する。当該米国仮出願の内容は全文、本願明細書において参照により援用される。
【０００２】
技術分野
本開示は、クラウドネットワーク環境における多層アプリケーションスタックを自動的および周期的にアップデートすることに関する。より特定的には、本開示は、アプリケーションへの新しいアップデートがリリースされたか否かを決定するよう、周期的に（たとえば規則的または不規則的な間隔で）中央サーバにアクセスするシステムおよび方法に関する。アプリケーションへの新しいアップデートがリリースされた場合、当該システムおよび方法は、最小のネットワーク負荷および最小のサービス中断で、当該アプリケーションに関連付けられるアプリケーション環境に当該新しいアップデートを自動的に適用する。
【背景技術】
【０００３】
背景
一般に、アプリケーションスタックは、ウェブ層（たとえばウェブサーバ）、アプリケーション層（たとえばアプリケーションサーバ）、およびデータベース層（たとえばデータベースサーバ）を含む複数の層を使用して実現され得る。これらの層の各々上では、ソフトウェアコンポーネントが独立して実行され得る。さまざまな層のソフトウェアコンポーネントは、同じサーバ内で、または、複数のサーバに亘って、通信するように構成され得る。さらに、これらのサーバは、同じデータセンタ内で稼働するバーチャルマシンもしくは物理マシンであり得るか、または、異なる領域におけるデータセンタに亘って稼働するバーチャルマシンもしくは物理マシンであり得る。時間の経過とともに、アプリケーションスタックは、たとえばバグをフィックスするために、または、アプリケーションスタックを使用して提供されるサービスに機能を加えるために、アップデートされる必要があり得る。アプリケーションスタックをアップデートすることは典型的に、ソフトウェアコンポーネントをインストールすることから発生する問題についてテストするよう、各層の異なるソフトウェアコンポーネントをアップデートおよびテストするためにベンダーを雇うことを伴う。しかしながら、このアップデートプロセスはネットワークリソースに対して負荷のあるものである。なぜならば、複雑なアプリケーションスタックは非常に大きくあり得（たとえば１テラバイト）、また、サーバのテストを可能にするためにはサーバは典型的に長期間オフラインにされる必要があるからである。
【０００４】
従来のアプリケーションスタックを有さないアプリケーション環境においてさえ、アプ

リケーションは、クラウドネットワーク環境におけるさまざまなリソースまたはコンポーネントに対する依存関係（dependencies）を有し得る。アプリケーションのイメージを上書きすることによりアプリケーションをアップデートすることは、これらの依存関係を破壊し得、さらに別のバグおよび／または障害を引き起こし得る。アプリケーションイメージが上書きされた後、アプリケーション環境のマネージャは、それらの依存関係のすべてを再接続または再構築することによって、アップデートプロセスについてフォローアップする必要がある。そのようなフォローアッププロセスは、一般に時間がかかるが、これらの依存関係の数が多い大きなアプリケーション環境では、さらに時間がかかることになる。

【発明の概要】

【課題を解決するための手段】

【０００５】

概要

本開示のある局面および特徴は、処理リソースに対する負荷が最小であり、サーバダウンタイムが限定された、クラウド環境において自動的に多層アプリケーションスタックをアップデートするためのシステムおよび方法に関する。特に、ある実施形態は、クラウド環境（たとえばウェブコードをホストするウェブサーバ、アプリケーションコードをホストするアプリケーションサーバ、および、データベースサービスを提供するデータベースサーバを含む、アプリケーションの３層モデル）においてアプリケーションスタック上で実行されるアプリケーションに関する。アプリケーションは、アプリケーションスタックへの新しいアップデートについてチェックするために、周期的に中央サーバ（たとえばアップデートサーバおよび中央サーバなど）と通信するように構成され得る。アプリケーションスタックへのアップデートは、（たとえば規則的または不規則的な間隔で）周期的に中央サーバに格納され得る。アップデートの非限定的な例は、アプリケーションバグをフィックスし、および／または、アプリケーションの特徴を追加もしくは修正するように構成されるアップデートイメージおよびパッチセットを含み得る。アップデートイメージ（たとえば、四半期（quarter）に一度発行されるメジャーリリース）は、アプリケーションの特定のバージョンのすべてのバグフィックスおよび新しい特徴を含んでいるデータパッケージであり得る。パッチセット（たとえば、メジャーリリース同士の間に発行されるマイナーバグフィックス）は、最後のアップデートイメージがリリースされてからレポートされた個々のバグフィックスを含むデータパッケージであり得る。いくつかの実施形態では、アップデートは、アプリケーションのアプリケーションコードに含まれるオブジェクト（たとえばバグフィックス）へのバイナリ変更の形態であり得る。新しいアップデートが中央サーバからのダウンロードにより利用可能な場合、アプリケーションは、アプリケーションスタックに関連付けられるネットワーク位置へ新しいアップデートをダウンロードし得る（たとえば、アプリケーションスタックに関連付けられるリポジトリへ新しいアップデートをダウンロードする）。

【０００６】

いくつかの実施形態では、アプリケーションは、プロダクション環境において現在実行されているインスタンスに関連付けられる。現在実行されているインスタンスは、中央サーバに格納されている新しいアップデートを含んでいないアプリケーションのバージョン（たとえば、まだアップデートされていないアプリケーションの古いバージョン、および／または、新しいアップデートがまだインストールされていないアプリケーションの最も最近のバージョン）に対応し得る。アプリケーションは、バーチャルマシンを生成し、バーチャルマシン上でアプリケーションスタックの新しいインスタンスを実行するように構成され得る。さらに、アプリケーションは、ダウンロードされた新しいアップデートをアプリケーションの新しいインスタンスにインストールし得る。たとえば、インストールされた新しいアップデートを有する新しいインスタンスは、新しいアップデートに含まれる任意のバグフィックスまたは新しい特徴を組み込む。さらに、アプリケーションは、現在実行されているインスタンスを新しいインスタンスと比較して差が存在するか否かを識別

し得る。たとえば、現在実行されているインスタンスと新しいインスタンスとの間の差は、新しいインスタンスに含まれる新しいアップデート（たとえば、アプリケーションコードに新しく追加されたオブジェクト、アプリケーションに追加された新しい特徴、アプリケーションのバイナリコードにおけるフィックスされたバグなど）を表わし得る。比較の間に検出される差は、アップデートデータパッケージへとシリアライズされ得、インストールのために、現在実行されているインスタンスにエクスポートされ得る。アップデートデータパッケージは、現在実行されているインスタンスにインストールされ得、その結果として、現在のインスタンスは、新しいバグフィックスおよび／または新しい特徴によりアップデートされる。たとえば、現在実行されているインスタンスにアップデートデータパッケージをインストールすることは、検出された差に関連付けられるオブジェクト（たとえば新しいアップデートによって修正、削除または追加されるファイルリファレンスオブジェクト）を識別することと、現在実行されているインスタンスのアプリケーションサーバに格納されるアプリケーションコードに、修正、削除または追加されたオブジェクトを組み込むこととを含み得る。新しいアップデートは、複数の層のうちのいずれかの上で実行されるソフトウェアコンポーネントをアップデートし得る。例示すると、非限定的な例として、新しいアップデートは、データベースサーバのソフトウェアコンポーネントをアップデートすることなく、ウェブサーバのソフトウェアコンポーネント中のバグをフィックスし得る。いくつかの実施形態では、アプリケーション環境は、アプリケーション環境が依存する１つ以上のコンポーネント（たとえばソフトウェアコンポーネント）に対する依存関係を含み得る（たとえば、これらの依存関係のため、アプリケーション環境の実行可能イメージは異なる）。

【０００７】

いくつかの実施形態では、現在実行されているインスタンスと新しいインスタンスとの比較は、現在実行されているインスタンスに関連付けられるメタデータを、（新しいアップデートを含む）新しいインスタンスに関連付けられるメタデータと比較することにより実現され得る。いくつかの実施形態では、現在実行されているインスタンスおよび新しいインスタンスのうちのいずれかまたは両方に関連付けられるメタデータは、ソフトウェア特徴フラグ、コードに含まれるファイルリファレンスオブジェクトの識別子、インストールもしくは実行セッティングまたは他のメタデータパラメータを含み得る。これらの実施形態では、現在実行されているインスタンスと新しいインスタンスとの間でメタデータを比較することは、古いインスタンスにおいて有効化／無効化された特徴フラグを新しいインスタンスにおいて有効化／無効化された特徴フラグと比較することと、古いインスタンスにおけるインストールもしくは実行セッティングまたはパラメータ値を新しいインスタンスにおけるインストールもしくは実行セッティングまたはパラメータ値と比較することと、どのファイルリファレンスオブジェクトが修正、削除、または、コードに追加されたかを識別するために、現在実行されているインスタンスのファイルリファレンスオブジェクトの識別子を新しいインスタンスのファイルリファレンスオブジェクトの識別子と比較することとを含み得る。いくつかの実施形態では、現在実行されているインスタンスに関連付けられるメタデータは、１つ以上のバグ番号を含み得る。バグ番号は、現在のインスタンスについてフィックスまたは修正されたバグを記述するバグレポートを一意に識別し得る。同様に、新しいアップデートを含む新しいインスタンスに関連付けられるメタデータはさらに、新しいインスタンスについてフィックスされたバグを表わす１つ以上のバグ番号を含み得る。現在実行されているインスタンスに関連付けられるバグ番号の集合が新しいインスタンスに関連付けられるバグ番号の集合と異なる場合、アプリケーションは、アップデートが新しいインスタンスに対して実行されたと決定し得る。アプリケーションは、現在実行されているインスタンスではなく新しいインスタンスについてフィックスされたバグに関するバグレポート（たとえばバグ番号間のデルタ）についてクエリ送信し抽出し得る。各バグ番号および／またはバグレポートは、バグフィックスを促進、実現または実行した１つ以上のオブジェクト（たとえばファイルリファレンスオブジェクト）に関連付けられる。現在実行されているインスタンスについてのバグ番号の集合と、新しいイ

ンスタンスについてのバグ番号の集合との間で異なるバグ番号が識別およびエクスポートされ得、識別されたバグ番号は、現在実行されているインスタンスに対して実施または実行され得、これにより、新しいインスタンスにおけるアップデートを含めるよう、現在実行されているインスタンスがアップデートされる。

【０００８】

いくつかの実施形態では、新しいアップデートは、データベースサーバに関係する１つ以上のオブジェクトへの変更を有するソースコードを含み得る。１つ以上のオブジェクトへの変更は、変更されたオブジェクトを表わすファイルリファレンスオブジェクトとして、新しいアップデートに関連付けられるメタデータに追加され得、ソースディレクトリに格納され得る。たとえば、ファイルリファレンスオブジェクトは、将来の参照および抽出のためにメタデータテーブルに格納される識別子を有し得る。別の例として、コード、ファイルまたはスクリプトへのバイナリ変更（たとえばＪａｖａ（登録商標）アーカイブ（ＪＡＲ：Java Archive）ファイルのようなｊａｖａファイルにおける変更）が、ファイルリファレンスオブジェクトとしてコード、ファイルまたはスクリプトのメタデータに追加され得る。ファイルリファレンスオブジェクトは、ファイルがデプロイされるネットワーク位置へ方向付けするパスを表わす。これらの実施形態では、現在実行されているインスタンスが新しいインスタンスと比較されると、現在実行されているインスタンスのファイルリファレンスオブジェクトと新しいインスタンスのファイルリファレンスオブジェクトとの間の差が、格納され、シリアライズされ、現在実行されているインスタンスに適用され得る。

【０００９】

有利なことに、本発明のある実施形態は、アプリケーションがプロダクション環境において動作している間、多層アプリケーションスタックを自動的にアップデートし得る。さらに、アプリケーションアップデート中におけるネットワークリソースにかかる負荷および／またはネットワークリソースの中断の技術的問題は、上記および本願明細書において記載される新規性の技術的なポイントによって解決される。たとえば、アプリケーションの新しいバーチャルインスタンスのメタデータをアプリケーションのオリジナルまたは現在実行されているインスタンス（すなわちプロダクション環境で実行されているアプリケーション）のメタデータと比較することと、比較に基づいて検出される差を表わすメタデータをインストールすることとによって、クラウドネットワーク環境における処理リソースの使用が改善され、アプリケーションスタックのサービス中断が低減される。さらに、アップデートのサイズ、または、現在実行されているインスタンスのバージョンと新しいインスタンスのバージョンとの間で発行されたアプリケーションの後継またはバージョンの数に関わらず、多層アプリケーションスタックはアップデートされ得る。

【００１０】

明細書は、以下の添付の図面を参照する。当該図面において、異なる図における同様の参照番号の使用は、同様または類似のコンポーネントを示すように意図される。

【図面の簡単な説明】

【００１１】

【図１Ａ】本開示のいくつかの実施形態に従った、ログデータを構成、収集および分析するための例示的なシステムを示す図である。

【図１Ｂ】ログデータを構成、収集および分析するためにシステムを使用するアプローチのフローチャートを示す図である。

【図２】レポートユーザインターフェイスの例を示す図である。

【図３Ａ】ホスト環境でのログ分析システムの内部構造を示すフロー図である。

【図３Ｂ】ホスト環境でのログ分析システムの内部構造を示すフロー図である。

【図３Ｃ】ホスト環境でのログ分析システムの内部構造を示すフロー図である。

【図４】例示的なネットワーク環境を示すブロック図である。

【図５】多層アプリケーションスタックを自動的にアップデートするための処理フローを示すブロック図である。

【図６】ある実施形態に従った例示的なインターフェイスの図である。
【図７】ある実施形態に従った例示的なインターフェイスの図である。
【図８】ある実施形態に従った例示的なインターフェイスの図である。
【図９】テンプレートを作成するための処理を示すフローチャートである。
【図１０】実施形態のうちの１つを実現するための分散型システムを示す簡略図である。
【図１１】システム環境の１つ以上のコンポーネントを示す簡略ブロック図である。
【図１２】本発明のさまざまな実施形態が実現され得る例示的なコンピュータシステムを示す図である。
【発明を実施するための形態】
【００１２】
詳細な説明
以下の記載において、本発明の実施形態が完全に理解されるように説明目的のために、多くの特定の詳細を記載する。しかしながら、これらの特定の詳細がなくてもさまざまな実施形態が実施され得るということは明らかであろう。図面および記載は、限定的であるように意図されない。
【００１３】
多くのタイプのコンピューティングシステムおよびアプリケーションは、当該コンピューティングシステムまたはアプリケーションの動作に関する莫大な量のデータを生成するか、または、当該動作の結果として得られる莫大な量のデータを生成する。その後、これらの莫大な量のデータは、ログファイル／レコードといったような収集位置に頻繁に格納され、システムまたはアプリケーションの挙動または動作を分析する必要がある場合に、その後の期間において検討され得る。
【００１４】
以下の説明において、例示として、「ログ」データに関して実施形態が記載され得るが、他のタイプのデータの処理もさらに考えられる。したがって、実施形態は、その適用において、ログデータのみに限定されるべきでない。さらに、以下の記載はさらに、処理されるデータを「レコード」または「メッセージ」として交換可能に言及する場合があるが、本発明の範囲を当該データについての任意の特定のフォーマットに限定する意図はない。
【００１５】
図１Ａは、本発明のいくつかの実施形態に従った、ログデータを構成、収集、および分析するための例示的なシステム１００を示す。システム１００は、いくつかの実施形態において、クラウドベースおよび／またはＳａａＳ（software as a service（ソフトウェア・アズ・ア・サービス））ベースのアーキテクチャとして具現化されるログ分析システム１０１を含む。これは、ログ分析システム１０１は、サービスを必要とする各顧客が、顧客自身のネットワークにサービスコンポーネントを個々にインストールおよび構成する必要がないように、ホストされたプラットホーム上のサービスとしてログ分析機能を提供することが可能であるということを意味する。ログ分析システム１０１は、複数の別個の顧客にログ分析サービスを提供することができ、如何なる数の顧客にもサービスを提供するようスケールされ得る。
【００１６】
各顧客ネットワーク１０４は、任意数のホスト１０９を含み得る。ホスト１０９は、１つ以上のログファイルとしてログデータを生成する、顧客ネットワーク１０４内のコンピューティングプラットフォームである。ホスト１０９内で作り出される生ログデータは、任意のログ生成源から生じ得る。たとえば、生ログデータは、データベース管理システム（ＤＢＭＳ： database management system）、データベースアプリケーション（DB App）、ミドルウェア、オペレーティングシステム、ハードウェアコンポーネント、または、任意の他のログ生成アプリケーション、コンポーネントもしくはシステムから生じ得る。ログ分析システム１０１と通信するために、１つ以上のゲートウェイ１０８が各顧客ネットワークにおいて提供される。
【００１７】

システム１００は、ログ分析システム１０１を動作させかつログ分析システム１０１とインタラクションするためにシステム１００を使用する１つ以上のユーザステーション１０３における１人以上のユーザを含み得る。ユーザステーション１０３は、システム１００においてログ分析システム１０１を動作するかまたはログ分析システム１０１とインターフェイス接続するために使用され得る任意のタイプのコンピューティングステーションを含む。そのようなユーザステーションの例はたとえば、ワークステーション、パーソナルコンピュータ、モバイルデバイスまたはリモートコンピューティングターミナルを含んでいる。ユーザステーションは、ディスプレイモニタのような、ユーザステーションにてユーザにユーザインターフェイスを表示するためのディスプレイデバイスを含む。ユーザステーションはさらに、システム１００のアクティビティに対して動作制御を提供するよう、ユーザのための１つ以上の入力デバイスを含む。入力デバイスとしては、ユーザ入力を生成するようグラフィカルユーザインターフェイスにおいてポインティングオブジェクトを操作するマウスまたはキーボードがある。いくつかの実施形態では、ユーザステーション１０３は、顧客ネットワーク１０４内に位置し得る（が顧客ネットワーク１０４内に位置する必要はない）。

【００１８】

ログ分析システム１０１は、ユーザステーション１０１におけるユーザにアクセス可能な機能を含んでおり、ログ分析システム１０１は、ログデータの構成、収集および分析を実行するよう、エンジン、メカニズムおよび／またはモジュール（ハードウェア、ソフトウェアまたはハードウェアおよびソフトウェアの混合に関わらず）の集合として実現される。ユーザインターフェイス（ＵＩ）メカニズムは、分類および分析結果を表示し、ユーザがログ分析システムとインタラクションすることを可能にするようＵＩを生成する。

【００１９】

図１Ｂは、ログデータを構成、収集および分析するためにシステム１００を使用するアプローチのフローチャートを示す。この図１Ｂの議論は、図１Ａにおけるシステム１００のために示されるコンポーネントに言及する。

【００２０】

１２０において、システム内でログモニタリングが構成される。これはたとえば、ユーザ／顧客が望むログモニタリング／データ収集のタイプを構成するようユーザ／顧客によって行われ得る。システム１０１内において、ＵＩ制御を含む構成メカニズム１２９が、ログ収集構成１１１およびログ収集構成についてのターゲット表現１１３を選択および構成するよう、ユーザによって動作可能である。

【００２１】

ログ収集構成１１１は、何のデータを収集するべきか（たとえばどのログファイル）、収集するべきデータの位置（たとえばディレクトリ位置）、データにどのようにアクセスするか（たとえば取得するべきログ内のログおよび／または特定フィールドのフォーマット）、および／または、データを収集するべきとき（たとえば周期ベース）を識別する情報の集合（たとえばログルール、ログソース情報およびログタイプ情報）を含む。ログ収集構成１１１は、サービスプロバイダによって含まれるアウト・オブ・ザ・ボックスルール（out-of-the-box rules）を含み得る。ログ収集構成１１１はさらに、顧客が規定したルール／顧客がカスタマイズしたルールを含み得る。

【００２２】

ターゲット表現１１３は、ログを含むおよび／または生成する、顧客環境内の個々のコンポーネントである「ターゲット」を識別する。これらのターゲットは、顧客環境において特定のコンポーネント／ホストに関連付けられる。例示的なターゲットは、１つ以上のログおよび／または１つ以上のホストに関連付けられる特定のデータベースアプリケーションであり得る。

【００２３】

１２２での次のアクションは、ユーザ構成に従ってログデータをキャプチャすることである。ログデータは、データベース管理システム、データベースアプリケーション、ミド

ルウェア、ハードウェアログ、オペレーティングシステムログ、アプリケーションログ、アプリケーションサーバログ、データベースサーバログ、および、システムまたはアプリケーションの挙動をモニタする任意の他のタイプのログといった、任意のログ生成ソース位置から生じ得る。

【００２４】

いくつかの場合では、ログルール１１１とターゲット表現との間の関連性が処理のために顧客ネットワーク１０４へ送信される。ログ分析システムのエージェントが、ホスト１０９上において適切なログからデータを収集するようホスト１０９の各々上に存在する。

【００２５】

いくつかの実施形態では、キャプチャされたデータに対してデータマスキングが実行され得る。マスキングは、顧客ネットワークを離れる前に顧客データを保護するよう、収集時間において実行される。たとえば収集されたログデータ中のさまざまなタイプの情報（たとえばユーザ名および他の個人情報）は、サーバへ送信される前にマスキングされるには十分にセンシティブであり得る。そのようなデータについて、サーバに収集される前に除去され得るか、および/または、プロキシデータに変更され得るパターンが識別される。これにより、センシティブなデータを隠しつつ、データが分析目的のために使用されることが可能になる。いくつかの実施形態は、センシティブなデータを永久的に除去する（たとえば、すべてのそのようなデータを「＊＊＊」記号に変更する）か、または、オリジナルデータが回復され得るようにマッピングされるデータに変更され得る。

【００２６】

１２４において、収集されたログデータが顧客ネットワーク１０４からログ分析システム１０１に送達される。顧客ネットワーク１０４中の複数のホスト１０９が、収集されたデータを少数の１つ以上のゲートウェイ１０８に提供し、次いで、当該ゲートウェイ１０８は、ログ分析システム１０１におけるエッジサービス１０６にログデータを送信する。エッジサービス１０６は、収集されたデータを１つ以上の顧客ネットワークから受け取り、任意のインテイク処理（intake processing）を実行し（たとえばクラスタ間でのメッセージ変動性の要素を欠く正規化されたメッセージまたはスケルトンメッセージに各メッセージを変換するために文法ルールを適用し、変換されたメッセージのハッシュを使用して識別される初期クラスタに各変換されたメッセージを割り当て）、ログ処理パイプライン１０７によるさらに別の処理のためにインバウンドデータストアにデータを配置し得る。

【００２７】

１２６において、ログ処理パイプライン１０７は、収集されたログデータに対して一連のデータ処理および分析動作を実行する。さまざまな場合において、これらの処理および分析動作は、データを格納する前であって、および／または、データストアから抽出されたデータに対してアクションを実行することによって実行されるアクションを含み得る。たとえば、１つ以上のログメッセージは、（たとえばソースからログメッセージを受け取る際に）インジェスト時間（ingest time）に初期クラスタに割り当てられ得、その後、初期クラスタリングを修正または補足し、クラスタリングに基づいて統計および／またはプレゼンテーションを生成するクエリに応答してログメッセージが抽出され得る。

【００２８】

次いで、１２８において、処理されたデータがデータストレージデバイス１１０へ格納される。コンピュータ読取可能ストレージデバイス１１０は、コンピュータ読取可能ストレージデバイス１１０に位置するデータへの実行可能なアクセスを可能にするハードウェアおよびソフトウェアの任意の組み合わせを含む。たとえば、コンピュータ読取可能ストレージデバイス１１０は、オペレーティングシステムによって動作可能に管理されるコンピュータメモリとして実現され得る。コンピュータ読取可能ストレージデバイス１１０中のデータは、データベースオブジェクト、クラウドオブジェクトおよび／またはファイルシステム中のファイルとして実現され得る。いくつかの実施形態では、処理されたデータは、（たとえばＳＯＬＲクラスタとしての）テキスト／インデックス付データストア１１０ａと、（たとえばＨＤＦＳクラスタとしての）生／履歴データストア１１０ｂとの両方

内に格納される。

【００２９】

ＳＯＬＲクラスタは、Ａｐａｃｈｅ（商標）オープンソースローカルサーチプラットホームに対応する。ＳＯＬＲクラスタは、ＨＤＦＳクラスタに格納されるデータのフルテキストインデキシングおよびサーチを実行するためにサーチライブラリを使用し得る。ＳＯＬＲクラスタは、他のプログラムおよびアプリケーションにサーチ機能をインターフェイス接続するために、さまざまな言語と互換性のあるＡＰＩを提供し得る。インデキシングはほぼリアルタイムで実行され得る。クラスタは、フォールトトレランスおよび可用性を促進するようにサーバの集合上で動作し得る。インデキシングおよびサーチタスクは、当該サーバの集合に亘って分散され得る。

【００３０】

ＨＤＦＳクラスタは、Ｈａｄｏｏｐ分散ファイルシステムクラスタに対応する。ＨＤＦＳクラスタは、ストレージ（たとえば直接的に取り付けられたストレージ）をホストし、ユーザアプリケーションによって規定されるタスクのようなタスクを実行するために、多くのサーバ（たとえば何千）を含み得る。ＨＤＦＳクラスタは、クラスタのネームスペースを管理するための単一のマスタサーバを有するマスタ／スレーブアーキテクチャを含み得る。ファイルは、ＨＤＦＳクラスタの複数のデータノード（DataNodes）に格納されるよう、ブロックに分割され得る。マスタサーバは、ファイル操作（たとえば、開く、閉じるなど）を実行し得、どのブロックがどのデータノードに格納されるべきであるかを決定し得る。マスタサーバは、対応するファイル操作の受信に応答して、データを読み出すまたは書き込む要求についてデータノードと通信し得る。

【００３１】

１３０において、レポートメカニズム／ＵＩ１１５を使用して、処理されたデータに対してレポートが実行され得る。図２に示されるように、レポートＵＩ２００は、ログサーチ機構２０２、１つ以上のダッシュボード２０４、および／または、処理されたログデータを分析／閲覧するための任意の好適なアプリケーション２０６を含み得る。そのようなレポートコンポーネントの例は以下により詳細に記載される。

【００３２】

１３２において、処理されたデータに対してインシデント管理が実行され得る。アラート条件の検出の際に、インシデント管理メカニズム１１７がインシデント／アラートをユーザの指定された集合に通知を提供するように、１つ以上のアラート条件がログ分析システム内に構成され得る。

【００３３】

１３４において、補正アクションエンジン１１９が、顧客ネットワーク１０４内で行われるべき任意の必要なアクションを実行し得る。たとえば、データベースシステムがダウンしているというログエントリが受け取られ得る。そのようなログエントリが識別された場合、可能な自動補正アクションは、データベースシステムを復旧することを試みることである。顧客は、この状況に対処するために補正アクションスクリプトを作成し得る。補正アクションを実行するためにスクリプトを実行するようトリガーが実行され得る（たとえば、当該トリガーは、スクリプトを実行するよう顧客ネットワーク上のエージェントへ命令を送信させる）。代替的な実施形態では、当該状況についての適切なスクリプトが、実行されるようサーバから顧客ネットワークへとプッシュダウンされる。さらに、１３６において、任意の他の付加的な機能および／またはアクションが、処理されたデータに少なくとも基づいて適切なように行われ得る。

【００３４】

図３Ａは、ホスト環境３４０におけるログ分析システムの内部構造と、ログ分析システムとインタラクションする顧客環境３４２内のコンポーネントとのさらに詳細な図を提供する。このアーキテクチャ３００は、大量のログデータのインジェストに対応可能であるログモニタリングのためのフローを提供するように構成される。

【００３５】

単一の顧客ホスト／サーバ３４４内の顧客環境３４２において、ＬＡ（ログ分析（log analytics））エージェント３３３は、ログモニタリング構成データ３３２（たとえばスニッファ構成またはターゲットサイド構成マテリアル）を取得し、ログファイルスニッファ３３６（本願明細書において「ログコレクタ」とも称される）を呼び出して１つ以上のログファイル３３８からログデータを収集する。

【００３６】

ログファイルスニッファ３３６にインターフェイス接続するようにデーモンマネージャ３３４が使用され得る。ログファイルスニッファ３３６は、ホストマシン３４４上の１つ以上のログファイル３３８から読み出しを行う。デーモンマネージャ３３４はログコンテンツを取得し、ＬＡエージェント３３３に戻され得るようにログコンテンツをパッケージ化する。なお、システムは、異なる種類の任意数のスニッファを含み得、ログスニッファ３３６は単に、システムにおいて使用され得る単一のタイプのスニッファの例である。したがって、レジストリ、データベース、ウィンドウイベントログなどをモニタするスニッファといった他のタイプのスニッファが本発明のさまざまな実施形態において使用されてもよい。さらに、いくつかの実施形態におけるログスニッファは、集合的な／圧縮されたファイル（たとえばｚｉｐファイル）を処理するように構成される。

【００３７】

ＬＡエージェント３３３は、ゲートウェイエージェント３３０に、収集されたログデータを送信する。ゲートウェイエージェント３３０は、複数の顧客ホスト／サーバから収集されるログデータをパッケージ化し、複数のホストからのログコンテンツを集合するアグリゲータとして本質的に動作する。その後、パッケージ化されたコンテンツは、ゲートウェイエージェント３３０からエッジサービス３０６に送信される。エッジサービス３０６は、任意数の異なる顧客環境３４２からの複数のゲートウェイエージェント３３０から大量のデータを受信する。

【００３８】

エッジサービス３０６において受信され得る潜在的に大きなボリュームのデータが与えられると、当該データは、初期クラスタに各ログメッセージを割り当てるようにすぐに処理され得、インバウンドデータストレージデバイス３０４（「プラットホームインバウンドクラスタリングストア」）に格納され得る。いくつかの場合では、初期処理または予備処理が、インジェスト時間において実行され得る。インジェスト時間は、データの格納に対応する時間（たとえば、データの格納の前の時間、データの格納の少し後もしくは直後の時間、または、データの格納と同時の時間）を含み得る。初期処理または予備処理は（たとえば）、データのどの部分が非変動コンポーネントであるかを検出することと、メッセージにおいて検出された非変動コンポーネントに基づいて各ログメッセージについて初期クラスタを決定することとを含み得る。たとえば、初期クラスタの識別子を生成するよう、各非変動コンポーネントの値にハッシング技術が適用され得る。その後、ログメッセージは初期クラスタの識別子に関連付けられて格納され得るか、または、ログメッセージが初期クラスタに関連付けられていることを示す他のクラスタデータが格納され得る。クラスタ割り当てはさらに、その後のリソース可用性の時間中に発生する処理のようなその後の処理中に、および／または、関連付けられたログメッセージに対応するかもしくは潜在的に対応するデータについてのクエリを受信することに応答して、改良、増強および／または使用され得る。

【００３９】

したがって、いくつかの場合では、キューが管理および維持される。キュー要素は、クラスタ割り当てが改良、増強および／または使用される１つ以上のログメッセージに対応する。（たとえば）キュー要素の初期格納の後、および／または、１つ以上の関連付けられるログメッセージに対応するかもしくは潜在的に対応するデータについてのクエリを受信することに応答して、要素がキューに加えられ得る。キューはログ処理パイプライン３０８のために使用され得る。

【００４０】

インバウンドデータストア内で処理されるべきアイテムを管理するためにデータ構造が提供される。いくつかの実施形態では、キュー内の処理されるべきアイテムをトラッキングするよう、（たとえば、Ｋａｆｋａプロダクトを使用して実現される）メッセージングプラットホーム３０２が使用され得る。ログ処理パイプライン３０８内において、キューコンシューマ３１０が、処理されるべきキュー内の次のアイテムを識別し、当該次のアイテムが次いでプラットホームインバウンドストアから抽出される。キューコンシューマ３１０は、プロセス、スレッド、ノードまたはタスクといった、キューからシステム内のワークを処理することが可能な任意のエンティティを含む。

【００４１】

抽出されたログデータは「パース（parse）」ステージ３１２を経て、ログエントリがパースされ、特定フィールドまたはコンポーネントへ分割される。ログのために構成された「ログタイプ」は、どのように所望のフィールドへとログエントリを分割するかを指定する。

【００４２】

「クラスタ」ステージ３１３では、ログデータはさらに、クラスタに個々のログメッセージを割り当てるよう分析される。具体的には、ログメッセージが（たとえば３０４において）インテイク処理中に割り当てられた複数の初期クラスタが、当該初期クラスタのうちのいくつかがともにマージされるべきか否かを決定するためにアセスメントがなされ得る。当該アセスメントは、各クラスタについて１つ以上の代表的なサンプルを識別することと、対の量的比較アセスメントを実行することとを含み得る。対の比較アセスメントを介してアセスメントがなされたクラスタ対は、同じ数または同様の数のコンポーネント（またはワード）を有するログメッセージを有するクラスタを含み得る。いくつかの場合では、クラスタの各対は、同じであるか、または、（たとえば、予め規定されているか、デフォルト数であるか、または、ユーザによって識別される）しきい値数未満だけ互いに異なる数のコンポーネントに関連付けられるクラスタを含み、上記アセスメントを使用して評価される。比較アセスメントは、反復しておよび／または構造化された態様で（たとえば同じ数のコンポーネントを有する対が、異なる数のコンポーネントを有する対を評価する前に評価されるように）実行され得る。

【００４３】

上記対の量的比較アセスメントは、たとえば、代表的なメッセージを使用して類似性メトリック（similarity metric）を生成することと、当該メトリックが（たとえば、予め規定されているか、デフォルト数であるか、ユーザによって識別される）しきい値メトリックを上回っているか否か決定することとを含み得る。類似性メトリックは（たとえば）、代表的なメッセージが、同じ（または同様の）数のコンポーネント、同じ（または同様の）数の変動（または非変動）コンポーネント、１つ以上の非変動コンポーネントの各々のコンテンツ、および、１つ以上の変動コンポーネントの特徴（たとえばフォーマット、文字タイプあるいは長さ）などを含むか否かに基づき得る。類似性メトリックは、クラスタ間のメッセージ同士の間の相関係数を生成することに基づき得るか、または、（たとえば、技術が、主コンポーネント分析または独立コンポーネント分析のようなコンポーネント分析を使用することを含んでいる場合）クラスタの代表的なメッセージが同じクラスタに割り当てられるか、もしくは、コンポーネントをシェアする程度までメッセージのより大きな集合を使用してクラスタリング技術を実行することに基づき得る。

【００４４】

「正規化」ステージ３１４では、識別されたフィールドが正規化される。たとえば、「時間」フィールドは異なるログにおいて任意数の異なる態様で表わされ得る。この時間フィールドは、単一の認識可能なフォーマット（たとえばＵＴＣフォーマット）へ正規化され得る。別の例として、「ｅｒｒｏｒ」という用語は、異なるシステム上において異なる態様で表わされ得る（たとえば、すべてが大文字の「ＥＲＲＯＲ」、すべてが小文字の「ｅｒｒｏｒ」、最初の文字が大文字である「Ｅｒｒｏｒ」、または、省略形の「ｅｒｒ」）。この状況においては、異なる用語形態／タイプが単一のフォーマット（たとえばすべ

てが小文字で省略形でない用語「ｅｒｒｏｒ」）へ正規化される必要がある。
【００４５】
「変換」ステージ３１６は、ログデータから新しいコンテンツを合成するために使用され得る。例として、「タグ」は、ログエントリに関する付加的な情報を提供するよう、ログデータに加えられ得る。別の例として、タグは、ログメッセージが割り当てられるクラスタを識別し得る。
【００４６】
「条件評価」ステージ３１８は、ログデータに対する特定条件について評価するために使用される。このステージは、ログデータ内のパターンを識別し、ログ内のアラート条件を作成／識別するために実行され得る。このステージにおいて、たとえば管理者／顧客に送信されるＥメール／テキストメッセージ／コール、または、別のシステムもしくはメカニズムへのアラートを含む任意のタイプの通知が実行され得る。一例として、条件は、所与のクラスタに割り当てられたログメッセージの量（たとえば数またはパーセンテージ）が（たとえば、ユーザ、クライアントまたはルールによって、固定されるか、あらかじめ規定されるかまたは規定される）しきい値を越えたことを検出するといった、クラスタ割り当ての変更に対応するイベントを規定し得る。しきい値を越えることは、たとえば、下側しきい値を下回るか、または、上側しきい値を上回ることである。別の例として、条件は、たとえば、時系列スロープについてのしきい値を識別すること、または、クラスタに割り当てられたログメッセージのカウントもしくはパーセンテージにおける２つの時間ビン（time bin）の間での差についてのしきい値を識別することによって、所与のしきい値に割り当てられるログメッセージの量が変化している程度に対応するイベントを規定し得る。さらに別の例として、条件は、（たとえばカーブフィット係数がしきい値量内に存在するのに十分に類似しているか否か決定し、時系列中の１つ以上のピークの時間が規定されたしきい値時間内にあるか否か決定し、クラスタの時系列同士間の相関係数がしきい値を越えているか否か決定し、および／または、個々のクラスタの各々の時系列の変動性と、時系列の合計の変動性との間の差がしきい値を上回っているか否かを決定することによって）複数のクラスタの各々の時系列が同様の形状を有することを示すイベントといった、複数のクラスタ割り当てに対応するイベントを規定し得る。
【００４７】
その後、ログライタ３２０は、処理されたログデータを１つ以上のデータストア３２４に書き込む。いくつかの実施形態では、処理されたデータは、（たとえばＳＯＬＲクラスタとしての）テキスト／インデックス付データストアと、（たとえばＨＤＦＳクラスタとしての）生および／または履歴データストアとの両方に格納される。ログライタはさらに、別の処理ステージ３２２および／または下流の処理エンジンにログデータを送信し得る。
【００４８】
図３Ｂに示されるように、いくつかの実施形態は、クライアントサイドのエージェント３３３を通じて進める必要なくログデータを収集するよう、サイドローディングメカニズム３５０を提供する。このアプローチでは、ユーザは、ローカルシステム上の１つ以上のファイルを選択するためにサーバにログインする。システムは、サーバにおいてそのファイルをロードし、（たとえばユーザにログタイプを提供させ、可能性のあるログタイプを試み、異なるログタイプによってロールすることによって、または、ログタイプの経験的（educated）「推測」を行うことによって）そのファイルを通じてスニッフィングを行う。次いで、スニッフィング結果が、前述のように、エッジサービスおよび処理に渡される。図３Ｃの実施形態では、サイドローディングメカニズム３５０のみがログファイルを収集するために存在する。すなわち、エージェント／スニッファエンティティはいずれも、クライアントサーバ３４４上にインストールされないか、および／または、クライアントサーバ３４４上にて必要とされない。
【００４９】
図４は、オンプレミスアプリケーションをクラウドシステムにエクスポートするための例示的なネットワーク環境４００を示す。ネットワーク環境４００はローカルシステム４

１０およびクラウドシステム４５０を含み得る。ローカルシステム４１０は、データベース４２０、アプリケーションサーバ４３０およびウェブサーバ４４０といった、１つ以上の内部サブシステムを含み得る。ローカルシステム４１０の内部サブシステムの各々は、クラウドシステムに含まれていない場合があり得る。いくつかの実現例では、データベース４２０、アプリケーションサーバ４３０およびウェブサーバ４４０の物理コンポーネントは、会社に関連付けられる施設においてオンプレミスであり得る。データベース４２０、アプリケーションサーバ４３０およびウェブサーバ４４０の各々は、少なくとも部分的に、１つ以上のアプリケーションを実行し得る。

【００５０】

「オンプレミス」という用語は会社の施設に位置するシステムを指すように使用されるが、本開示はそれに限定されないということが認識されるであろう。たとえば、ローカルシステム４１０およびその内部サブシステムは、会社の施設において物理的にオンプレミスであってもよく、物理的にオンプレミスでなくてもよいが、内部サブシステムは、会社に関連していないエリアに亘って分散され得る（たとえば賃貸したサーバ）。さらに、ローカルシステム４１０は、会社によって管理される１つ以上のアプリケーションを実行するために使用され得る。

【００５１】

いくつかの実現例では、クラウドシステム４５０は、リポジトリ４６０、クラウドマネージャ４７０、クラウドマネージャアプリケーションスタック４７５、および、中央サーバ４８０を含み得る。多層アプリケーションスタックへのアップデートは、中央サーバ４８０に周期的に格納され得る。たとえば、アップデート４９０は、１つ以上のアプリケーションをアップデートするコードを含み得る。アップデート４９０の非限定的な例は、バグフィックス、アプリケーションへの特徴の追加、アプリケーションへの機能の追加、特徴の修正、機能の修正、特徴の削除、および／または、機能の削除を含み得る。アップデート４９０は、１人以上のプログラマによって規定され得る（たとえばコードが記述された）か、または、いくつかの場合では、アップデート４９０は、１つ以上のルールおよび／または機械学習技術に基づいて自動的に作成され得る。新しいアップデートが周期的に規定されると、当該アップデートはアップデート４９０として中央サーバ４８０に格納され得る。中央サーバ４８０はさまざまなアプリケーションのために複数のアップデートを格納し得る。リポジトリ４６０は、ローカルシステムからクラウドシステム４５０にエクスポートされている入力アプリケーションのデプロイメントパッケージを格納するための１つ以上のストレージシステムを含み得る。たとえば、デプロイメントパッケージは、システムに亘ってエクスポート可能なアプリケーションのシリアライズされたバージョンであり得る。クラウドマネージャ４７０は、クラウドマネージャアプリケーションスタック４７５の１つ以上の局面を管理するように構成される１つ以上のアプリケーション、システムまたはエンジンであり得る。たとえば、クラウドマネージャアドミニストレータは、クラウドマネージャアプリケーションスタック４７５を使用してクラウドマネージャアプリケーションを実行するよう、クラウドマネージャ４７０を動作し得るか、または、クラウドマネージャ４７０にアクセスし得る。クラウドマネージャ４７０は、クラウドシステム４５０において環境のデプロイメントを提供するアプリケーションであり得る。たとえば、クラウドマネージャ４７０は、自動化された態様でクラウドシステム４５０上のさまざまなアプリケーションをデプロイし得る。クラウドマネージャアプリケーションに関する付加的な詳細については、「クラウドベースシステムを使用するアプリケーションの分散したバージョニング（Distributed Versioning of Applications Using Cloud-Based Systems）」という名称を有する２０１７年５月１６日に出願された米国連続番号６２／５０７，０８６号を参照すること。その開示は、すべての目的について、本願明細書において全文、参照により援用される。

【００５２】

いくつかの例では、ローカルシステム４１０のネットワークアドミニストレータは、（たとえばエンタープライズアプリケーションのスケーラビリティを増加させるために）ク

ラウドへエンタープライズアプリケーションをエクスポートする必要があり得る。本開示の実施形態は、ローカルシステムのネットワークアドミニストレータが内部サブシステムからクラウドシステム４５０にエンタープライズアプリケーションをエクスポートすることを可能にする。いくつかの例では、アプリケーションのライフサイクルの間、アプリケーションアドミニストレータは、オンプレミスアプリケーションをクラウドシステム４７０に移行させ得る。移行プロセスの間、アプリケーションアドミニストレータは、デプロイメントパッケージを使用してオンプレミス環境をエクスポートし、クラウドシステム４５０へデプロイメントパッケージをインポートし得る。クラウドマネージャ４７０は、エンタープライズアプリケーションのデプロイメントパッケージを読み出し得、クラウドシステム４５０上で当該アプリケーションをデプロイし得る。

【００５３】

新しい特徴セットまたは新しいイメージが、中央サーバにおいて公開されるか、中央サーバにおいて利用可能であるか、または、中央サーバに格納され得るということが認識されるであろう。新しい特徴セットまたは新しいイメージは、アプリケーションの最新バージョンのすべての新しい特徴を含み得る。その新しい特徴セットまたは新しいイメージは、アプリケーションの古いバージョンと比較され得、新しい特徴セットまたは新しいイメージは、自動化されたワンステッププロセスでアプリケーションの古いバージョンに移動され得る。たとえば、新しい特徴セットまたは新しいイメージは、ユーザからの単一の入力により、アプリケーションの古いバージョンに適用され得る。

【００５４】

さらに、クラウドシステム４５０はメタデータ駆動アーキテクチャとして実現され得るということが認識されるであろう。たとえばソフトウェアコンポーネント、ｊａｖａコード、ｐｙｔｈｏｎコード、任意の他の好適なソースコードは、ランタイム環境内においてデータベースオブジェクトとして個々に表わされ得る。この例を引き続き参照して、アプリケーションの基礎となるソースコードに含まれるソフトウェアコンポーネントは、ファイルリファレンスオブジェクトと称され得る。各ファイルリファレンスオブジェクトは、メタデータテーブルに格納されるメタデータに関連付けられ得る。たとえば、ファイルリファレンスオブジェクトのメタデータは、ファイルリファレンスオブジェクトの内部バージョン識別子を含み得る。内部バージョン番号は、ファイルリファレンスオブジェクトおよび／またはファイルリファレンスオブジェクトのバージョンを一意に識別する一意識別子であり得る。ファイルリファレンスオブジェクトの新バージョンが作成される場合（たとえばファイルリファレンスオブジェクトが修正またはアップデートされる場合）、ファイルリファレンスオブジェクトの新バージョンが新しい内部バージョン番号に関連付けられ得る。いくつかの実現例において、ファイルリファレンスオブジェクトは、バイナリオブジェクトとして表わされ得、そのバイナリオブジェクト（たとえば一意識別子）のリファレンスは、バイナリオブジェクトおよび／またはバイナリオブジェクトのネットワーク位置にリファレンスを関連付けるよう、データベースに格納され得る。

【００５５】

いくつかの実現例では、コードのブロックは、データベースオブジェクトに対応し得るか、データベースオブジェクトと称され得るか、または、データベースオブジェクトであり得る。データベースオブジェクトは、内部一意識別子に対応し得、および／または、データベースオブジェクトの属性を記述する付加的な内部メタデータに対応し得る。たとえば、データベースオブジェクトに関連付けられるメタデータとして格納されるデータベースオブジェクトの属性は、コードのブロックがどのように実行されるか、および、コードのブロックまたはデータベースがどのようにランタイムにて表わされるかなどを含み得る。本開示はファイルリファレンスオブジェクトおよびデータベースオブジェクトに限定されず、その代わりに、任意のタイプのオブジェクトが本開示の実施形態において使用されてもよいということが認識されるであろう。

【００５６】

さらに、オブジェクト（たとえばデータベースオブジェクト）が作成されると、オブジ

ェクトのバージョンが生成され、オブジェクトの属性を規定または記述するメタデータがデプロイメントテーブルまたは表現テーブルに格納される。たとえば、２つのオブジェクトについて表現テーブルに格納されるメタデータについてクエリ送信することによって、アプリケーションおよび／または変更アシスタントツールが、２つのオブジェクト間の差および／または類似点を識別するよう構成され得る。非限定的な例として、アプリケーションおよび／または変更アシスタントツールは、ターゲット（たとえばアプリケーションの現在実行されているインスタンス）をソース（たとえばアプリケーションの新しいインスタンス）と比較して、ターゲットとソースとの間の差を識別するよう構成され得る。比較の結果は、ソースがファイルリファレンスオブジェクトＡＢＣ，バージョン１２（一意識別子ＡＢＣ１２に関連付けられる）を含み、ターゲットが、ファイルリファレンスオブジェクトＡＢＣ，バージョン１１（一意識別子ＡＢＣ１１に関連付けられる）を含むという決定であり得る。したがって、アプリケーションは、一意識別子ＡＢＣ１２およびＡＢＣ１１の各々に関連付けられるメタデータを格納する表現テーブルにアクセスし得る。その後、アプリケーションは、識別子ＡＢＣ１２に関連付けられるファイルリファレンスオブジェクトをアップデートデータパッケージへとシリアライズし、現在実行されているインスタンスのファイルリファレンスオブジェクトＡＢＣのバージョン１１がバージョン１２にアップデートされ得るように、現在実行されているインスタンスにそのアップデートデータパッケージを適用する（たとえば、インストールする）。いくつかの実現例では、ＡＢＣ１２とＡＢＣ１１との間で異なる任意の他のメタデータが、識別され、現在実行されているインスタンスにインストールされ得、これにより、新しいインスタンスにインストールされたアップデートのすべてを含むよう、現在実行されているインスタンスをアップデートする。さらに、アップデートデータパッケージは、システム同士の間でエクスポート可能なファイルであり得る。アップデートデータパッケージは、現在実行されているインスタンスに新しいインスタンスの全体をエクスポートする代わりに、表現テーブルに格納されるメタデータの特徴が、現在実行されているインスタンスにエクスポートされることを可能にする。有利なことに、アップデートされたオブジェクトのエクスポートは単に、現在実行されているインスタンスに送信または移動されるエクスポートファイルのサイズを低減することによってネットワークリソースの使用を改善する。

【００５７】

図５は、多層アプリケーションスタックを自動的にアップデートするための処理フローを示すブロック図である。周期的に、多層アプリケーションスタックへのアップデートは中央サーバ４８０に格納され得る。アプリケーション環境５３０は、中央サーバ４８０からアップデート（たとえばアップデート４９０）を抽出し格納するように構成されるリポジトリ４６０を含み得る。アップデートの非限定的な例はアップデートイメージおよびパッチセットを含み得る。たとえば、アップデートイメージ（たとえば四半期につき約１回発行するメジャーアプリケーションアップデート）は、アプリケーションの特定のバージョンのすべてのバグフィックスおよび新しい特徴を含む１つ以上のデータパッケージであり得る。パッチセット（たとえば、バグをフィックスするためのマイナーアプリケーションアップデートであって、メジャーアプリケーションアップデート同士の間に発行するマイナーアプリケーションアップデート）は、最後のアップデートイメージがリリースされてからレポートされた個々のバグフィックスを含む１つ以上のデータパッケージであり得る。アップデートイメージがリリースされるごとに、アップデートイメージのソースファイルは中央サーバ４８０に格納され得る。たとえば、図５では、中央サーバ４８０はアップデートイメージまたはパッチセットとしてアップデート４９０を格納し得る。中央サーバ４８０はクラウドシステム４５０でのすべてのアプリケーションスタックにアクセス可能であり、したがって、中央サーバ４８０は、任意数のアプリケーションスタックについて任意数のアップデートを格納し得る。図５を参照して、アップデート４９０は特定の多層アプリケーションスタックのためのアップデートであり得る。

【００５８】

いくつかの実現例では、アップデート４９０に関連付けられるアプリケーションスタッ

クは、周期的に（規則的または不規則的な間隔で）アップデートのために中央サーバ４８０にクエリ送信するように構成され得る。いくつかの実現例では、中央サーバ４８０は１つ以上のアプリケーションスタックに通知を送信し得る。通知は、アップデート４９０がたとえば中央サーバ４８０からのダウンロードにより現在利用可能であるということをアプリケーションのユーザまたはアプリケーション自体に通知するよう機能し得る。図６は、新しいアップデートが中央サーバ４８０において利用可能であるという通知を示す例示的なインターフェイスを示す。新しいアップデートが中央サーバ４８０において利用可能な場合、アプリケーションスタックの現在実行されているインスタンスのバージョンは古くなっている場合があり得る。本発明のある実施形態は、新しいアップデートが中央サーバ４８０において利用可能な場合、アプリケーションが最新バージョンに自動的にアップデートすることを可能にする。

【００５９】

いくつかの実施形態では、リポジトリ４６０は、中央サーバ４８０から抽出されたアップデートを格納し得る。たとえば、リポジトリ４６０は、任意のアップデートについて中央サーバ４８０にクエリ送信し得、アップデート４９０が、利用可能な場合、リポジトリ４６０へダウンロードされ得る。さらに、アプリケーション環境５３０は、多層アプリケーションスタックに関連付けられる現在実行されているインスタンス５００を含み得る。たとえば、現在実行されているインスタンス５００はアプリケーション環境５３０において実行され得、また、プロダクション環境においてアプリケーションスタック上で実行され得る。たとえば、アプリケーションがユーザにサービスを提供することを可能にするよう、実際のユーザデバイスが、プロダクション環境において現在実行されているインスタンス５００と通信している場合がある。アプリケーションがプロダクション環境において実行されているので、アップデートのための如何なるサーバダウンタイムも負のインパクトを有する。ユーザがアプリケーションおよびその対応するサービスにアクセスすることができない場合があり得るからである。有利なことに、新規性の技術的なポイントとして、本発明の実施形態によって、アップデートのサイズ（たとえば１個のバグフィックスまたは１０，０００個のバグフィックス）に関わらず、アプリケーションが最小のサーバダウンタイムでアップデートされることが可能になる。

【００６０】

上述したように、現在のインスタンス５００はアプリケーションの古いバージョンであり得る。たとえば、現在のインスタンス５００はアップデート４９０を含んでいないアプリケーションのバージョンであり得る。アップデート４９０により現在のインスタンス５００をアップデートするために、アプリケーション環境５３０は新しいインスタンス５１０を生成し得る。現在のインスタンス５００および新しいインスタンス５１０の各々は、予め規定された構成セッティング（たとえばＣＰＵパラメータおよびＲＡＭアロケーションなど）を有する同じまたは異なるバーチャルマシン上で実行され得る。さらに、アプリケーション環境５３０は、新しいインスタンス５１０にアップデート４９０をインストールするように構成され得る。したがって、新しいインスタンス５１０は、アプリケーションの最新バージョン（たとえばアップデート４９０がインストールされユーザに利用可能であるアプリケーションのバージョン）であり得る。

【００６１】

アプリケーション環境５３０は、現在のインスタンス５００と新しいインスタンス５１０との間に任意のデルタ５２０（たとえば差）が存在するか否かを識別するために、新しいインスタンス５１０と現在のインスタンス５００とを比較するように構成され得る。いくつかの実施形態では、デルタ５２０を識別するために現在のインスタンス５００と新しいインスタンス５１０とを比較することは、各インスタンスに関連付けられるバグ番号を比較することを含み得る。たとえば、バグ番号は、アプリケーションのソースコード内においてフィックスされた（またはソースコードがバグに対処するために修正された）１つ以上のバグに対応する特定のバグレポートを識別する一意識別子であり得る。たとえば、インスタンスのバグ番号はインスタンスに関連付けられるメタデータとして格納され得る

。この例において、現在のインスタンス５００についてのメタデータは、バグ番号１および３が以前にフィックスされたか、または、現在のインスタンス５００に適用されたということを示すデータを格納し得る。さらに、新しいインスタンス５１０についてのメタデータは、バグ番号１、３および６が新しいインスタンス５１０に適用されたことを示すデータを格納し得る。この例において、新しいインスタンス５１０は現在のインスタンス５００の新しく生成されたインスタンスであるので、バグ番号１および３は、新しいインスタンス５１０に既に適用されており、したがって、バグ番号６がアップデートとして検出されることになる。たとえば、アプリケーション環境５３０は、これらの２つのインスタンス同士間のデルタ５２０がバグ番号６であるということを決定するよう、現在のインスタンス５００のメタデータを新しいインスタンス５１０のメタデータと比較し得る。

【００６２】

したがって、アップデート４９０が、現在のインスタンス５００において実行されているアプリケーションのバージョンの後の次のバージョンまたは後継である状況において、アップデート４９０はバグ番号６に対応し、アップデート４９０がバグ番号６に関連付けられる１つ以上のバグをフィックスするということを示す。デルタ５２０がバグ番号６に対応することをアプリケーション環境５３０が識別すると、アプリケーション環境５３０は、バグ番号６のバグレポートによって対処されるバグに関連付けられる１つ以上のオブジェクトにアクセスし得る。アプリケーション環境５３０は１つ以上のオブジェクトをエクスポートし得、また、バグ番号６に関連付けられるバグフィックスをアプリケーション環境５３０のソースコードへ組み込み得る。ウェブサーバ、アプリケーションサーバおよび／またはデータベースサーバ上で実行される特定のソフトウェアコンポーネントにおいてバグがフィックスされ得るので、１つ以上のオブジェクトが、関連するソフトウェアコンポーネントに適用され得る（たとえば、バグがウェブサーバのソフトウェアコンポーネントにおいてフィックスされる場合、１つ以上のオブジェクトが、現在のインスタンス５００に関連付けられるウェブサーバのソフトウェアコンポーネントに適用される）。有利なことに、プロダクション環境において実行されておりユーザにアクセス可能である現在のインスタンス５００は、最小のダウンタイムで自動的にアップデートされ得る。

【００６３】

比較が実行されても、現在のインスタンス５００がユーザへのサービスを提供し続ける（たとえば、サーバのダウンタイムがない）ことが可能であるということが認識されるであろう。アプリケーションスタックがデルタ５２０に基づいてアップデートされている場合、アプリケーション環境５３０はロックを開始し得、これにより、規定された期間中のトランザクションを防止する。ロック中は、現在のインスタンス５００はデルタ５２０に関連付けられる１つ以上のオブジェクトによりアップデートされ得る。現在のインスタンス５００がアップデートされた後、現在のインスタンス５００がリブートされ得、アップデート４９０が利用可能であり得るか、または、現在のインスタンス５００に組み込まれ得るように、ロックが解除され得る。

【００６４】

アプリケーションの新バージョンがリリースされると、アプリケーションの新しいイメージが中央サーバに公開される（たとえば、アップデートが中央サーバに格納される）ということが認識されるであろう。新しいイメージは、アプリケーションの新バージョンの特徴のすべてを含む。アプリケーションは、新バージョンの特徴をアプリケーションの新しいインスタンスに提供するように構成され得る。変更アシスタントツールは、新しいインスタンスのメタデータを、アプリケーションの新バージョンより古いバージョンとなるアプリケーションの現在実行されているインスタンスと比較するように構成され得る。その後、アプリケーションは、データベースオブジェクトおよびバイナリランタイムの両方を含み得るアップデートされたソースコードまたはアップデートされたバイナリコードのメタデータをインストールし得る。さらに、アップデートされたソースコードのメタデータは、アップデートデータパッケージとして、シリアライズされた形態で、現在実行されているインスタンスに移動され得、現在実行されているインスタンスにインストールまた

は適用され得る。

【００６５】

図６～図８は、ユーザが多層アプリケーションスタック上で実行されるアプリケーションにアクセスする場合、ユーザデバイス上で提示される例示的なインターフェイスを示す。たとえば、インターフェイス６００は、「新しいフィックスがクラウドマネージャについて利用可能であり、適用する準備ができています」という通知を提示する。いくつかの実現例では、ユーザがアプリケーションのセッティング構成ページにアクセスする場合、インターフェイス６００はユーザデバイス上に提示され得る。非限定的な例として、いくつかの構成は、ファイルサーバ構成、バーチャルマシン構成およびアップデート管理構成といったセッティング構成ページ上で規定され得る。新しいアップデートが中央サーバ（たとえば中央サーバ４８０）に格納され利用可能な場合、インターフェイス６００は通知を提示し得るが、いくつかの実施形態では、中央サーバにおいて利用可能な新しいアップデートは、自動的にダウンロードされてアプリケーションに適用され、したがって、通知は必要でなくなる。いくつかの実施形態では、利用可能なアップデートが中央サーバからダウンロードされ、現在実行されているインスタンスにインストールされた後（たとえば比較が実行されてデルタが決定された後）でユーザに通知されてもよい。図７は、中央サーバ（たとえば中央サーバ４８０）において利用可能である（たとえば格納されている）アップデートを示す例示的なインターフェイスの図である。インターフェイス７００から分かり得るように、アップデートの例は、アップデートの一部としてフィックスされた１つ以上のバグを表わすバグ番号であり得る。例示のために、インターフェイス７００に示されるアイテム番号１０はバグ番号２２０９３１９９に対応する。バグ番号２２０９３１９９の記載は「デザインフォームウィザードを使用する場合にＯｋボタンが表示されない」である。この例示では、バグ番号２２０９３１９９はアプリケーションにおけるエラーをフィックスするように構成される。すなわち、具体的には、デザインフォームウィザードに「Ｏｋ」ボタンが表示されないというエラーをフィックスするように構成される。この例において、現在実行されているインスタンスは、まだバグ番号２２０９３１９９をインストールしておらず、したがって、現在実行されているインスタンスはアプリケーションの古いバージョンである。しかしながら、バグ番号２２０９３１９９に対応するバグフィックスは、アプリケーションの新しいインスタンスにインストールされる。現在実行されているインスタンスのメタデータを新しいインスタンスのメタデータと比較することにより、バグ番号２２０９１９９は、現在実行されているインスタンスにまだ適用されていないバグ番号であると識別される。いくつかの実施形態では、中央サーバに格納された利用可能なアップデートのうちのいくつかまたはすべては、アプリケーションに関連付けられるリポジトリ（たとえばリポジトリ４６０）へ自動的にダウンロードされる。いくつかの実施形態では、インターフェイス７００は、ユーザがどのアップデートをアプリケーションの現在実行されているインスタンスに適用することを望むのかをユーザが選択することを可能にする。いくつかの実施形態では、現在実行されているインスタンスに関連付けられるアップデートは、ユーザから入力または命令を受け取る必要なく、現在実行されているインスタンスに自動的に適用される。これらの実施形態では、現在実行されているインスタンスは、常に最新のバージョンで実行されている。

【００６６】

図８は、アプリケーションの現在実行されているインスタンスを自動的にアップデートするために行われる例示的なステップを示す例示的なインターフェイス８００の図である。各ステップのステータスは個々にインターフェイス８００上に提示され得る。ステップ１は、アプリケーションの新しいインスタンスがデプロイされ得る新しい環境テンプレートを作成することを含み得る。ステップ２は、新しい環境テンプレートにおいて新しいインスタンスをデプロイすることを含み得る。ステップ３は、ソースデータベース（たとえばアップデートをインストールする新しいインスタンス）をターゲットデータベース（たとえばアップデートがインストールされていない現在実行されているインスタンス）と比較するツールである「変更アシスタント」をインストールすることを含む。ステップ４で

は、アップデート（たとえばソースコード内のバグフィックスを含み得るパッチセット）が、ソースデータベース（たとえばアップデートがインストールされている新しいインスタンス）に適用され得る。ステップ５では、アップデートデータパッケージ（たとえば、１つの「メイク・ミー・カレント（make me current）」変更パッケージ）が規定され得る。アップデートデータパッケージは、ソースデータベースとターゲットデータベースとの間で異なるオブジェクトの識別子を含み得る。たとえば、ソースデータベースとターゲットデータベースとの間で異なるオブジェクトの識別子は、ソースデータベースにインストールされたアップデートを表わし得る。いくつかの例において、ソースデータベースとターゲットデータベースとの間の差がひとたび識別されると、変更アシスタントは、アップデートによって変更されたオブジェクトのすべてを含むアップデートデータパッケージを生成し得る。オブジェクトがソースデータベースとターゲットデータベースとの間で異なる場合、そのオブジェクトは一意識別子によって表わされ得る。そのオブジェクトは、ひとたび識別されると、ファイルにダウンロードされ得、次いで、シリアライズされた形態でエクスポートされ得る。ステップ６では、アップデートデータパッケージが作成される（たとえば、ソースデータベースとターゲットデータベースとの間で変更されたと検出されたオブジェクトの識別子を含むデータパッキング）。ステップ７では、アップデートデータパッケージがターゲットデータベースに適用され得、ターゲットデータベースのファイルがアップデートされ得、アプリケーションの最新のバージョンにされ得る。

【００６７】

図９は、多層アプリケーションを自動的にアップデートするための例示的なプロセス９００を示すフローチャートである。上述したように、メジャーリリースは、規則的または不規則的ベースで周期的に利用可能になり得、マイナーリリースは、メジャーリリース同士間で周期的に利用可能になり得る。たとえば、メジャーリリース同士の間で、マイナーアップデートが、メジャーリリースで見つかったバグをフィックスするよう利用可能であり得る。ある実施形態は、最も最近のアップデートがメジャーリリースまたはマイナーアップデートであったか否かに関わらず、アプリケーションスタックが最小のダウンタイムおよび最小のネットワーク負荷で自動的にアップデートされることを可能にする。

【００６８】

処理フロー９００はブロック９１０において開始し、ブロック９１０では、アプリケーションが、アップデートが中央サーバに格納されているか否かを決定するために周期的に中央サーバにアクセスする。たとえば、アプリケーションは、中央サーバにおいて利用可能または格納されている任意の新しいアップデートについて、ランダムにまたは規則的に中央サーバにクエリ送信するように構成され得る。ブロック９２０では、アプリケーションは、アップデートが利用可能か否か決定し得る。アップデートが利用可能でない場合、アプリケーションは、別の時間に、アップデートについて中央サーバにクエリ送信し続ける。しかしながら、１つ以上のアップデートが中央サーバにおいて格納されている場合、当該１つ以上のアップデートは、ブロック９３０においてのように、アプリケーションに関連付けられるリポジトリへダウンロードされ得る。アップデートがひとたび中央サーバからリポジトリへダウンロードされると、アップデートは中央サーバから削除され得るか、または、ダウンロード済であるとしてフラグが立てられ得る。ブロック９４０では、アプリケーション環境は、クラウドにおいて自身の新しいインスタンスを提供し、当該新しいインスタンスに１つ以上のアップデートをインストールする。たとえば新しいインスタンスを提供することは、あらかじめ規定された構成セッティングまたは規定されたオンザフライの構成セッティング（たとえばＣＰＵおよびＲＡＭアロケーション）を有する新しいバーチャルマシンを生成することと、１つ以上のアップデートがインストールされた状態で新しいバーチャルマシン上でアプリケーションスタックを実行することとを含み得る。このとき、アプリケーション環境は、アプリケーションの現在実行されているインスタンス（たとえばユーザと現在インタラクションしているアプリケーションのインスタンス）と、ダウンロードされたアップデートがインストールされたアプリケーションの新しいインスタンスとの両方に関連付けられ得る。しかしながら、いくつかの実現例では、現在

実行されているインスタンスはプロダクション環境にあり、その一方、アプリケーションの新しいインスタンスはプロダクション環境にない。これらの実現例では、エンドユーザはアプリケーションの新しいインスタンスと通信できず、および／または、アプリケーションの新しいインスタンスは、新しいインスタンスとエンドユーザデバイスとの間のインタラクションに基づいてトランザクションを処理しない。代わりに、エンドユーザは現在実行されているインスタンスと通信し続ける。

【００６９】

ブロック９５０では、アプリケーションは、２つのインスタンス同士間に任意の差（たとえばデルタ）が存在するか否かを決定するために、現在実行されているインスタンスを新しいインスタンスと比較する。たとえば、上述したように、２つのインスタンスを比較することは、現在実行されているインスタンスのメタデータと、新しいインスタンスのメタデータとを比較することを含み得る。いくつかの実現例では、メタデータは、インスタンスのソースコードに適用されたことがあるかまたは以前に適用されたバグ番号を含み得る。新しいインスタンスのメタデータに含まれるバグ番号の集合が、現在のインスタンスのメタデータに含まれるバグ番号の集合と異なる場合、アプリケーションは、アップデートがバグ番号における差（たとえばデルタ）に対応すると決定し得る。たとえば、バグ番号１および３は以前に現在のインスタンスに適用されており、また、バグ番号１、３および５が以前に新しいインスタンスに適用された場合、バグ番号５のデルタは、現在のインスタンスをアプリケーションの最新バージョンにアップデートするために、現在のインスタンスに適用されるべきであると決定される。メタデータがバグ番号以外のデータを含んでもよいことが認識されるであろう。たとえば、修正されたソースコードを表わすファイルリファレンスオブジェクトの識別子および／またはファイルリファレンスオブジェクトがメタデータとして格納され得る。この例において、新しいインスタンスのメタデータを現在のインスタンスのメタデータと比較することによって、新しいインスタンスにインストールされたアップデートによって修正されたファイルリファレンスオブジェクトが識別され得る。ブロック９６０では、アプリケーションは、現在のインスタンスのメタデータと新しいインスタンスのメタデータとの間のデルタを含むアップデートデータパッケージを作成し得る。ブロック９７０では、現在のインスタンスは、自身にデータパッケージを適用し得、これにより、現在のインスタンスを実際上アップデートする。有利なことに、現在のインスタンスに対するアップデートでは、アプリケーションによって提供されるサービスの中断が最小となり、ネットワーク効率を改善する。

【００７０】

図１０は、実施形態のうちの１つを実現するための分散型システム１０００の簡略図を示す。例示される実施形態において、分散型システム１０００は、ネットワーク１０１０を介して、ウェブブラウザまたはプロプラエタリクライアント（たとえばOracle Forms）などといったクライアントアプリケーションを実行および動作するように構成される１つ以上のクライアントコンピューティングデバイス１００２、１００４、１００６および１００８を含む。サーバ１０１２は、ネットワーク１０１０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１００２、１００４、１００６および１００８と通信可能に結合され得る。

【００７１】

さまざまな実施形態では、サーバ１０１２は、システムのコンポーネントの１つ以上によって提供される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得る。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）モデルとして、クライアントコンピューティングデバイス１００２、１００４、１００６および／または１００８のユーザに提供され得る。クライアントコンピューティングデバイス１００２、１００４、１００６および／または１００８を動作するユーザは、これらのコンポーネントによって提供されるサービスを利用するためにサーバ１０１２とインタラクションするよう１つ以上のクライアントアプリケーションを利用する。

【００７２】
図において示される構成では、システム１０００のソフトウェアコンポーネント１０１８、１０２０および１０２２は、サーバ１０１２上で実現されるように示されている。他の実施形態では、システム１０００のコンポーネントの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も、クライアントコンピューティングデバイス１００２、１００４、１００６および／または１００８の１つ以上によって実現され得る。クライアントコンピューティングデバイスを動作するユーザは、これらのコンポーネントによって提供されるサービスを使用するために、１つ以上のクライアントアプリケーションを利用し得る。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その組み合わせで実現されてもよい。分散型システム１０００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認識されるべきである。したがって、図面において示される実施形態は、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って、限定的であるようには意図されない。
【００７３】
クライアントコンピューティングデバイス１００２、１００４、１００６および／または１００８は、ポータブルハンドヘルドデバイス（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、携帯情報端末（ＰＤＡ））、または、ウェアブルデバイス（たとえばＧｏｏｇｌｅ　Ｇｌａｓｓ（登録商標）ヘッドマウントディスプレイ）、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ　Ｍｏｂｉｌｅ（登録商標）のような実行されているソフトウェア、ならびに／または、ｉＯＳおよびＷｉｎｄｏｗｓ　Ｐｈｏｎｅ、Ａｎｄｒｏｉｄ、ブラックベリー１０、Ｐａｌｍ　ＯＳのようなさまざまなモバイルオペレーティングシステムなどであり得、インターネット、Ｅメール、ショートメッセージサービス（ＳＭＳ：short message service）、ブラックベリー（登録商標）、または、他の通信プロトコルが有効化されている。クライアントコンピューティングデバイスは、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を含む汎用パーソナル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、たとえばＧｏｏｇｌｅ　Ｃｈｒｏｍｅ　ＯＳといった様々なＧＮＵ／Ｌｉｎｕｘオペレーティングシステムを含むがそれに限定されないさまざまな商業的に利用可能なＵＮＩＸ（登録商標）またはＵＮＩＸライクの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。代替的または付加的には、クライアントコンピューティングデバイス１００２、１００４、１００６および１００８は、シンクライアントコンピュータ、インターネット有効化ゲームシステム（たとえばＫｉｎｅｃｔ（登録商標）ジェスチャ入力デバイスを有するかまたは有していないＭｉｃｒｏｓｏｆｔ　Ｘｂｏｘゲームコンソール）、および／または、パーソナルメッセージングデバイスといった、ネットワーク１０１０を介して通信することができる任意の他の電子デバイスであり得る。
【００７４】
例示的な分散型システム１０００は４つのクライアントコンピューティングデバイスを有しているのが示されているが、任意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得る。センサなどを有するデバイスのような他のデバイスがサーバ１０１２とインタラクションしてもよい。
【００７５】
分散型システム１０００におけるネットワーク１０１０は、ＴＣＰ／ＩＰ（transmission control protocol/Internet protocol）、ＳＮＡ（systems network architecture）、ＩＰＸ（Internet packet exchange）、およびＡｐｐｌｅＴａｌｋなどを含むがこれらに限定されないさまざまな商業的に利用可能なネットワークプロトコルのいずれかを用いるデータ通信をサポートし得る、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り得る。単に例として、ネットワーク１０１０はたとえば、イーサネット（登録商

標）および／またはトークンリングなどに基づいたローカルエリアネットワーク（ＬＡＮ）であり得る。ネットワーク１０１０は、ワイドエリアネットワークおよびインターネットであり得る。当該ネットワークは、バーチャルネットワークを含み得、当該バーチャルネットワークは、バーチャル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、ワイヤレスネットワーク（たとえばInstitute of Electrical and Electronics（ＩＥＥＥ）８０２．１１のプロトコルスイートに従って動作するネットワーク、ブルートゥース（登録商標）、もしくは、任意の他のワイヤレスプロトコル）、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がこれらに限定されない。

【００７６】

サーバ１０１２は、１つ以上の汎用コンピュータ、特殊サーバコンピュータ（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、任意の他の適切な構成および／もしくは組み合わせから構成され得る。さまざまな実施形態では、サーバ１０１２は、上記開示に記載される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得る。たとえば、サーバ１０１２は、本開示の実施形態に従った、上記の処理を実行するためのサーバに対応し得る。

【００７７】

サーバ１０１２は、上で論じた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と、任意の商業的に利用可能なサーバオペレーティングシステムとを実行し得る。サーバ１０１２はさらに、さまざまな付加的な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いずれかを実行し得、ＨＴＴＰ（ハイパーテキストトランスポートプロトコル）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）サーバおよびデータベースサーバなどを含む。例示的なデータベースサーバはＯｒａｃｌｅ、Ｍｉｃｒｏｓｏｆｔ、Ｓｙｂａｓｅ、およびＩＢＭ（International Business Machines）などから商業的に利用可能なものを含むがこれらに限定されない。

【００７８】

いくつかの実現例では、サーバ１０１２は、クライアントコンピューティングデバイス１００２、１００４、１００６および１００８のユーザから受け取られるデータフィードおよび／またはイベントアップデートを分析および統合するために１つ以上のアプリケーションを含み得る。例として、データフィードおよび／またはイベントアップデートは、Ｔｗｉｔｔｅｒ（登録商標）フィード、Ｆａｃｅｂｏｏｋ（登録商標）アップデート、または、１つ以上のサードパーティ情報源および連続的なデータストリームから受け取られるリアルタイムアップデートを含み得るがこれらに限定されない。当該リアルタイムアップデートは、センサデータアプリケーションに関係するリアルタイムイベント、金融ティッカ、ネットワークパフォーマンス測定ツール（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、および自動車交通モニタリングなどを含み得る。サーバ１０１２はさらに、クライアントコンピューティングデバイス１００２、１００４、１００６および１００８の１つ以上のディスプレイデバイスを介してデータフィードおよび／またはリアルタイムイベントを表示するために１つ以上のアプリケーションを含み得る。

【００７９】

分散型システム１０００はさらに、１つ以上のデータベース１０１４および１０１６を含み得る。データベース１０１４および１０１６は、さまざまな位置に存在し得る。例として、データベース１０１４および１０１６の１つ以上は、サーバ１０１２に対してローカルである（および／またはサーバ１０１２中に存在する）一時的でない記憶媒体上で存在し得る。代替的には、データベース１０１４および１０１６はサーバ１０１２からリモートであり得、ネットワークベースの接続または専用接続を介してサーバ１０１２と通信

し得る。実施形態のうちの１つの集合では、データベース１０１４および１０１６はストレージエリアネットワーク（ＳＡＮ： storage-area network）に存在し得る。同様に、サーバ１０１２に起因する機能を実行するための任意の必要なファイルが適切なように、サーバ１０１２上にローカルにおよび／またはリモートに格納され得る。実施形態のうちの１つの集合では、データベース１０１４および１０１６は、Ｏｒａｃｌｅによって提供されるデータベースのような、ＳＱＬフォーマットのコマンドに応答してデータを格納、更新、および抽出するよう適合されるリレーショナルデータベースを含み得る。

【００８０】

図１１は、本開示の実施形態に従った、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得るシステム環境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。例示される実施形態では、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とインタラクションするためにユーザによって使用され得る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、ウェブブラウザ、プロプラエタリクライアントアプリケーション（たとえばOracle Forms）またはいくつかの他のアプリケーションといった、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使用するためにクラウドインフラストラクチャシステム１１０２とインタラクションす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使用され得るクライアントアプリケーションを動作するように構成され得る。

【００８１】

図において示されるクラウドインフラストラクチャシステム１１０２は、示されたもの以外のコンポーネントを有し得るということが認識されるべきである。さらに、図において示される実施形態は、本発明の実施形態を組み込み得るクラウドインフラストラクチャシステムの単なる一例である。他のいくつかの実施形態では、クラウドインフラストラクチャシステム１１０２は、図において示されるよりも多いまたは少ないコンポーネントを有し得るか、２つ以上のコンポーネントを組み合わせ得るか、または、コンポーネントの異なる構成または配置を有し得る。

【００８２】

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、１００２、１００４、１００６および１００８について上で記載されたものと同様のデバイスであり得る。

【００８３】

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を有するように示されているが、任意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得る。センサなどを有するデバイス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とインタラクションし得る。

【００８４】

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との間のデータの通信および交換を促進し得る。各ネットワークは、ネットワーク１０１０について上で記載されたものを含む、さまざまな商業的に利用可能なプロトコルのいずれかを用いるデータ通信をサポートし得る、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り得る。

【００８５】

クラウドインフラストラクチャシステム１１０２は、サーバ１０１２について上で記載されたものを含み得る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。

【００８６】

ある実施形態では、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、ウェブベースのＥメールサービス、ホストされた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、および、管理された技術サポートサービスなどといった、クラウドインフラストラクチャシステムのユーザにオンデマンドで利用可能になるサービスのホストを含み得る。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サービスは、そのユーザのニーズを満たすために動的にスケールされ得る。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サービスの特定のインスタンス化は、本願明細書において「サービスインスタンス」と称される。一般に、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なる任意のサービスは、「クラウドサービス」と称される。典型的に、パブリッククラウド環境では、クラウドサービスプロバイダのシステムを構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がアプリケーションをホストし得、ユーザが、インターネット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し使用し得る。

【００８７】

いくつかの例において、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、ホストされたデータベース、ホストされたウェブサーバ、ソフトウェアアプリケーション、もしくは、クラウドベンダーによってユーザに提供される他のサービス、または、当該技術において別の態様で知られるものへの保護されたコンピュータネットワークアクセスを含み得る。たとえば、サービスは、インターネットを通じてクラウド上のリモートストレージへのパスワードで保護されたアクセスを含み得る。別の例として、サービスは、ネットワーク接続されたデベロッパによるプライベートの使用のためのウェブサービスベースのホスト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み得る。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上にホストされたＥメールソフトウェアアプリケーションへのアクセスを含み得る。

【００８８】

ある実施形態では、クラウドインフラストラクチャシステム１１０２は、セルフサービスであり、サブスクリプションベースであり、弾力的にスケーラブルであり、信頼性があり、高可用性であり、かつ、セキュアな態様で顧客に送達されるアプリケーションスイート、ミドルウェアおよびデータベースサービスオファリングを含み得る。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、本願の譲受人によって提供されるOracle Public Cloudである。

【００８９】

さまざまな実施形態では、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の顧客のサブスクリプションを自動的に提供、管理およびトラッキングするように適合され得る。クラウドインフラストラクチャシステム１１０２は、異なるデプロイメントモデルを介してクラウドサービスを提供し得る。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がクラウドサービスを販売する組織によって所有され（たとえば、Ｏｒａｃｌｅによって所有され）、サービスが一般大衆または異なる企業に利用可能になるパブリッククラウドモデルに従って提供され得る。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が単一の組織のためにのみ動作され、当該組織内の１つ以上のエンティティにサービスを提供し得るプライベートクラウドモデルに従って提供され得る。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、関連す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に従って提供され得る。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み合わせであるハイブリッドクラウドモデルに従って提供され得る。

【００９０】

いくつかの実施形態では、クラウドインフラストラクチャシステム８０２によって提供されるサービスは、ソフトウェア・アズ・ア・サービス（ＳａａＳ）カテゴリ、プラット

ホーム・アズ・ア・サービス（ＰａａＳ： Platform as a Service）カテゴリ、インフラストラクチャ・アズ・ア・サービス（ＩａａＳ： Infrastructure as a Service）カテゴリに従って提供される１つ以上のサービス、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み得る。顧客は、サブスクリプションオーダを介して、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し得る。クラウドインフラストラクチャシステム１１０２は、顧客のサブスクリプションオーダにおいてサービスを提供するために処理を実行する。

【００９１】

いくつかの実施形態では、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は、アプリケーションサービス、プラットホ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み得るがこれらに限定されない。いくつかの例において、アプリケーションサービスはＳａａＳプラットホームを介し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得る。ＳａａＳプラットホームは、ＳａａＳカテゴリに該当するクラウドサービスを提供するように構成され得る。たとえば、ＳａａＳプラットホームは、統合開発およびデプロイメントプラットホーム上でオンデマンドアプリケーションスイートを構築し送達する能力を提供し得る。ＳａａＳプラットホームは、ＳａａＳサービスを提供するために存在する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得る。ＳａａＳプラットホ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上で実行されるアプリケーションを利用し得る。顧客は、別個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取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得る。その例は、販売実績管理、エンタープライズインデグレーション、および、大きな組織のためのビジネスフレキシビリティのためのソリューションを提供するサービスを含むがこれに限定されない。

【００９２】

いくつかの実施形態では、プラットホームサービスは、ＰａａＳプラットホームを介し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得る。ＰａａＳプラットホームは、ＰａａＳカテゴリに該当するクラウドサービスを提供するように構成され得る。プラットホームサービスの例は、（Ｏｒａｃｌｅのような）組織がシェアされた共通のアーキテクチャ上で既存のアプリケーションを統合することを可能にするサービスと、プラットホームによって提供されるシェアされたサービスを強化する新しいアプリケーションを構築する能力とを含み得るがこれらに限定されない。ＰａａＳプラットホームは、ＰａａＳサービスを提供するために存在する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得る。顧客は、別個のライセンスおよびサポートを購入する必要なく、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サービスを取得することができる。プラットホームサービスの例は、Oracle Java Cloud Service（ＪＣＳ）およびOracle Database Cloud Service（ＤＢＣＳ）などを含み得るがこれらに限定されない。

【００９３】

ＰａａＳプラットホ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できる。いくつかの実施形態では、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プラットホームサービスは、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 Middleware services）およびＪａｖａクラウドサービスを含み得る。一実施形態では、データベースクラウドサービスは、組織がデータベースリソースをプールしデータベースクラウドの形態でデータベース・アズ・ア・サービス（Database as a Service）を顧客に提供することを可能にするシェアされたサービスデプロイメントモデルをサポートし得る。ミドルウェアクラウドサービスは、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するための顧客のためのプラットホームを提供し得、Ｊａｖａクラウドサービスはクラウドインフラストラクチャシステムにおいて、Ｊａｖａアプリケーシ

ョンをデプロイするための顧客のためのプラットホームを提供し得る。
【００９４】
さまざまな異なるインフラストラクチャサービスは、クラウドインフラストラクチャシステムにおいてＩａａＳプラットホームによって提供され得る。インフラストラクチャサービスは、ＳａａＳプラットホームおよびＰａａＳプラットホームによって提供されるサービスを顧客が利用するためのストレージ、ネットワークおよび他の基礎的なコンピューティングリソースといった存在するコンピューティングリソースの管理および制御を促進する。
【００９５】
ある実施形態では、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様々なサービスを提供するために使用されるリソースを提供するためのインフラストラクチャリソース１１３０を含み得る。一実施形態では、インフラストラクチャリソース１１３０は、ＰａａＳプラットホームおよびＳａａＳプラットホームによって提供されるサービスを実行するために、サーバ、ストレージおよびネットワークリソースといったハードウェアのあらかじめ統合され最適化された組み合わせを含み得る。
【００９６】
いくつかの実施形態では、クラウドインフラストラクチャシステム１１０２におけるリソースは、複数のユーザによってシェアされ、要求に応じて動的に再度割り当てられ得る。さらに、リソースは異なる時間ゾーンにおいてユーザに割り当てられ得る。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ユーザの第１の集合が、規定された数の時間の間、クラウドインフラストラクチャシステムのリソースを利用することを可能にし得、その後、異なる時間ゾーンに位置するユーザの別の集合への同じリソースの再割り当てを可能にし得、これによりリソースの利用を最大化する。
【００９７】
ある実施形態では、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と、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とによってシェアされる多くの内部シェアサービス１１３２が提供され得る。内部シェアサービスは、セキュリティおよびアイデンティティサービス、インテグレーションサービス、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、ウィルススキャニング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、および、ファイル転送サービスなどを含み得るがこれらに限定されない。
【００９８】
ある実施形態では、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得る。一実施形態では、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によって受け取られる顧客のサブスクリプションなどを提供、管理およびトラッキングするための能力を含み得る。
【００９９】
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、オーダ管理モジュール１１２０、オーダ調整モジュール１１２２、オーダ提供モジュール１１２４、オーダ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といった１つ以上のモジュールによって提供され得る。これらのモジュールは、汎用コンピュータ、特殊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もしくは組み合わせであり得る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か、または、当該１つ以上のコンピュータおよび／またはサーバを使用して提供され得る。
【０１００】

例示的な動作１１３４では、クライアントデバイス１１０４、１１０６または１１０８といったクライアントデバイスを使用する顧客は、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１つ以上のサービスを要求し、クラウドインフラストラクチャシステム１１０２によってオファーされる１つ以上のサービスについてのサブスクリプションのオーダ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とインタラクションし得る。ある実施形態では、顧客は、クラウドユーザインターフェイス（ＵＩ）、クラウドＵＩ１１１２、クラウドＵＩ１１１４および／またはクラウドＵＩ１１１６にアクセスし得、これらのＵＩを介してサブスクリプションオーダを行い得る。顧客がオーダを行った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８０２によって受け取られるオーダ情報は、顧客と、顧客がサブスクライブする意図がある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つ以上のサービスとを識別する情報を含み得る。

【０１０１】

オーダが顧客によって行われた後、オーダ情報は、クラウドＵＩ１１１２、１１１４および／または１１１６を介して受け取られる。

【０１０２】

動作１１３６では、オーダはオーダデータベース１１１８に格納される。オーダデータベース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とともに他のシステム要素と協働して動作されるいくつかのデータベースのうちの１つであり得る。

【０１０３】

動作１１３８において、オーダ情報は、オーダ管理モジュール１１２０に転送される。いくつかの場合では、オーダ管理モジュール１１２０は、オーダに関係する、オーダの照合およびオーダの照合の際のオーダの記帳といった請求および会計機能を実行するように構成され得る。

【０１０４】

動作１１４０において、オーダに関する情報がオーダ調整モジュール１１２２に通信される。オーダ調整モジュール１１２２は、顧客によって行われたオーダについてサービスおよびリソースの提供を調整するためにオーダ情報を利用し得る。いくつかの場合、オーダ調整モジュール１１２２は、オーダ提供モジュール１１２４のサービスを使用して、サブスクライブされたサービスをサポートするために、リソースの提供を調整し得る。

【０１０５】

ある実施形態において、オーダ調整モジュール１１２２は、各オーダ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、オーダが提供に進むべきか否か決定する。動作１１４２において、新しいサブスクリプションについてのオーダを受け取ると、オーダ調整モジュール１１２２は、リソースを割り当てるとともに、サブスクリプションオーダを履行するために必要とされるそれらのリソースを構成するよう、オーダ提供モジュール１１２４に要求を送信する。オーダ提供モジュール１１２４は、顧客によってオーダされたサービスのためのリソースの割り当てを可能にする。オーダ提供モジュール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リソースを提供するために使用される物理的なインプリメンテーション層との間の抽象化レベルを提供する。したがって、オーダ調整モジュール１１２２は、サービスおよびリソースが実際にオンザフライで提供されるか、または、あらかじめ提供され要求の際にのみ割り当てられ得る／アサインされ得るか否かといったインプリメンテーションの詳細から分離され得る。

【０１０６】

動作１１４４において、サービスおよびリソースがひとたび提供されると、提供されたサービスの通知が、クラウドインフラストラクチャシステム１１０２のオーダ提供モジュール１１２４によって、クライアントデバイス１１０４、１１０６および／または１１０８上の顧客に送信され得る。

【０１０７】
動作１１４６において、顧客のサブスクリプションオーダは、オーダ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得る。いくつかの場合では、オーダ管理およびモニタリングモジュール１１２６は、使用されたストレージの量、転送されたデータ量、ユーザの数、および、システムアップタイムおよびシステムダウンタイムの量といった、サブスクリプションオーダにおけるサービスについての使用統計を収集するように構成され得る。
【０１０８】
ある実施形態では、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を含み得る。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においてアクセス管理および認証サービスのようなアイデンティティサービスを提供するように構成され得る。いくつかの実施形態では、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関する情報を制御し得る。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と、それらの顧客がさまざまなシステムリソース（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ことが承認されているかを記述する情報とを含み得る。アイデンティティ管理モジュール１１２８はさらに、各顧客に関する記述情報と、その記述情報がどのようにおよび誰にアクセスおよび修正され得るのかに関する記述情報との管理を含み得る。
【０１０９】
図１２は、本発明のさまざまな実施形態が実現され得る例示的なコンピュータシステム１２００を示す。システム１２００は、上記のコンピュータシステムのうちのいずれかを実現するために使用され得る。図に示されるように、コンピュータシステム１２００は、バスサブシステム１２０２を介して多くの周辺サブシステムと通信する処理ユニット１２０４を含む。これらの周辺サブシステムは、処理アクセレレーションユニット１２０６、Ｉ／Ｏサブシステム１２０８、ストレージサブシステム１２１８および通信サブシステム１２２４を含み得る。ストレージサブシステム１２１８は、有形的なコンピュータ読取可能記憶媒体１２２２およびシステムメモリ１２１０を含む。
【０１１０】
バスサブシステム１２０２は、コンピュータシステム１２００のさまざまなコンポーネントおよびサブシステムを意図されるように互いと通信させるためのメカニズムを提供する。バスサブシステム１２０２は単一のバスとして概略的に示されるが、バスサブシステムの代替的な実施形態は複数のバスを利用してもよい。バスサブシステム１２０２は、さまざまなバスアーキテクチャのいずれかを使用す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いずれかであり得る。たとえば、そのようなアーキテクチャは、産業標準アーキテクチャ（ＩＳＡ： Industry Standard Architecture）バスと、マイクロチャネルアーキテクチャ（ＭＣＡ： Micro Channel Architecture）バスと、エンハンスドＩＳＡ（ＥＩＳＡ： Enhanced ISA）バスと、ビデオエレクトロニクス標準組織（ＶＥＳＡ：Video Electronics Standards Association）ローカルバスと、ＩＥＥＥＰ１３８６．１規準に従って製造されるメザニンバスとして実現され得るペリフェラルコンポーネントインターコネクト（ＰＣＩ）バスとを含み得る。
【０１１１】
１つ以上の集積回路（たとえば従来のマイクロプロセッサまたはマイクロコントローラ）として実現され得る処理ユニット１２０４は、コンピュータシステム１２００の動作を制御する。１つ以上のプロセッサが処理ユニット１２０４に含まれ得る。これらのプロセッサは、シングルコアまたはマルチコアプロセッサを含み得る。ある実施形態では、処理ユニット１２０４は、１つ以上の独立した処理ユニット１２３２および／または１２３４として実現され得、シングルまたはマルチコアプロセッサが各処理ユニットに含まれてい

る。他の実施形態では、処理ユニット１２０４はさらに、２つのデュアルコアプロセッサをシングルチップへと統合することにより形成されるクワッドコア処理ユニットとして実現され得る。

【０１１２】

さまざまな実施形態では、処理ユニット１２０４は、プログラムコードに応答してさまざまなプログラムを実行し得、複数の同時に実行しているプログラムまたはプロセスを維持し得る。任意の所与の時間において、実行されるべきプログラムコードのうちのいくつかまたはすべてがプロセッサ１２０４および／またはストレージサブシステム１２１８に存在し得る。好適なプログラミングを通じて、プロセッサ１２０４は、上に記載されたさまざまな機能を提供し得る。コンピュータシステム１２００はさらに、デジタル信号プロセッサ（ＤＳＰ）および／または専用プロセッサなどを含み得る処理アクセレレーションユニット１２０６を含み得る。

【０１１３】

Ｉ／Ｏサブシステム１２０８は、ユーザインターフェイス入力デバイスおよびユーザインターフェイス出力デバイスを含み得る。ユーザインターフェイス入力デバイスは、キーボードと、マウスまたはトラックボールのようなポインティングデバイスと、ディスプレイに組み込まれるタッチパッドまたはタッチスクリーンと、スクロールホイールと、クリックホイールと、ダイヤルと、ボタンと、スイッチと、キーパッドと、音声コマンド認識システム、マイクロフォンおよび他のタイプの入力デバイスを有する音声入力デバイスとを含み得る。ユーザインターフェイス入力デバイスはたとえば、Ｍｉｃｒｏｓｏｆｔ Ｋｉｎｅｃｔ（登録商標）モーションセンサのようなモーション感知および／またはジェスチャ認識デバイスを含み得る。当該モーション感知および／またはジェスチャ認識デバイスは、ジェスチャおよび音声コマンドを使用するナチュラルユーザインターフェイスを通じて、ユーザがＭｉｃｒｏｓｏｆｔ Ｘｂｏｘ（登録商標）３６０ゲームコントローラのような入力デバイスを制御し、当該入力デバイスとインタラクションすることを可能にする。ユーザインターフェイス入力デバイスはさらに、Ｇｏｏｇｌｅ Ｇｌａｓｓ（登録商標）の瞬き検出器のような、ユーザからの目のアクティビティ（たとえば写真を撮る間および／またはメニュー選択を行う間の「瞬き」）を検出し、この目の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、アイジェスチャ（eye gesture）認識デバイスを含み得る。さらに、ユーザインターフェイス入力デバイスは、音声コマンドを通じてユーザが音声認識システム（たとえばＳｉｒｉ（登録商標）ナビゲータ）とインタラクションすることを可能にする音声認識感知デバイスを含み得る。

【０１１４】

ユーザインターフェイス入力デバイスはさらに、３次元（３Ｄ）マウスと、ジョイスティックまたはポインティングスティックと、ゲームパッドおよびグラフィックタブレットと、スピーカのようなオーディオ／ビジュアルデバイスと、デジタルカメラと、デジタルビデオカメラと、ポータブルメディアプレイヤーと、ウェブカメラと、イメージスキャナと、フィンガープリントスキャナと、バーコードリーダ３Ｄスキャナと、３Ｄプリンタと、レーザ距離計と、視線トラッキングデバイスとを含むがこれらに限定されない。さらに、ユーザインターフェイス入力デバイスはたとえば、コンピュータトモグラフィ、磁気共鳴イメージング、ポジションエミッショントモグラフィ、医療用超音波検査法デバイスのような医療用画像入力デバイスを含み得る。ユーザインターフェイス入力デバイスはさらに、たとえば、ＭＩＤＩキーボードおよびデジタル楽器などのようなオーディオ入力デバイスを含み得る。

【０１１５】

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、または、オーディオ出力デバイスなどのノンビジュアルディスプレイを含み得る。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用いるもののようなフラットパネルデバイス、投射デバイスおよびタッチスクリーンな

どであり得る。一般に、「出力デバイス」という用語の使用は、コンピュータシステム１２００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すべての可能なタイプのデバイスおよびメカニズムを含むように意図される。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といった、視覚的にテキスト、グラフィックスおよびオーディオ／ビデオ情報を伝達するさまざまなディスプレイデバイスを含み得るがこれらに限定されない。

【０１１６】

コンピュータシステム１２００は、システムメモリ１２１０内に現在位置しているのが示されるソフトウェア要素を含むストレージサブシステム１２１８を含み得る。システムメモリ１２１０は、プロセスユニット１２０４上にロード可能および実行可能なプログラム命令と、これらのプログラムの実行中に生成されるデータとを格納し得る。

【０１１７】

コンピュータシステム１２００の構成およびタイプに依存して、システムメモリ１２１０は、揮発性（たとえばランダムアクセスメモリ（ＲＡＭ））であり得、および／または、不揮発性（たとえばリードオンリメモリ（ＲＯＭ）、フラッシュメモリなど）であり得る。ＲＡＭは典型的に、処理ユニット１２０４へすぐにアクセス可能であり、および／または、処理ユニット１２０４によって現在動作および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１２１０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）といった複数の異なるタイプのメモリを含み得る。いくつかの実現例では、たとえば起動中にコンピュータシステム１２００内の要素同士間で情報を転送するのを支援する基本ルーチンを含む基本入出力システム（ＢＩＯＳ）が典型的にＲＯＭに格納され得る。限定ではなく例として、システムメモリ１２１０はさらに、クライアントアプリケーション、ウェブブラウザ、ミッドティアアプリケーション、リレーショナルデータベース管理システム（ＲＤＢＭＳ）などを含み得るアプリケーションプログラム１２１２と、プログラムデータ１２１４と、オペレーティングシステム１２１６とを示す。例として、オペレーティングシステム１２１６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のさまざまなバージョン、さまざまな商業的に利用可能なＵＮＩＸ（登録商標）あるいはＵＮＩＸライクのオペレーティングシステム（様々なＧＮＵ／ＬｉｎｕｘオペレーティングシステムおよびＧｏｏｇｌｅ Ｃｈｒｏｍｅ（登録商標）ＯＳなどを含むがこ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のようなモバイルオペレーティングシステムを含み得る。

【０１１８】

ストレージサブシステム１２１８はさらに、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造を格納するための有形的コンピュータ読取可能記憶媒体を提供し得る。プロセッサによって実行されると、上で記載された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１２１８に格納され得る。これらのソフトウェアモジュールまたは命令は、処理ユニット１２０４によって実行され得る。ストレージサブシステム８１１２は、さらに本発明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得る。

【０１１９】

ストレージサブシステム１２００はさらに、コンピュータ読取可能記憶媒体１２２２にさらに接続され得るコンピュータ読取可能記憶媒体リーダ１２２０を含み得る。それとともにおよび随意に、システムメモリ１２１０と組み合わせて、コンピュータ読取可能記憶媒体１２２２は、リモートストレージデバイス、ローカルストレージデバイス、固定ストレージデバイスおよび／またはリムーバブルストレージデバイスと、一時的におよび／ま

たはより永続的にコンピュータ読取可能情報を含有、格納、送信、および抽出するための記憶媒体とを包括的に表す。

【０１２０】

コードまたはコードの部分を含むコンピュータ読取可能記憶媒体１２２２はさらに、当該技術において公知または使用される、記憶媒体および通信媒体を含む任意の適切な媒体を含み得、たとえば、情報の格納および／または送信のための任意の方法もしくは技術で実現される揮発性および不揮発性、リムーバブルならびに非リムーバブル媒体であるがこれらに限定されない。これは、ＲＡＭ、ＲＯＭ、電子的に消去可能なプログラマブルＲＯＭ（ＥＥＰＲＯＭ： electronically erasable programmable ROM）、フラッシュメモリもしくは他のメモリ技術、ＣＤ－ＲＯＭ、デジタルバーサタイルディスク（ＤＶＤ）もしくは他の光学ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージもしくは他の磁気ストレージデバイス、または、他の有形的コンピュータ読取可能媒体といった、有形的なコンピュータ読取可能記憶媒体を含み得る。これはさらに、データ信号、データ送信または任意の他の媒体といった、所望の情報を送信するために使用され得るとともにコンピューティングシステム１２００によってアクセスされ得る無形的なコンピュータ読取可能媒体を含み得る。

【０１２１】

例として、コンピュータ読取可能記憶媒体１２２２は、非リムーバブル不揮発性磁気媒体からの読み出しおよび非リムーバブルの不揮発性磁気媒体への書き込みを行うハードディスクドライブと、リムーバブル不揮発性磁気ディスクからの読み出しおよびリムーバブル不揮発性磁気ディスクへの書き込みを行う磁気ディスクドライブと、ＣＤ－ＲＯＭ、ＤＶＤおよびブルーレイ（登録商標）ディスクまたは他の光学媒体といったリムーバブル不揮発性光ディスクからの読み出しおよびリムーバブル不揮発性光ディスクへの書き込みを行う光ディスクドライブとを含み得る。コンピュータ読取可能記憶媒体１２２２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、および、デジタルビデオテープなどを含み得るがこれらに限定されない。コンピュータ読取可能記憶媒体１２２２はさらに、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およびソリッドステートＲＯＭなどといった不揮発性メモリに基づいたソリッドステートドライブ（ＳＳＤ）と、ソリッドステートＲＡＭ、ダイナミックＲＡＭ、スタティックＲＡＭといった揮発性メモリに基づいたＳＳＤと、ＤＲＡＭベースのＳＳＤと、磁気抵抗ＲＡＭ（ＭＲＡＭ）ＳＳＤと、ＤＲＡＭおよびフラッシュメモリベースのＳＳＤの組み合わせを使用するハイブリッドＳＳＤとを含み得る。ディスクドライブおよびそれらの関連するコンピュータ読取可能媒体は、コンピュータシステム１２００のためのコンピュータ読取可能命令、データ構造、プログラムモジュールおよび他のデータの不揮発性ストレージを提供し得る。

【０１２２】

通信サブシステム１２２４は、他のコンピュータシステムおよびネットワークへのインターフェイスを提供する。通信サブシステム１２２４は、コンピュータシステム１２００からの他のシステムからデータを受け取るとともに当該他のシステムにデータを送信するためのインターフェイスとして機能する。たとえば、通信サブシステム９２４は、コンピュータシステム１２００がインターネットを介して１つ以上のデバイスに接続することを可能にし得る。いくつかの実施形態では、通信サブシステム１２２４は、（たとえば携帯電話技術、３Ｇ、４ＧもしくはＥＤＧＥ（enhanced data rates for global evolution）のような高度なデータネットワーク技術、ＷｉＦｉ（ＩＥＥＥ１２０２．１１ファミリスタンダード）、もしくは、他の移動通信技術、または、その任意の組み合わせを使用する）ワイヤレス音声および／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）トランシーバコンポーネント、グローバルポジショニングシステム（ＧＰＳ）レシーバコンポーネント、ならびに／または他のコンポーネントを含み得る。いくつかの実施形態では、通信サブシステム１２２４は、ワイヤレスインターフェイスに加えてまたは

そのワイヤレスインターフェイスの代わりに、有線ネットワーク接続（たとえばイーサネット）を提供し得る。

【０１２３】

いくつかの実施形態では、通信サブシステム１２２４はさらに、コンピュータシステム１２００を使用する１人以上のユーザに代わりに、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１２２６、イベントストリーム１２２８ならびにイベントアップデート１２３０などの形態で入力通信を受け取り得る。

【０１２４】

例として、通信サブシステム１２２４は、Ｔｗｉｔｔｅｒ（登録商標）フィードおよびＦａｃｅｂｏｏｋ（登録商標）アップデート、リッチサイトサマリ（ＲＳＳ： Rich Site Summary）フィードのようなウェブフィードといったソーシャルネットワークおよび／もしくは他の通信サービスのユーザからデータフィード１２２６、ならびに／または、１つ以上のサードパーティ情報源からのリアルタイムアップデートをリアルタイムで受け取るように構成され得る。

【０１２５】

さらに、通信サブシステム１２２４は、連続データストリームの形態でデータを受け取るように構成され得る。連続データストリームは、リアルタイムイベントのイベントストリーム１２２８および／またはイベントアップデート１２３０を含み得、本質的に明確な終わりを有さない連続的または無境界であり得る。連続データを生成するアプリケーションの例は、たとえば、センサデータアプリケーション、金融ティッカ、ネットワークパフォーマンス測定ツール（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、および、自動車交通モニタリングなどを含み得る。

【０１２６】

通信サブシステム１２２４はさらに、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１２２６、イベントストリーム１２２８、イベントアップデート１２３０などを、コンピュータシステム１２００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に構成され得る。

【０１２７】

コンピュータシステム１２００は、ハンドヘルドポータブルデバイス（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）ヘッドマウントディスプレイ）、ＰＣ、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、任意の他のデータ処理システムを含むさまざまなタイプのコンピュータシステムのうちの１つであり得る。

【０１２８】

コンピュータおよびネットワークの絶えず変化する性質により、図に示されるコンピュータシステム１２００の説明は特定の例としてのみ意図される。図示されたシステムよりも多くまたはより少ないコンポーネントを有する多くの構成が可能である。たとえば、カスタマイズされたハードウェアも使用される場合もあり、および／または、特定の要素は、ハードウェア、ファームウェア、ソフトウェア（アプレットを含む）、または、組み合わせで実現される場合もある。さらに、ネットワーク入出力デバイスのような他のコンピューティングデバイスへの接続も採用され得る。当業者であれば、本願明細書において提供される開示および教示に基づいて、さまざまな実施形態を実現する他の態様および／または方法を理解するであろう。

【０１２９】

上記明細書において、特定の実施形態を参照して本発明の局面が記載されているが、当業者であれば、本発明が特定の実施形態に限定されないことを認識するであろう。上記発明のさまざまな特徴および局面は、個々にまたは共同で使用されてもよい。さらに、実施形態は、明細書のより広い精神および範囲から逸脱することがなければ、本願明細書にお

いて記載されたものを越えた任意数の環境およびアプリケーションにおいて利用され得る。したがって、明細書および図面は、限定的ではなく例示的であるとみなされるべきである。

【図面】

【図１Ａ】

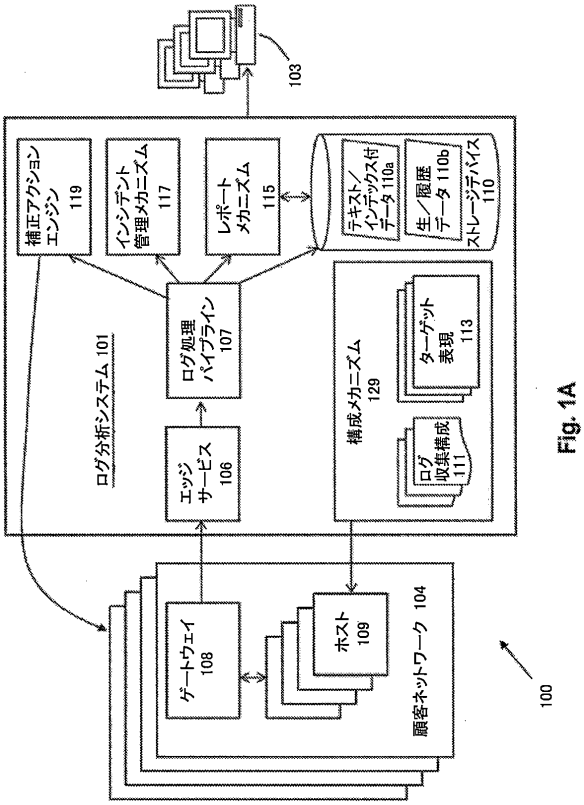


Fig. 1A

【図１Ｂ】

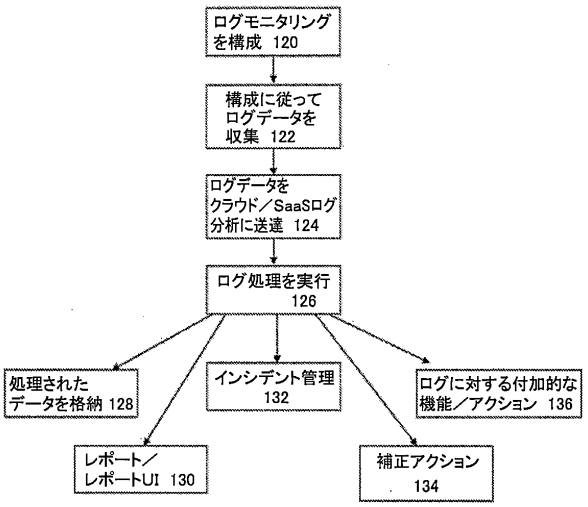


Fig. 1B

【図２】

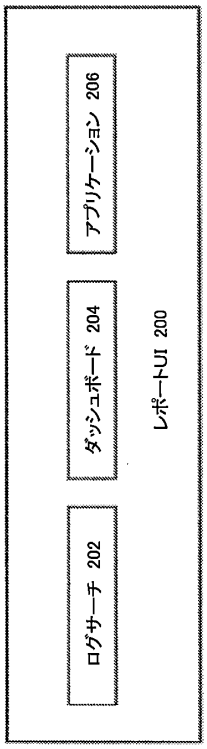


Fig. 2

【図３Ａ】

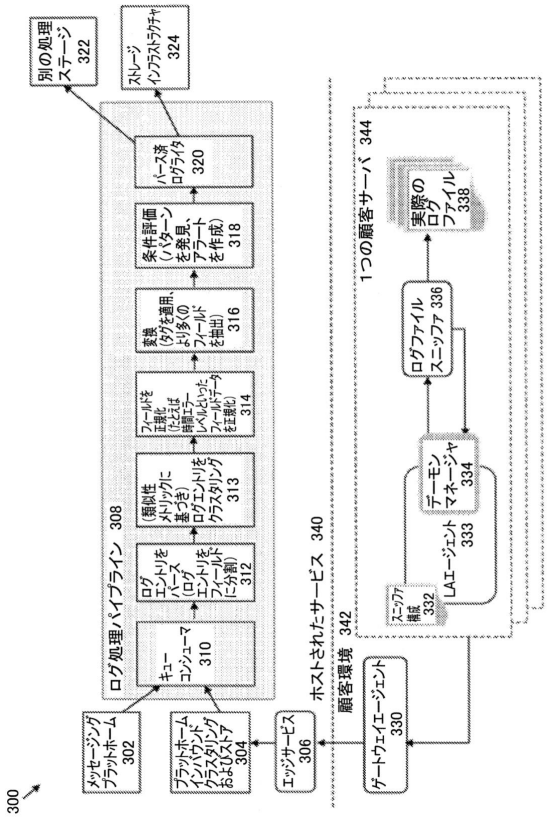


Fig. 3A

【図３Ｂ】

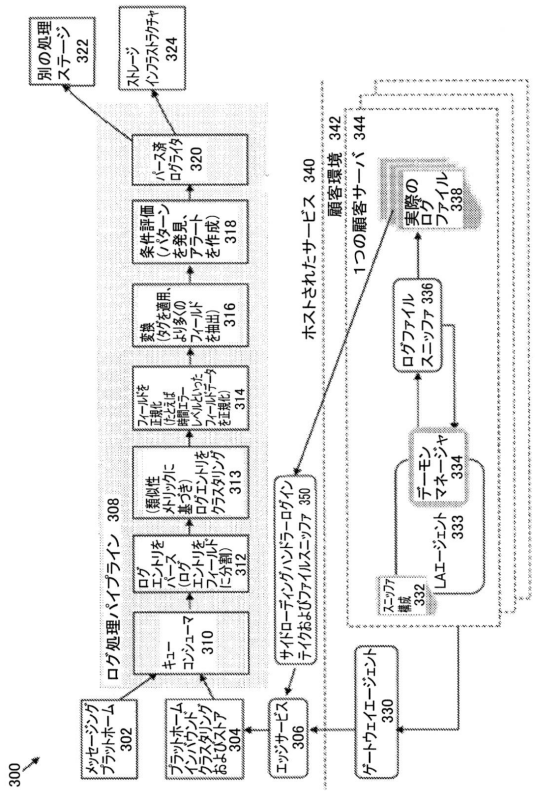


Fig. 3B

【図３Ｃ】

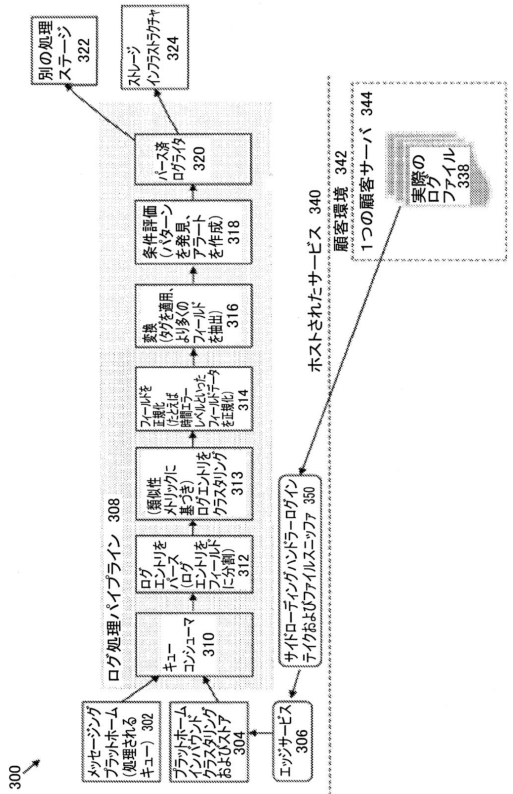


Fig. 3C

【図４】

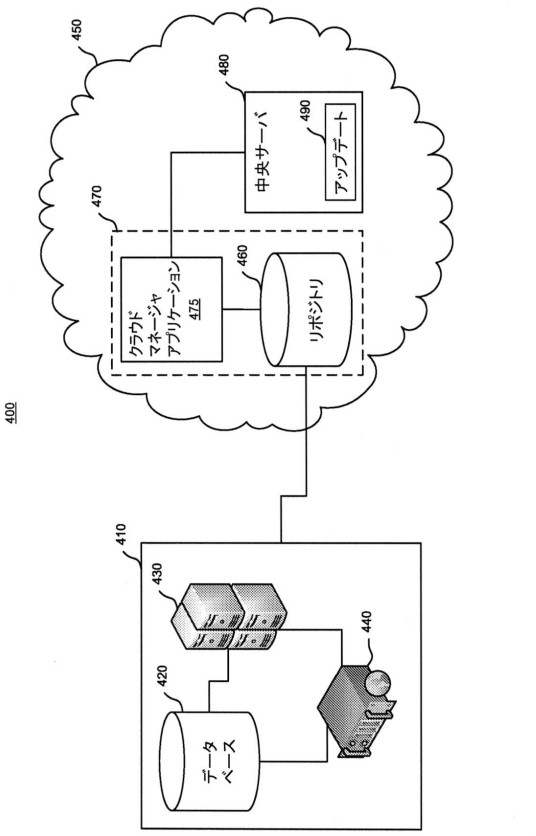


FIG. 4

【図５】

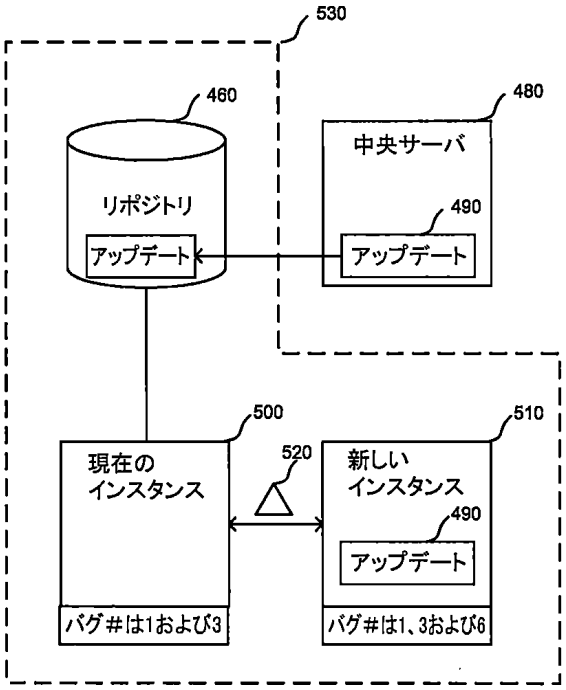


FIG. 5

【図６】

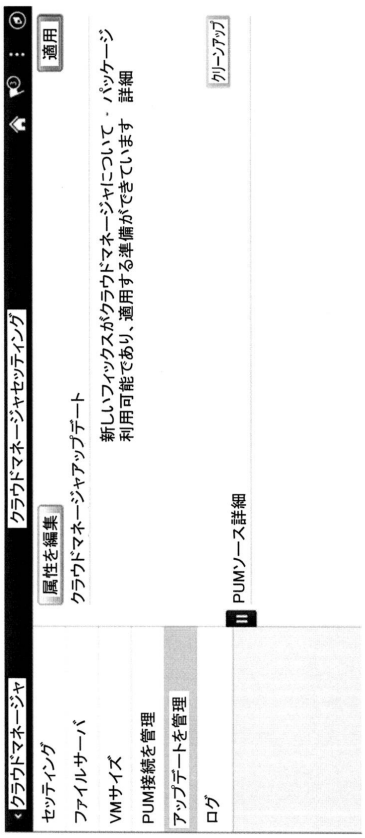


FIG. 6

【図７】

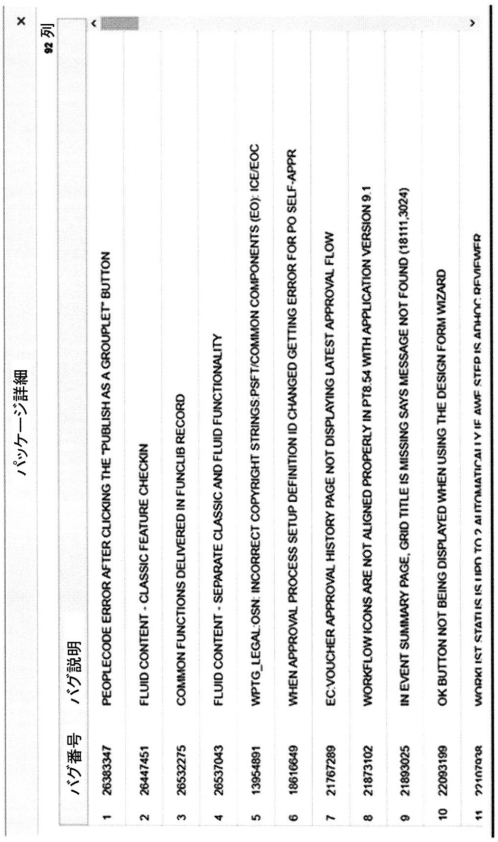
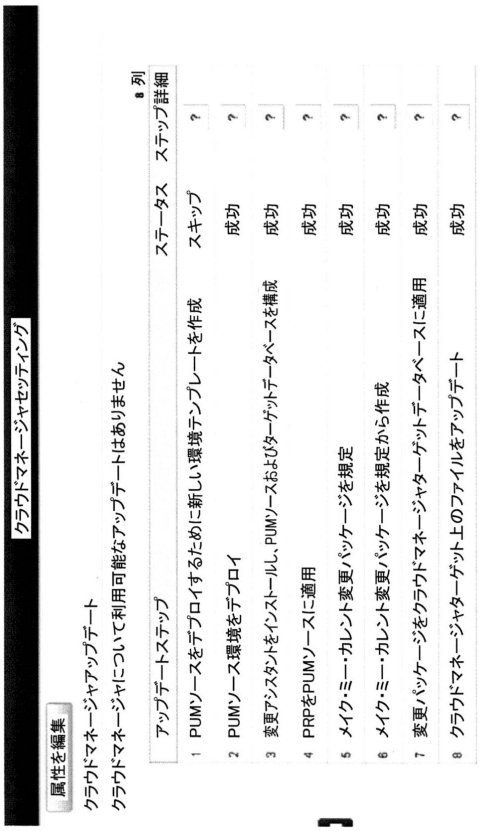


FIG. 7

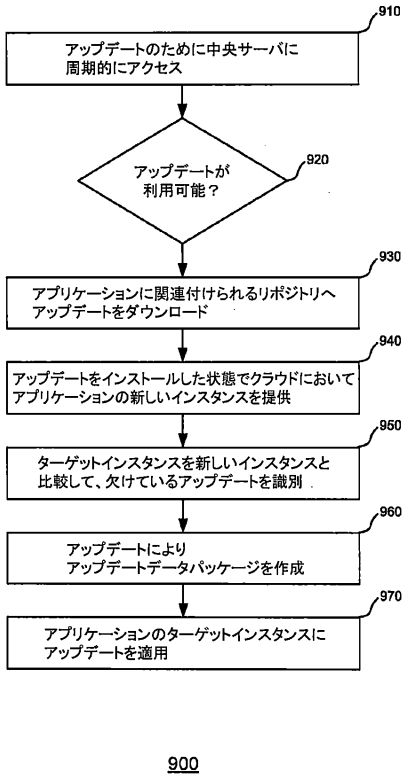
【図８】



800

FIG. 8

【図９】



900

FIG. 9

【図１０】

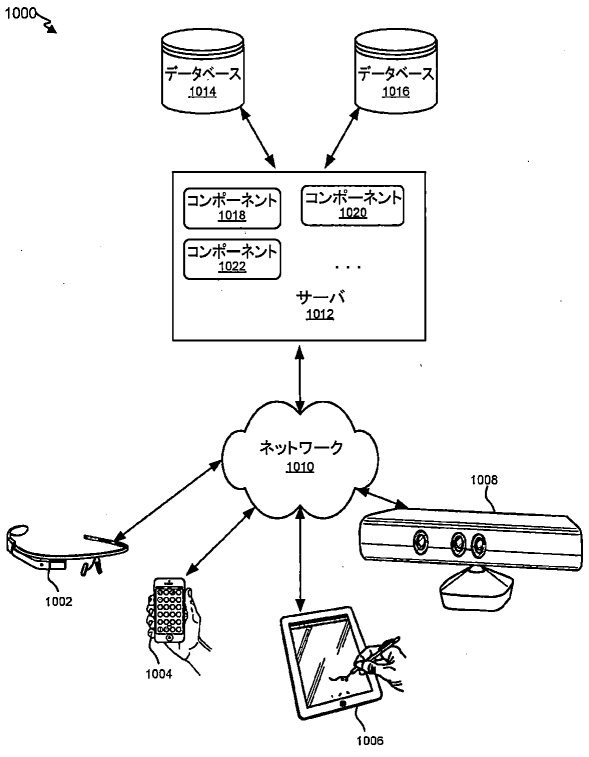


FIG. 10

【図１１】

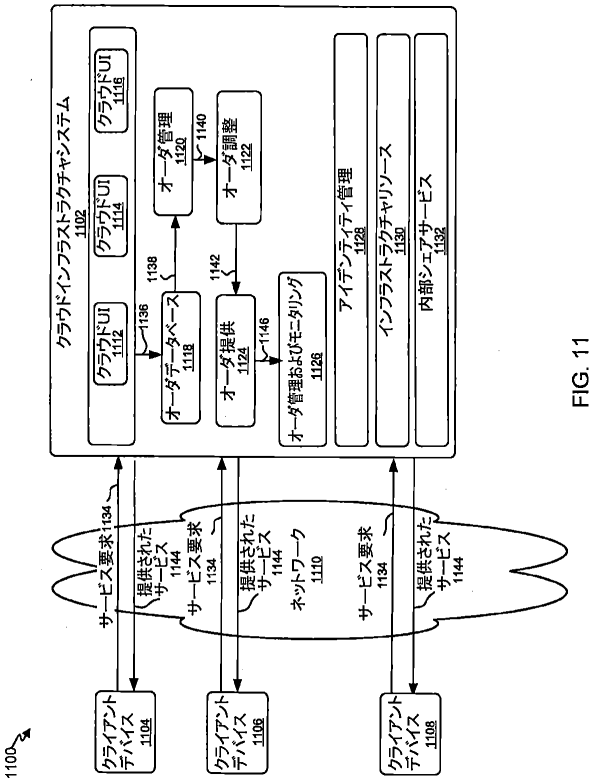


FIG. 11

【図１２】

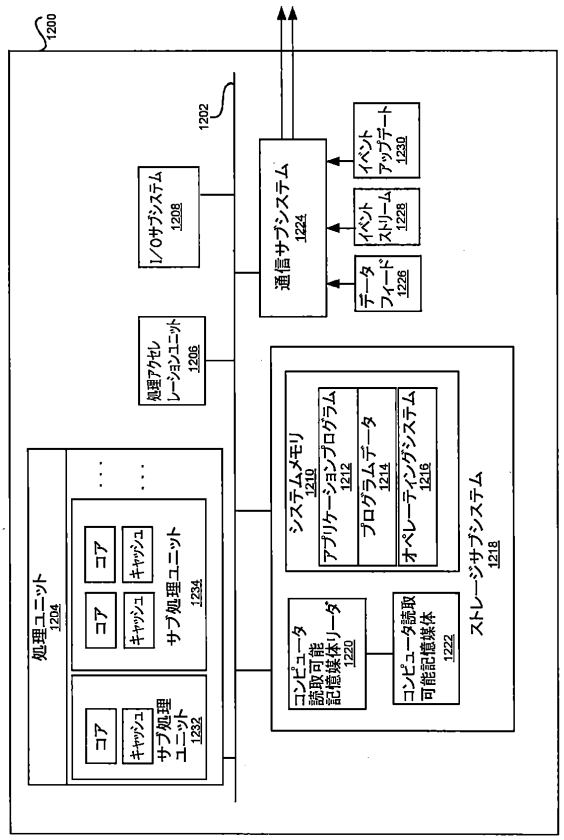

1200
1202
処理ユニット
1204
コア
キャッシュ
サブ処理ユニット
1232
コア
コア
キャッシュ
サブ処理ユニット
1234
処理アクセラレーションユニット
1206
I/Oサブシステム
1208
システムメモリ
1210
アプリケーションプログラム
1212
プログラムデータ
1214
オペレーティングシステム
1216
コンピュータ読取可能記憶媒体リーダ
1220
コンピュータ読取可能記憶媒体
1222
ストレージサブシステム
1218
通信サブシステム
1224
データフィード
1226
イベントストリーム
1228
イベントアップデート
1230

FIG. 12

フロントページの続き

(72)発明者　スジャータ，アニーシュ・アズハケサン
インド、６９１５１１ ケーララ、コラム、ケーララプラム、ベリモン・ピィ・オゥ、スジャババン
(72)発明者　アルドンカール，ミリンド・ゼテ・チャティム
インド、４０３８０２ ゴア、バスコ・ダ・ガマ、バイーナ、ニア・ピィ・ダブリュ・ディ・オフィス、バルミキ・ビルディング、フラット・ナンバー・エス－５
(72)発明者　ナラヤナン，ディーパンカール
インド、６９５００３ ケーララ、トリバンドラム、コウディアー・ピィ・オゥ、クラバンコナム、アッパー・メリディアン・ロード、コンドル・ダフォディルズ、１・エイ・２

審査官　中村 信也
(56)参考文献　米国特許出願公開第２０１５／０２２０３２４（ＵＳ，Ａ１）
米国特許出願公開第２０１０／０２８１４７３（ＵＳ，Ａ１）
特表２０１３－５２２７６９（ＪＰ，Ａ）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　８／６５
Ｇ０６Ｆ　９／４５５