

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7387734号
(P7387734)

(45)発行日 令和5年11月28日(2023.11.28)

(24)登録日 令和5年11月17日(2023.11.17)

(51)国際特許分類

F I

G 0 6 F 8/60 (2018.01)

G 0 6 F 8/60

G 1 6 Y 40/10 (2020.01)

G 1 6 Y 40/10

請求項の数 20 (全41頁)

(21)出願番号	特願2021-530235(P2021-530235)	(73)特許権者	521206419
(86)(22)出願日	令和1年11月12日(2019.11.12)		バンティック インコーポレイテッド
(65)公表番号	特表2022-510227(P2022-510227 A)		V A N T I Q, I N C.
(43)公表日	令和4年1月26日(2022.1.26)		アメリカ合衆国 9 4 5 9 6 カリフォルニア州 ウォルナット クリーク エヌ カリフォルニア ブルバード 1 9 9 0 ス イート 4 0 0
(86)国際出願番号	PCT/US2019/060999	(74)代理人	100105957
(87)国際公開番号	WO2020/112349		弁理士 恩田 誠
(87)国際公開日	令和2年6月4日(2020.6.4)	(74)代理人	100068755
審査請求日	令和4年11月11日(2022.11.11)		弁理士 恩田 博宣
(31)優先権主張番号	62/773,142	(74)代理人	100142907
(32)優先日	平成30年11月29日(2018.11.29)		弁理士 本田 淳
(33)優先権主張国・地域又は機関	米国(US)	(72)発明者	バターワース、ポール
			アメリカ合衆国 9 4 5 0 7 カリフォルニア州 最終頁に続く

(54)【発明の名称】 イベント駆動型アプリケーションのルールに基づく割り当て

(57)【特許請求の範囲】

【請求項1】

分散型IoT (Internet of Things) コンピューティング環境内のイベント駆動型アプリケーションの複数のイベント駆動型アプリケーション構成要素をデプロイする方法であって、前記イベント駆動型アプリケーションは、前記分散型IoT (Internet of Things) コンピューティング環境内の複数のセンサーからイベントデータを受信するように構成されており、前記方法は、

1つまたは複数のプロセッサを用いて、前記イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析することであって、前記複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、

割り当てられた計算ノードにおいてサポートされるべき前記複数のイベント駆動型アプリケーション構成要素の複数の構成要素要件を決定すること、前記複数のイベント駆動型アプリケーション構成要素への割り当てのために複数のノードセットのうちの抽象ノードセットを識別することであって、前記複数のイベント駆動型アプリケーション構成要素のサブセット内の各イベント駆動型アプリケーション構成要素によるリモート参照の記法宣言を識別することを含む前記抽象ノードセットを識別すること、前記複数のイベント駆動型アプリケーション構成要素の前記複数の構成要素要件、および前記複数のイベント駆動型アプリケーション構成要素間の識別された前記複数の関係に基づいて、前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、

10

20

を含む前記イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析すること、

前記分散型 I o T (Internet of Things) コンピューティング環境の実際のノードセットを前記抽象ノードセットに割り当てること、

1つまたは複数のプロセッサを用いて、前記複数のイベント駆動型アプリケーション構成要素を、前記分散型 I o T (Internet of Things) コンピューティング環境の前記実際のノードセットに自動的にデプロイすることであって、前記実際のノードセットは、前記抽象ノードセットに割り当てられている、前記実際のノードセットに自動的にデプロイすること、を備える方法。

【請求項 2】

各リモート参照は、論理コンピューティングリソース制約を満たす抽象的な一組のコンピューティングリソースを表し、前記リモート参照の前記記法宣言は、前記リモート参照の論理コンピューティングリソース制約の宣言を含む、請求項 1 に記載の方法。

【請求項 3】

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てることは、割り当てルールにさらに基づく、請求項 2 に記載の方法。

【請求項 4】

前記複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別することは、第 1 のイベント駆動型アプリケーション構成要素による第 2 のイベント駆動型アプリケーション構成要素への参照を識別することを含み、

前記複数のイベント駆動型アプリケーション構成要素についての複数の構成要素要件を識別することは、前記第 1 のイベント駆動型アプリケーション構成要素についての前記複数の構成要素要件を識別することを含み、

前記抽象ノードセットを識別することは、前記第 1 のイベント駆動型アプリケーション構成要素による第 1 のリモート参照の記法宣言を識別することを含む、請求項 3 に記載の方法。

【請求項 5】

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てることは、

前記第 1 のイベント駆動型アプリケーション構成要素の前記複数の構成要素要件に基づいて、前記第 1 のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、

前記割り当てルール及び前記第 1 のイベント駆動型アプリケーション構成要素による前記参照に基づいて、前記第 2 のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、を含む、請求項 4 に記載の方法。

【請求項 6】

前記第 2 のイベント駆動型アプリケーション構成要素は、プロシージャ構成要素またはタイプ構成要素である、請求項 4 に記載の方法。

【請求項 7】

前記複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別することは、第 3 のイベント駆動型アプリケーション構成要素によるイベント構成要素の生成を識別することを含み、

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てることは、

前記イベント構成要素を前記抽象ノードセットに割り当てること、

前記割り当てルール及び前記第 3 のイベント駆動型アプリケーション構成要素による識別された前記生成に基づいて、前記第 3 のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、を含む、請求項 3 に記載の方法。

【請求項 8】

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当て

10

20

30

40

50

ることは、共通の一組のイベントアプリケーション構成要素を、前記複数のノードセットの対応する抽象ノードセットの各ノードに割り当てることをさらに含む、請求項 1 に記載の方法。

【請求項 9】

対応するイベント駆動型アプリケーション構成要素が割り当てられる前記抽象ノードセットは、前記対応するイベント駆動型アプリケーション構成要素に割り当てられる最小コードの決定に基づいて識別される、請求項 1 に記載の方法。

【請求項 10】

前記アプリケーションソースコードを分析することは、特定のイベント駆動型アプリケーション構成要素の実行を動作的にトリガーするリソースを決定することを含む、請求項 1 に記載の方法。

【請求項 11】

前記アプリケーションソースコードを分析することは、特定のイベント駆動型アプリケーション構成要素によって参照されるリソースを決定することを含む、請求項 1 に記載の方法。

【請求項 12】

前記アプリケーションソースコードを分析することは、特定のイベント駆動型アプリケーション構成要素の識別されたコードセグメントの依存関係を識別することを含む、請求項 1 に記載の方法。

【請求項 13】

コンピューティング装置であって
少なくとも 1 つのプロセッサと、
複数の命令を記憶するメモリと、を備え、前記複数の命令は、前記少なくとも 1 つのプロセッサによって実行されると、前記コンピューティング装置を、
分散型 IoT (Internet of Things) コンピューティング環境内のイベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析することであって、前記イベント駆動型アプリケーションは、前記分散型 IoT (Internet of Things) コンピューティング環境内の複数のセンサーからイベントデータを受信するように構成されており、
前記イベント駆動型アプリケーションの複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、
割り当てられた計算ノードにおいてサポートされるべき前記複数のイベント駆動型アプリケーション構成要素の複数の構成要素要件を決定すること、
前記複数のイベント駆動型アプリケーション構成要素への割り当てのために複数のノードセットのうちの抽象ノードセットを識別することであって、前記複数のイベント駆動型アプリケーション構成要素のサブセット内の各イベント駆動型アプリケーション構成要素によるリモート参照の記法宣言を識別することを含む前記抽象ノードセットを識別すること、
前記複数のイベント駆動型アプリケーション構成要素の前記複数の構成要素要件、および前記複数のイベント駆動型アプリケーション構成要素間の識別された前記複数の関係に基づいて、前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、
を含む前記イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析すること、
前記分散型 IoT (Internet of Things) コンピューティング環境の実際のノードセットを前記抽象ノードセットに割り当てること、
1 つまたは複数のプロセッサを用いて、前記複数のイベント駆動型アプリケーション構成要素を、前記分散型 IoT (Internet of Things) コンピューティング環境の前記実際のノードセットに自動的にデプロイすることであって、前記実際のノードセットは、前記抽象ノードセットに割り当てられている、前記実際のノードセットに自動的にデプロイすること、
を実行するように設定する、コンピューティング装置。

【請求項 14】

各リモート参照は、論理コンピューティングリソース制約を満たす抽象的な一組のコンピューティングリソースを表し、前記リモート参照の前記記法宣言は、前記リモート参照の論理コンピューティングリソース制約の宣言を含む、請求項 13 に記載のコンピューティング装置。

【請求項 15】

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てることは、割り当てルールにさらに基づく、請求項 14 に記載のコンピューティング装置。

【請求項 16】

前記複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別することは、第 1 のイベント駆動型アプリケーション構成要素による第 2 のイベント駆動型アプリケーション構成要素への参照を識別することを含み、

10

前記複数のイベント駆動型アプリケーション構成要素についての複数の構成要素要件を識別することは、前記第 1 のイベント駆動型アプリケーション構成要素についての前記複数の構成要素要件を識別することを含む、請求項 15 に記載のコンピューティング装置。

【請求項 17】

前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てることは、

前記第 1 のイベント駆動型アプリケーション構成要素の前記複数の構成要素要件に基づいて、前記第 1 のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、

20

前記割り当てルール及び前記第 1 のイベント駆動型アプリケーション構成要素による前記参照に基づいて、前記第 2 のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、を含む、請求項 16 に記載のコンピューティング装置。

【請求項 18】

複数の命令を含む非一時的なコンピュータ記憶媒体であって、前記複数の命令は、少なくとも 1 つのコンピュータによって実行されると、前記少なくとも 1 つのコンピュータに複数の動作を実行させ、前記複数の動作は、

分散型 IoT (Internet of Things) コンピューティング環境内のイベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析することであって、前記イベント駆動型アプリケーションは、前記分散型 IoT (Internet of Things) コンピューティング環境内の複数のセンサーからイベントデータを受信するように構成されており、前記イベント駆動型アプリケーションの複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、

30

割り当てられた計算ノードにおいてサポートされるべき前記複数のイベント駆動型アプリケーション構成要素の複数の構成要素要件を決定すること、

前記複数のイベント駆動型アプリケーション構成要素への割り当てのために複数のノードセットのうちの抽象ノードセットを識別することであって、前記複数のイベント駆動型アプリケーション構成要素のサブセット内の各イベント駆動型アプリケーション構成要素によるリモート参照の記法宣言を識別することを含む前記抽象ノードセットを識別すること、前記複数のイベント駆動型アプリケーション構成要素の前記複数の構成要素要件、および前記複数のイベント駆動型アプリケーション構成要素間の識別された前記複数の関係に基づいて、前記複数のイベント駆動型アプリケーション構成要素を前記抽象ノードセットに割り当てること、

40

を含む前記イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析すること、

前記分散型 IoT (Internet of Things) コンピューティング環境の実際のノードセットを前記抽象ノードセットに割り当てること、

1 つまたは複数のプロセッサを用いて、前記複数のイベント駆動型アプリケーション構成要素を、前記分散型 IoT (Internet of Things) コンピューティング環境の前記実際の

50

ノードセットに自動的にデプロイすることであって、前記実際のノードセットは、前記抽象ノードセットに割り当てられている、前記実際のノードセットに自動的にデプロイすること、を含む、非一時的なコンピュータ記憶媒体。

【請求項 19】

前記複数のノードセットはデフォルトノードセットを含み、各イベント駆動型アプリケーション構成要素は、前記自動的に分析することの前に前記デフォルトノードセットに割り当てられる、請求項 1 に記載の方法。

【請求項 20】

前記複数のノードセットはデフォルトノードセットを含み、各イベント駆動型アプリケーション構成要素は、前記自動的に分析することの前に前記デフォルトノードセットに割り当てられる、請求項 13 に記載のコンピューティング装置。

【発明の詳細な説明】

【背景技術】

【0001】

リアルタイムのイベント駆動型アプリケーションが次世代のビジネスアプリケーションとして注目的になりつつあり、デジタル事業へと向かう事業の移行を支えている。パーソナライズされた最適な体験を提供する次世代の計画、運用および顧客エンゲージメントアプリケーションは、リアルタイムの感知と、ほぼリアルタイムの決定とに依存する。このようなアプリケーションは、現代的なイベント駆動型アプリケーションプラットフォームに構築されなければならない。

【0002】

「アプリケーション分割 (application partitioning)」という用語は、アプリケーションロジックをネットワーク内の 2 以上のコンピュータに分散するアプリケーション開発プロセスをいう。最も単純なケースでは、アプリケーションは、リモートサービスとして 1 台の PC 上で動作し、実行するためのタスク要求をサーバに送信する。より高度なケースでは、アプリケーションロジックは、数台のサーバに分散することができる。

【0003】

現在のアプリケーション分割システムは、ローカルエリアネットワークに分散されるオブジェクト指向型アプリケーションの小分け (portioning) に集中する (focus)。このような分割システムは、ユーザが特定のオブジェクトインスタンスを特定し、さらにそれを手動で特定の計算ノードに配置することに依存する。手動アサイン (manual assignments) が実施されると、分割システムは、手動アサインを考慮せずに、続いて残りの構成要素をパーティションに割り当て、分散型通信システムでのアクセスのために各ノードを表すオブジェクトをバインドする (binds)。

【発明の概要】

【0004】

例示的な実施形態は、コンピュータネットワーク全体に分散された複数の計算ノードにイベント駆動型コンピュータアプリケーションの複数の構成要素を自動的に割り当てる (またはアサインする) 分割システムに関する。複数の構成要素の関係の分析と複数の構成要素の割り当ての両方を含む自動割り当ては、アプリケーションの複数の構成要素を最適な計算ノードに割り当てるために必要な手作業 (manual labor) を減らし、コンピュータネットワーク内の各ノードに必要なすべての構成要素を組み合わせる際のエラーを減らし、分割されたアプリケーションの効率を向上させる。

【0005】

例示的な分割システムは、分散された計算を意図的に表現するメカニズムを含む。分割システムは、ソースコード内の構成要素関係を識別し、複数の計算ノードへの複数の構成要素の最適な割り当てを決定するため、および各計算ノードに割り当てられた構成要素の実行をサポートするために必要かつ十分な一組の構成要素を決定するために、一組のルールを前記分析の結果に適用する一組のソースコードアナライザを実装する。

【図面の簡単な説明】

【0006】

ある特定の要素または行為の考察を特定しやすくするために、参照符号における最上位の1つまたは複数の桁は、その要素が最初に紹介される図面の番号を示す。

【図1】一実施形態による、サービスとしてのプラットフォーム（PaaS：Platform-as-a-Service）100を示す。

【図2】一実施形態による、アーキテクチャ200を示す。

【図3】一実施形態による、デプロイメント300を示す。

【図4】一実施形態による、デプロイメント環境400を示す。

【図5】一実施形態による、デプロイメント環境500を示す。

【図6】一実施形態による、デプロイメント環境600を示す。

【図7】一実施形態による、分散型コンピューティング環境700を示す。

【図8】一実施形態による、動作1202を示す。

【図9】一実施形態による、方法900を示す。

【図10】一実施形態による、動作1000を示す。

【図11】一実施形態による、動作1100を示す。

【図12】一実施形態による、動作902を示す。

【図13】いくつかの例示的な実施形態による、本開示を実施することのできるソフトウェアアーキテクチャを示すブロック図である。

【図14】いくつかの例示的な実施形態による、本明細書に述べられる方法論のうちの任意の1つまたは複数を機械に行わせるための命令セットを実行することのできるコンピュータシステムの形態の機械の図表示である。

【発明を実施するための形態】

【0007】

「搬送波信号（Carrier Signal）」とは、機械（machine）により実行するための命令を格納、符号化または搬送することが可能な任意の無形媒体をいい、当該命令の通信を可能にするためのデジタルもしくはアナログの通信信号または他の無形媒体を含む。命令は、ネットワークインターフェースを介して伝送媒体を使用してネットワークで送受信されてもよい。

【0008】

「通信ネットワーク」とは、アドホックネットワーク、イントラネット、エクストラネット、仮想プライベートネットワーク（VPN：virtual private network）、ローカルエリアネットワーク（LAN：local area network）、無線LAN（WLAN：wireless LAN）、ワイドエリアネットワーク（WAN：wide area network）、無線WAN（WWAN：wireless WAN）、メトロポリタンエリアネットワーク（MAN：metropolitan area network）、インターネット、インターネットの一部、公衆交換電話網（PSTN：Public Switched Telephone Network）の一部、基本電話サービス（POTS：plain old telephone service）網、携帯電話ネットワーク、無線ネットワーク、Wi-Fi（登録商標）ネットワーク、別のタイプのネットワーク、または2以上のこのようなネットワークの組合せとしてもよい、ネットワークの1つまたは複数の部分をいう。例えば、ネットワークまたはネットワークの一部は、無線またはセルラーネットワークを含んでもよく、結合（coupling）は、符号分割多元接続（CDMA：Code Division Multiple Access）接続、グローバル移動体通信システム（GSM：Global System for Mobile）接続、または他のタイプのセルラーもしくは無線結合としてもよい。この例では、結合は、シングルキャリア無線伝送技術（1xRTT：Single Carrier Radio Transmission Technology）、エボリューション・データ・オブティマイズド（EVDO：Evolution-Data Optimized）技術、汎用パケット無線サービス（GPRS：General Packet Radio Service）技術、GSM進化型高速データレート（EDGE：Enhanced Data rates for GSM Evolution）技術、3Gなどの第三世代パートナーシッププロジェクト（3GPP：third Generation Partnership Project）、第四世代無線（4G）ネットワーク、ユニバーサル移動体通信システム（UMTS：Universal Mobile Telecommunications System）、高速パケットアクセ

10

20

30

40

50

ス（HSPA：High Speed Packet Access）、マイクロ波アクセスの世界的な相互運用性（WiMAX：Worldwide Interoperability for Microwave Access）、ロングターム・エボリューション（LTE：Long Term Evolution）規格、様々な標準化団体が定義する他のもの、その他長距離通信プロトコル、またはその他データ転送技術など、多様なタイプのデータ転送技術のいずれかを実施してもよい。

【0009】

「構成要素（Component）」とは、機能もしくはサブルーチンの呼出し、分岐点、API、または特定の処理もしくは制御機能の分割もしくはモジュール化を供する他の技術によって画定される境界を有するデバイス、物理的エンティティ、またはロジックをいう。機械プロセスを遂行するために、構成要素は、そのインターフェースを介して他の構成要素と組み合わせてもよい。構成要素は、他の構成要素とともに用いられるように設計されるパッケージ化された機能的ハードウェアユニット、および通常は関連機能のうちの特定の機能を果たすプログラムの一部であってもよい。構成要素は、ソフトウェア構成要素（例えば、機械可読媒体上に具現されるコード）またはハードウェア構成要素のいずれを構成してもよい。「ハードウェア構成要素」とは、一定の動作を行うことが可能な有形ユニットであり、一定の物理的な形で構成または配置されてもよい。様々な例示的な実施形態において、1つもしくは複数のコンピュータシステム（例えば、独立型コンピュータシステム、クライアントコンピュータシステム、もしくはサーバコンピュータシステム）またはコンピュータシステムの1つもしくは複数のハードウェア構成要素（例えば、プロセッサもしくはプロセッサのグループ）は、ソフトウェア（例えば、アプリケーションまたはアプリケーションの一部／構成要素）により、本明細書で説明される一定の動作を行うように作動するハードウェア構成要素として構成してもよい。ハードウェア構成要素は、機械的、電子的、またはその任意の適した組合せで実施されてもよい。例えば、ハードウェア構成要素は、一定の動作を行うように永久的に構成された専用回路系（dedicated circuitry）またはロジックを含んでもよい。ハードウェア構成要素は、フィールドプログラマブルゲートアレイ（FPGA：field-programmable gate array）または特定用途向け集積回路（ASIC：application specific integrated circuit）などの専用プロセッサとしてもよい。ハードウェア構成要素は、ソフトウェアにより、一定の動作を行うように一時的に構成されたプログラマブルロジックまたは回路系を含んでもよい。例えば、ハードウェア構成要素は、汎用プロセッサまたはその他プログラマブルプロセッサによって実行されるソフトウェアを含んでもよい。このようなソフトウェアによって構成されれば、ハードウェア構成要素は、構成された機能を果たすように特異的に誂えられた（uniquely tailored）特殊な機械（または機械の特殊構成要素）になり、汎用プロセッサではなくなる。ハードウェアを機械的に、専用の永久的に構成された回路系、または一時的に構成された回路系（例えば、ソフトウェアによって構成される）に実装するという決定が、費用および時間を考慮して推進されうことは認識されるであろう。したがって、「ハードウェア構成要素」（または「ハードウェア実装構成要素」）という語句は、本明細書に説明される一定の態様で動作するように、または一定の動作を行うように物理的に構築され、永久的に構成され（例えば、ハードウェアに組み込まれ）、または一時的に構成される（例えば、プログラミングされる）エンティティである、有形のエンティティを包含すると理解されるべきである。ハードウェアが一時的に構成される（例えば、プログラミングされる）実施形態を考えると、ハードウェア構成要素のそれぞれを一度に構成またはインスタンス化する必要はない。例えば、ハードウェア構成要素が、専用プロセッサになるようにソフトウェアによって構成された汎用プロセッサを備える場合、汎用プロセッサは、異なる時にそれぞれ異なる専用プロセッサとして構成されてもよい（例えば、異なるハードウェア構成要素を備える）。それに応じて、ソフトウェアは、例えば、ある時点で特定のハードウェア構成要素を構成し、異なる時点では異なるハードウェア構成要素を構成するように、特定の1つまたは複数のプロセッサを構成する。ハードウェア構成要素は、他のハードウェア構成要素に情報を提供するとともに、他のハードウェア構成要素から情報を受信することができる。したがって、説明されるハードウェア構成要素は、通信可能

10

20

30

40

50

に結合されているとみなすこともできる。複数のハードウェア構成要素が同時に存在する場合、通信は、2以上のハードウェア構成要素間での信号伝送によって（例えば、適切な回路およびバスで）実現してもよい。複数のハードウェア構成要素が異なる時に構成またはインスタンス化される実施形態では、当該ハードウェア構成要素間の通信は、例えば、複数のハードウェア構成要素がアクセスできるメモリ構造内での情報の格納および探索により実現してもよい。例えば、1つのハードウェア構成要素が動作を行い、それが通信可能に結合されているメモリデバイスにその動作の出力を格納してもよい。それから、別のハードウェア構成要素が、後の時点で、メモリデバイスにアクセスして、格納された出力を探索して処理してもよい。ハードウェア構成要素は、入力デバイスまたは出力デバイスとの通信を開始してもよく、リソース（例えば、情報の集合体）に働きかけることができる。本明細書で説明される例示的な方法の様々な動作は、少なくとも部分的には、関連動作を行うように、（例えば、ソフトウェアによって）一時的に構成された、または永久的に構成された1つまたは複数のプロセッサによって行ってもよい。一時的に構成されるか永久的に構成されるかを問わず、当該プロセッサは、本明細書で説明される1つもしくは複数の動作または機能を行うように作動するプロセッサ実装構成要素を構成してもよい。本明細書で使用されるとき、「プロセッサ実装構成要素（processor-implemented component）」とは、1つまたは複数のプロセッサを用いて実装されるハードウェア構成要素をいう。同様に、本明細書で説明される方法は、少なくとも部分的にプロセッサに実装されてもよく、特定の1つまたは複数のプロセッサは、ハードウェアの例である。例えば、方法の動作の少なくともいくつかは、1つもしくは複数のプロセッサまたはプロセッサ実装構成要素によって行ってもよい。さらに、1つまたは複数のプロセッサは、「クラウドコンピューティング」環境で、または「サービスとしてのソフトウェア（software as a service）」（SaaS）として関連動作の遂行をサポートするように作動してもよい。例えば、動作の少なくともいくつかは、コンピュータのグループによって（プロセッサを含む機械の例として）行ってもよく、これらの動作は、ネットワーク（例えば、インターネット）を介して、および1つまたは複数の適切なインターフェース（例えば、API）を介してアクセス可能である。一定の動作の遂行は、1台の機械の中に常駐しているだけでなく、多数の機械にデプロイされている（deployed）プロセッサ間に分散してもよい。いくつかの例示的な実施形態において、プロセッサまたはプロセッサ実装構成要素は、1つの地理的な場所に所在してもよい（例えば、家庭環境、オフィス環境、またはサーバファーム内）。他の例示的な実施形態では、プロセッサまたはプロセッサ実装構成要素は、多数の地理的な場所にわたって分散していてもよい。

【0010】

「コンピュータ可読媒体」とは、機械記憶媒体および伝送媒体の両方をいう。したがって、この用語は、記憶装置／媒体と、搬送波／変調されたデータ信号の両方を含む。「機械可読媒体」、「コンピュータ可読媒体」および「デバイス可読媒体」という用語は同じものを意味し、本開示において交換可能に使用することができる。

【0011】

「機械記憶媒体」とは、実行可能な命令、ルーチンおよび／またはデータを格納する、1台もしくは複数の記憶装置および媒体または記憶装置もしくは媒体（例えば、集中型もしくは分散型データベースと、関連のキャッシュおよびサーバと、の両方または一方）をいう。したがって、この用語は、プロセッサの内部または外部のメモリを含め、固体メモリ、ならびに光学媒体および磁気媒体を含むが、これだけに限定されないと解釈されるものとする。機械記憶媒体、コンピュータ記憶媒体および／またはデバイス記憶媒体の具体的な例には、例として半導体メモリデバイスを含む不揮発性メモリ、例えば、消去可能プログラマブル読み取り専用メモリ（EPROM：erasable programmable read-only memory）、電氣的に消去可能なプログラマブル読み取り専用メモリ（EEPROM：electrically erasable programmable read-only memory）、FPGA、およびフラッシュメモリデバイス、内部ハードディスクおよびリムーバブルディスクなどの磁気ディスク、光磁気ディスク、ならびにCD-ROMおよびDVD-ROMディスクが含まれる。「機械記

10

20

30

40

50

憶媒体」、「デバイス記憶媒体」、「コンピュータ記憶媒体」という用語は同じものを意味し、本開示において交換可能に使用することができる。「機械記憶媒体」、「コンピュータ記憶媒体」および「デバイス記憶媒体」という用語は、特に、搬送波、変調されたデータ信号および他のこのような媒体を除外し、そのうちの少なくともいくつかは、「信号媒体 (signal medium)」という用語でカバーされる。

【0012】

「信号媒体」とは、機械によって実行するための命令を格納、符号化または搬送することが可能な任意の無形媒体をいい、ソフトウェアまたはデータの通信を可能にするデジタルもしくはアナログの通信信号またはその他の無形媒体を含む。「信号媒体」という用語は、任意の形態の変調されたデータ信号、搬送波などを含むと解釈されるものとする。「変調されたデータ信号」という用語は、信号内の情報を符号化するようなことで設定または変更される1つまたは複数の特性を有する信号を意味する。「伝送媒体 (transmission medium)」および「信号媒体」という用語は同じものを意味し、本開示において交換可能に使用することができる。

【0013】

「メッシュネットワーク (Mesh Network)」とは、計算ノードができるだけ多くの他のノードに、例えば、直接、動的におよび非階層的に接続する分散型コンピューティング環境のためのネットワークトポロジ (network topology) をいう。そのため、ノードが協力し合って、そのソースからその宛先までのネットワークトラフィックの経路を指定してもよい。メッシュネットワークは、動的に自己組織化し、自己設定してもよい。他のすべてのピアノードに各ノードが接続しているメッシュネットワークは、フルメッシュネットワーク (full mesh network) として知られる。大きいけれども潜在的に選択的なピアノードの集合に各ノードが接続しているメッシュネットワークは、パーシャルメッシュ (partial mesh) として知られる。

【0014】

分割 (PARTITIONING)

いくつかの例示的な実施形態による、イベントベースのアプリケーションを分割する技術およびメカニズムが、本明細書において説明される。具体的には、サービスとしてのプラットフォーム (Platform-as-a-Service : PaaS) 100が説明され、これは、分散型コンピューティング環境内のノードセットを特定し、さらにイベントベースのアプリケーションにおいてイベントの識別を使用して、当該イベントベースのアプリケーションの自動分割を行うように作動するデプロイメントマネージャ (deployment manager) を含む。このために、デプロイメントマネージャ、より具体的には、デプロイメントマネージャの一部を形成する分割システムがイベント駆動型アプリケーションのソースコードを分析して、イベント駆動型アプリケーションの構成要素間の関係を推定する。分割システムは、次いで、ルールシステム (例えば、アサインルール) を介して、既知の関係のナレッジベース (knowledge base) を適用して、分割作業を行う。次いで、分割システムは、1つまたは複数の設定ファイルとして、自動分割を反映する設定 (configurations) を出力する。

【0015】

説明される実施形態は、分割作業を駆動するためにオブジェクトインスタンスを前もって生成することを必要としないことから、説明される実施形態は、公知の解決策を上回る多数の技術的な利点を提供する。また、説明される実施形態は構成要素の配置を最適化するのに対し、多くの現在の解決策は、コンピューティング環境内のすべての計算ノード上に実行可能なコードをすべて配置するように作動する。さらに、説明される実施形態は、オブジェクトがただ単に必要なためにオブジェクトと一緒にバインドせず、バインディング (bindings) は、イベントベースのアプリケーションのアーキテクチャの機能として動的である。

【0016】

イベントブローカー (EVENT BROKER)

いくつかの例示的な実施形態によると、メッシュイベントブローカーが提供され、当該メッシュイベントブローカーは、1つまたは複数のイベントエコシステムに参加する分散ノードにイベントの取得、拡張および配信を分散し、当該分散ノードは、イベント発行者、イベントサブスクライバおよび仲介者を含む。いくつかの実施形態では、メッシュイベントブローカーは、メディエータを単一障害点（single point of failure）として除き、作業負荷を分散し、スケーリングを単純化および拡大し、異なるネットワーク内のイベント発行者とイベントサブスクライバとの相互接続を提供することを図る。

【0017】

いくつかの例示的な実施形態によると、協働するエージェントの集合体が「メディエータ」として作用し、直接的な2点間通信をサポートするメッシュネットワークを形成する。協働する各エージェントは、拡張をサポートするとともに、サポート対象の発行者によって発行されるイベントに発行者とサブスクライバとの特定のセットの配信を行うためにプロビジョニングされる（provisioned）。同様に、サブスクライバは、自身がローカルサブスクライバであるイベントを拡張および配信するためにプロビジョニングされてもよい。加えて、協働するエージェントは、メッシュとして、メッセージをメッシュに転送して、イベント伝送の効率を改善するとともに、異なるネットワーク上に設定されるエージェント間の橋渡しをすることができる。

【0018】

イベント駆動型アプリケーション（EVENT-DRIVEN APPLICATIONS）

いくつかの例示的な実施形態によると、イベント駆動型アプリケーションは、応答性、ロバストネスおよびセキュリティを改善するために、分散されるようにデプロイしてもよい。本明細書で説明されるとき、イベント駆動型アプリケーションは、1つのクラウドアプリケーションで開発されてから、自動的に分割されてもよく、その結果、アプリケーションの構成要素は、ノードがクラウドホスト型か、データセンターホスト型か、エッジのインテリジェントデバイスか、またはそれらの組合せかどうかを問わず、実行のために最適なノードに分散されることになる。ロジックは、それが最も効果的なところに配置される。星型、階層型およびピアツーピア型などの広範囲のシステムトポロジがサポートされる。これらのネットワークのプロビジョニングと管理は、自動化されて、本明細書で説明するサービスとしてのプラットフォーム（PaaS）100のインテリジェント機能によって管理される。アプリケーション構成要素は、システムが動作している間、何千または何万のノードのための分散型環境のどこでも動的に変更することができる。

【0019】

説明される例示的な実施形態は、システムの開発がビジネスロジックに集中することができ、必ずしも基盤となるインフラストラクチャに集中しなくてよいように、リアルタイムのイベント駆動型アプリケーションの設計、プロビジョニングおよび管理の自動化も図る。このために、サービスとしてのプラットフォーム（PaaS）100は、イベント駆動型ビジネスアプリケーションを構築し、デプロイしおよび作動させることのできる、速度と効率の改善を図る能力および統合を提供する。

【0020】

いくつかの例示的な実施形態によるイベント駆動型アプリケーションは、以下のフローを組み込んでもよい。

- ・入力は、例えば長期間かけて、多数のセンサから受信する。センサは、例えば、物理的センサ、他のエンタープライズシステムによって生成されるデータストリームまたは公開データストリームとしてもよい。

- ・センサデータを分析して、情報およびコンテキストからなるイベントを発生させ、それに対して自動化、推奨（recommendation）および協調の決定が行われる。追加のコンテキストを他のシステムから抽出して、センサデータを拡張してもよい。

- ・イベントは、リアルタイムで評価して、取る必要のあるアクションを決定する。例えば、リアルタイム評価を行うために、個別的ルールおよび機械学習戦略のうちの少なくとも一方を使用してもよい。

10

20

30

40

50

・アクションを、実施を担当するシステムに伝送するか、または担当者によって人間と機械の協調を開始し、現状に対して最も適切な応答を決定する。

【0021】

リアルタイムのイベント駆動型ビジネスアプリケーションでは、処理は、制御下のデバイスにローカルに行ってもよいので、応答時間および信頼性が改善される。例えば、工業的な環境では、材料取扱いシステムの位置を管理するには、数百ミリ秒内のほぼリアルタイムの応答をする必要がある。ネットワークに問題があれば数千ミリ秒の遅れが生じうる遠隔意思決定システムでは、当該応答時間を保証することはできない。状況データへのアクセスおよび制御アクションを開始する能力を慎重に管理するセキュアな環境で、処理が行われる。

【0022】

高レベルでは、本明細書に説明されるようなイベント駆動型アプリケーションは、データ取得、状況分析および検出された状況に対する応答アクションなどの動作を行うように作動してもよい。

【0023】

まずデータ取得または感知に対処して、イベント駆動型アプリケーションは、多数のセンサのうちの任意のものからデータを受信してもよい。センサには、例えば、以下のものを含めてもよい。

- ・センサの生データから導出されるロケーション、加速度、音声、映像および行動パターンを含むセンサデータをホストする携帯機器

- ・腕時計、活動量計、ヘルスマニタ、音声および映像ヘッドセットなどのウェアラブルデバイス

- ・産業機械、陸上および航空輸送機関、家電、ならびに感知と制御のうちの少なくとも一方ができる任意の機械的または電子的機器を含む機械。例えば、様々な破砕点 (crush points) を有する物体にかかる圧力を変えるために圧力センサを装備されているロボットのマニピュレータを想像されたい。

- ・数多くデプロイされる独立型センサ。例えば、作物の成長率を最大化しながら、水消費量を最小化するために、農場中に分散される水分センサ。

- ・センサデータと考えることができるものを大量に生成する映像および音声フィード。認識ソフトウェアを使用して映像が表すものを決定し、映像をより個別的なイベントに翻訳して、自動化の決定がそれに依存できるようにする。

- ・トランザクションのストリームを生成する既存のエンタープライズアプリケーション。

【0024】

このようなセンサは、それ自体のIP通信スタックでインターネットに直接接続することができ、またはエッジノードを介して間接的にインターネットに接続してもよい。後者の場合、センサ自体は、ModbusまたはZigBeeなどのより特化したプロトコルでプロトコル変換をもたらすエッジノードと通信してもよいので、センサは、IoTに参加する仮想ノードのように見える。

【0025】

ここで状況分析に移ると、データが取得されてしまえば、リアルタイムのイベント駆動型アプリケーションが、データの分析と、応答を要するビジネスまたは、技術的な条件を表すイベントまたは状況の発生とを担ってもよい。イベント駆動型アプリケーションは、次いで、機械もしくは顧客の現状に対する自動応答、および適切なオペレーションスタッフとシステムとの間の協調、の両方または一方を開始して、最適な応答を発生させてもよい。

【0026】

イベントおよび状況は、ルール、統計手法、および機械学習を用いて、データストリームおよびそのコンテキストを分析することによって検出してもよい。

分析中に検出されうるイベントまたは状況の例には、単なる例として、以下のものが含まれる。

- ・高温または低速などの条件があり予想通りに機能していない機器。

10

20

30

40

50

- ・店舗または施設内の関心のある場所に到着した顧客。例えば、顧客がレジまたは特定の商品ディスプレイのところに立っている。
- ・ユーザが安全ではないエリアにいて、助けが必要である。
- ・製品管理の注意を要する注文の配送が変更された。

【0027】

状況が検知されたら、その状況に対する応答をイベント駆動型アプリケーションによって生成してもよい。その応答は、自動化システムによって自動的に開始される応答であっても、または自動化システムと担当者との協調によって決定される応答であってもよい。応答には、以下のものを含んでもよい。

- ・コンシューマの現在の状況に基づいて、コンシューマに関連のある応答を与えること（例えば、セール中のアイテム、施設の地図、緊急時対応の推奨）。
- ・異常な状態に対してインテリジェントに応答すること（例えば、弁を閉じる、スプリンクラーをオンにする、誤作動するロボットを停止する）。
- ・現在の状況に基づいて、機会／問題をスタッフに早めに警告すること（例えば、利用可能な追加の配送トラック、サプライチェーンの一部の不足）。
- ・生産性と顧客満足度とのうちの少なくとも一方を上げるために、ユーザまたはビジネスリソースを最適化すること（例えば、組立ラインのスピードアップ、スポーツ参加者に自家用車までの最短経路を知らせる）。

【0028】

状況に応じて、自動化された応答は、リアルタイムのイベント駆動型ビジネスアプリケーションによって直接受けてもよく、または実施するためにより特化されたシステムに転送してもよい。例えば、機械を停止させるアクションは、アプリケーションに停止コマンドを機械へ直接送信させるのではなく、機械を直接管理する制御システムに転送してもよい。

【0029】

最適な応答がやや曖昧になりうる状況、または最適な応答の決定がシステムの能力を超えている状況では、システムと担当者とを巻き込んだ協調作業で最適な応答を引き出す。例えば、センサの読取値は、機械に潜在的な問題があることを示している可能性があるが、自動的に機械の停止を決定するほどには十分な情報がないことがある。代わりに、オペレーションチームがシステムと協調して、現在のデータを検証し、例えば、機械の目視点検により追加情報を得て、その状況が機器の停止を必要とするかどうかを判断する。

【0030】

- 協調が最適な結果を生み出すことのできるいくつかのケースは、以下の通りである。
- ・データストリームが、根本原因を一意に明らかにして最善なアクションの方針を決定するには十分ではない例外状況。
- ・オペレーションチームが、システムには入手できない追加情報を知りうる立場にある状況。
- ・システムの側に、オンラインで制御できない手動アクションを取らなければならない状況。
- ・アクションを行えるようになる前に、ポリシーまたは規制により、より徹底的な状況分析が要求される状況。

【0031】

別の重要な協調のクラスは、取られるアクションと、その結果としてのシステムの新たな状態とを関係者に通知する。通知は、他の自動化システムに配信して、かかる他の自動化システムが独立して状況に応答できるようにすることができ、または、デスクトップPC、携帯機器およびウェアラブルデバイスを介して担当スタッフに配信することができる。通知は、推奨されるアクションおよび未解決の問題の状況認識も含むことができる。

【0032】

リアルタイムのイベント駆動型ビジネスアプリケーションを分散させてもよい。例えば、製造環境において、プログラマブルロジックコントローラ（PLC：Programmable Lo

10

20

30

40

50

gic Controller) は、エリアコントローラおよびエッジノードと通信し、これらがデータを、より集中化された IT システムに転送する。コンシューマ環境では、データは、ローカルに処理される多数の位置センサから、即座の自動決定が行われる論理的な場所に集められ、コンシューマの体験を最適化するリモートシステムに転送してもよい。このような多様な分散型アプリケーションは、中央サイトに直接報告するデバイスから、階層的構造にされた自動化システム、組織またはビジネスの集合体を改善するために協調する連合ピア (federated peers) までに及ぶ、同様に幅広い分散型トポロジのセットのサポートを必要とする。

【0033】

単純なアーキテクチャは、センサが中央サイトに報告する。例えば、携帯電話からセンサデータを収集して、そのデータをクラウドサービスに報告するシステムは、集中型アーキテクチャの例を表す。

10

【0034】

より高度なアーキテクチャは、追加の処理および接続性レベルを含む。階層型システムは、より複雑で、多くの既存の物理的および組織的な構造体に似ている。例えば、ローカルコントローラに報告するセンサからなり、そのローカルコントローラがプラント全体のコントローラに報告し、それが部門本部に報告し、それが企業本社に報告する、産業用 IoT システムは、木トポロジ (tree topology) を表す。これらのシステムは、集中型および非集中型両方の監視および制御を提供する。このようなシステムは、リアルタイムまたはほぼリアルタイムの状況において応答性が高い。例えば、データを収集し、データを企業本社に伝送し、企業本社のシステムに機械の次のアクションを判断させることによって、工場設備をリアルタイムで制御することは最適ではないことがある。ローカルコントローラでこのような分析を行い、状況と取るアクションとを工場全体のコントローラに、その後地域および企業本社に報告するだけだと、より効果的となりうる。応答時間の迅速化、可用性の改善およびローカル制御により、状況評価、協調的な意思決定、および応答処理を階層型トポロジに分散させることは、すべてを本社に移動して、すべての決定を集中的に行うよりも効率的である。

20

【0035】

階層型リアルタイムのイベント駆動型ビジネスアプリケーションの別の例は、センサおよび制御点の集合体のためのローカルプロセッサとして機能するエッジノードを使用することであり、エッジノードは、さらに、より集中化されたシステムと対話する。

30

【0036】

高度な分散型リアルタイムのイベント駆動型ビジネスアプリケーションの例は、ピアツーピアシステムであり、そこでピアは、個別の組織によって管理される。例えば、電力需要応答システムでは、システム全体は電力会社が管理するセンサと電力需要家 (utility customers) が管理するセンサとからなるが、システムの制御は、電力会社およびその需要家に分散される。リアルタイムの需要応答を提供するために、電力会社システムと需要家システムとは協調しなければならない。これは、各システムがローカルな決定を行い、ローカルな状況とローカルな決定との両方を相手側に伝送し、さらに互いからのフィードバックに基づいてそれぞれのリアルタイムの行動を修正することに合意することによって実現される。

40

【0037】

サービスとしてのプラットフォーム (PLATFORM-AS-A-SERVICE : PaaS)

図 1 は、説明される技術の例示的な実施形態をデプロイしてもよいサービスとしてのプラットフォーム (PaaS) 100 の高レベルの機能性を示すブロック図である。サービスとしてのプラットフォーム (PaaS) 100 は、リアルタイムのビジネスアプリケーションの開発、デプロイメントおよび動作をサポートするように設計され、構築される。具体的には、サービスとしてのプラットフォーム (PaaS) 100 は、開発者ポータル 102 を含み、それを使用して開発者は、イベント駆動型アプリケーション 104 を開発し、これをさらに分散型ランタイムノード 106 にデプロイする。システムモニタ 108

50

は、分散型ランタイムノード (distributed run-time nodes) 106 上のイベント駆動型アプリケーション 104 の動作を監視し、開発者がイベント駆動型アプリケーション 104 を進化させることができるように、開発者ポータル 102 にフィードバックをする。

【0038】

イベント駆動型アプリケーション 104 は、イベント駆動されてもよい (例えば、データを格納して、最新のステータスチェックを行うのではなく、イベントに対して瞬時に反応する)。サービスとしてのプラットフォーム (PaaS) 100 はさらに、リアクティブフレームワーク (Reactive framework) に実装してもよいので、非同期のノンブロッキングプラットフォーム (non-blocking platform) を提供することにより、リアルタイムの機能性をサポートする。高度に分散され大規模な環境でのイベントストリーム (例えば、もののインターネット (IoT: Internet of Things) 環境からイベントを受信する場合) は、従来の三層アーキテクチャから、イベントベースのモデルに移行する技術的な動機になる。

【0039】

サービスとしてのプラットフォーム (PaaS) 100 はさらに、数多くのイベントを提示するイベント駆動型アプリケーション 104 の設計およびランタイムをサポートする。このために、サービスとしてのプラットフォーム (PaaS) 100 は、分散型環境に大量の数の分散型ランタイムノード 106 のトポロジを使用可能にする。分散型ランタイムノード 106 は、追加の処理能力を提供するために、水平にピア接続され (peered horizontally) てもよい。収集されるデータ (または生成されるイベント) の量が中央プロセッサへのアップロードの限界を超える場合、または低レイテンシが要求される場合、分散型ランタイムノード 106 は、トポロジのエッジのデータの近くに処理を移動させるために、木構造で配置されてもよい。

【0040】

さらに、分散型ランタイムノード 106 は、ミッションクリティカルな可用性 (mission-critical availability) を確保するために水平にクラスタ化されてもよい。

イベント駆動型アプリケーション 104 は、イベントベースのアーキテクチャ、およびリアクティブプログラミングの利点をもたらす一方で、開発者ポータル 102 は、「ローコード (low-code)」開発ツールの提供により JavaScript および SQL を理解するだけでよい。開発ツールは、イベント駆動型アプリケーション 104 のより複雑な要素のための高生産性かつ高水準の、スクリプティングがビジュアル開発に向かない構成要素のビジュアル宣言をサポートする。具体的には、開発者ポータル 102 は、ルール、タイプ、ソース、協調、トピックおよび設定のためのビジュアルエディタと、ルールおよびプロシージャのためのスクリプトエディタと、既存のスキルを活用するために SQL および JavaScript に基づいてドメイン固有言語 (domain-specific language: DSL) とを提供してもよい。加えて、開発者ポータル 102 は、ルール及びプロシージャデバッガ (rule and procedure debugger) による検査能力、トレーシングおよびロギングファシリティ (tracing and logging facilities)、リアルタイムのサブスクリプションサポートおよびデータ視覚化、合成データ生成器、および増分デプロイメントを提供する。さらに、開発者ポータル 102 は、分散型構成によるデプロイメント (例えば、クラウド、プライベートクラウド、オンプレミス、ハイブリッドおよびエッジ) と、ビジュアルデプロイメントツール (例えば、以下に説明されるようなイベントバインディング (event binding) を可能にするイベントバインディングツール 110) とをサポートする。

【0041】

イベント駆動型アプリケーション

図 2 は、いくつかの例示的な実施形態による、サービスとしてのプラットフォーム (PaaS) 100 のイベント駆動型アプリケーション 104 のアーキテクチャ 200 に関するさらなる詳細を示すブロック図である。

【0042】

サービスとしてのプラットフォーム (PaaS) 100 は、以下のものからなる高性能

分散型リアルタイムのイベント駆動型ビジネスアプリケーション（例えば、イベント駆動型アプリケーション 104）の開発、デプロイメントおよび作動を行うプラットフォームサポートを提供する。

1. データ取得：IoTおよびエンタープライズソースからデータを取得し、データをフィルタリングして、それを自動決定エンジンに利用できるようにする技術。
2. イベントおよび状況分析：データをリアルタイムで分析し、その結果に基づいて決定を行う決定エンジン。
3. アクション：制御情報をデバイスに送信し、自動化ソリューションによって行うその後のアクションに関する決定または推奨を外部システムおよびユーザに通知するための技術。複雑な状況に対する最適な応答を引き出すために、自動化システムと担当者との協調を管理する技術。

【0043】

このために、図2は、イベント駆動型アプリケーション104が、データアダプタ（data adapters）202と制御アダプタ204を含むいくつかのアダプタ（adapters）を含むことを示す。イベント駆動型アプリケーション104は、多数のルール、具体的にはデータインGEST・エンリッチ化ルール（data ingestion and enrichment rules）206、状況識別ルール210および協調ルール212も含む。

【0044】

データインGEST・エンリッチ化ルール206は、データアダプタ202が受信するデータのインGESTおよびエンリッチ化を担う。データアダプタ202およびデータインGEST・エンリッチ化ルール206は、データ取得サブシステムの一部を形成し、いくつかのエンタープライズシステム、公開データソース、ソーシャルデータソース（例えば、メッセージングシステム、またはRESTインターフェースを備える任意のシステム）との統合を可能にする。データインGEST・エンリッチ化ルール206は、データアダプタ202が受信するデータのインGESTおよびエンリッチ化を担う。

【0045】

大まかにいうと、データ取得サブシステムは、例えば、REST、MQTTおよびAMQPなどの標準的なプロトコルを使用することによって、幅広いデータソース群からデータを取得する。データソースは、例えば、IoTデバイスおよびエンタープライズシステムを含んでもよく、これらは、センサから流れ、センサデータを適切なコンテキストに配置するデータを評価するために必要なコンテキストを保持する。例えば、イベント駆動型アプリケーションが顧客のロケーションを追跡することによって顧客を支援している場合、顧客のプロファイル情報を取得し、ユーザをその現在のロケーションで支援する機会を評価するには、顧客リソース管理（CRM：Customer Resource Management）システム内の情報へのアクセスが必要となりうる。このため、アプリケーションの一部としての既存システムの統合が非常に重視される。サービスとしてのプラットフォーム（PaaS）100は、既存のエンタープライズシステムをリアルタイムのイベント駆動型ビジネスアプリケーションに組み込むことを可能にする広範囲の宣言型統合（declarative integrations）を提供する。

【0046】

サービスとしてのプラットフォーム（PaaS）100は、以下のことをサポートしてもよい。

- ・プッシュおよびプル（push and pull）の両モデル
- ・同期モデルおよび非同期モデル
- ・RPC（Remote Procedure Call：遠隔手続き呼出し）、ならびに蓄積交換式メッセージングシステム
- ・ソースは、文書化された指定フォーマットをマッピングしてデータを送信する選択をしてもよく、または100にネイティブソースフォーマットを受け入れさせて、フィルタリングシステムを使用してそれを内部処理用に適切なフォーマットに変換する選択をすることができる。

10

20

30

40

50

【0047】

これらの能力を備えるデータ取得サブシステムは、サービスとしてのプラットフォーム（PaaS）100のメッセージングモデルにソースをマッチングさせることを要するのではなく、対話モデルとソースのメッセージプロトコルとをマッチングするだけでソース統合（source integration）を行う。

【0048】

また、サービスとしてのプラットフォーム（PaaS）100は、外部システムにデータソースとの直接的な通信を許さないファイヤウォール（例えば、図6に示されるファイヤウォール624）の背後でホストされるデータを管理するためのモデルをサポートする。

【0049】

データ取得サブシステムの柔軟な性質により、ファイヤウォール（例えば、図6に示されるファイヤウォール624）を通して配信することのできない外部要求に対する応答をソースに要求するのではなく、当該ソースがその裁量でデータを提供することが可能になる。

【0050】

ピアノード内のデータにアクセスするには、サービスとしてのプラットフォーム（PaaS）100にユーザ指定のクレデンシャルを使用するよう要求することによって、セキュリティを維持してもよい。したがって、すべてのノードは、どのピアノードがローカルノードへのアクセスが許可されているかを判定するにあたり完全な制御を有する。

【0051】

イベントおよび状況の識別および分析は、状況識別ルール210によって行う。具体的には、状況識別ルール210は、単純な設定および複雑な設定の両方で、ストリーミングデータを処理してもよい。

・複数のストリームからのデータを、状況分析を助けるために相関付けることができる。開発者は、SQLから導出される単純なドメイン固有言語を使用して、あるストリームで検出されたイベントを別のストリームのイベントの前か後にこななければならない、または両方のイベントがどちらの順序で起こっても特定の時間枠内に生じなければならないと指定する。イベントが起こらない場合でも、一般的なエラーインジケータを単純なやり方で指定することができる。複雑な条件の指定を単純にするイベント制約をどのレベルにも構成することができる。例えば、自動化システムは、機械的デバイスの2つのセンサストリームを監視してもよく、第1のストリームが速度を報告し、第2のものが位置を報告する。自動化システムがデバイスに停止要求を送る場合、第1センサで読み取られるデバイスの速度がゼロになり、速度ゼロの読み取りが確認されたら、デバイスの位置は変わらないままであることが確認されると予想する。速度ゼロになった「後」の位置変化が報告された場合、警告が発せられる。また、速度ゼロが報告されてから30秒以内に位置が報告され「なけれ」ば、デバイス制御システムの故障の可能性を示す警告が発せられる。

・あるストリーミングデータは、即座に処理されるか、または短時間のみ保持されて、時系列構造を促進する一方、他のデータは、より長期間維持されなければならない長期的な時系列または履歴データを表してもよい。100は、連続データおよび集合データをその一時的および永続的形式の両方で表すために使用される抽象化を一元化することによって、一時的および永続的データの両方の使用を単純にする。

・データは、ルールの個別的な集合体により、または機械学習システムによって生成されるアルゴリズムにより分析され、その後アプリケーションに統合される。

・データをSQLベースのドメイン固有言語を使用して分散型トポロジの他のノードに転送し、分散型環境全体でリアルタイム処理を容易にサポートするための完全なサービスセットが利用できる。

【0052】

自動化および協調は、制御アダプタ204および協調ルール212によってサポートされる。

協調ルール212は、人間のユーザとサービスとしてのプラットフォーム（PaaS）

10

20

30

40

50

100の構成要素との間に、人間と機械との協調を実施するために使用される。協調ルール212は、人間のユーザとサービスとしてのプラットフォーム(PaaS)100内の機械とが、状況に応じて、できるだけ独立してまたは協調的に働き、互いの要求を調整することを可能にすることを図る(例えば、システムが反応しているときに人間のユーザが動作(operations)を駆動する(drives)、またはユーザが反応しているときにシステムが動作を駆動する)。

【0053】

アクションは、システムの内部状態に直接適用してもよい。アクションをREST、MQTT、AMQPその他などの標準的な統合、またはカスタム統合を使用して外部デバイスまたはエッジノードに配信するソース統合(例えば、制御アダプタ204)を使用して、アクションを外部デバイスに適用してもよい。

10

【0054】

サービスとしてのプラットフォーム(PaaS)100は、アプリケーションとそのユーザとの間の協調を必要とするアクションまたは応答を作成するモデルを提供する。協調モデルは、開発者ポータル102のグラフィカルエディタを使用して、高レベルの協調パターンを構成することにより、協調の開発をサポートする。例えば、協調ルール212は、以下を含む多数の協調パターンをサポートしてもよい。

・通知-SMS、EMAIL、プッシュ通知およびメッセージングシステムを介して通知および応答を処理する。

・アサイン-タスクへのユーザのアサインをネゴシエートする。

20

・ロケーション追跡-ユーザが指定目的地に到着するとき、およびその目的地に向かって移動中のユーザの現在のロケーションを知るタスクを大幅に簡素化する。

・会話-サードパーティメッセージングシステムによりユーザ間の会話を媒介する。

・エスカレーション-タスク完了の重大な遅延に応答する。

【0055】

サービスとしてのプラットフォーム(PaaS)100は、協調的な意思決定プロセス全体に人を統合しやすくするために使用することのできるモバイルクライアントもサポートする。クライアントは、自然で効率的な対話をサポートするように設計される。ユーザには、注目する必要のある状況が自動的に通知され、各通知のカスタムインターフェースが、ユーザに必要な情報を供給する。ユーザは、携帯機器のデータキャプチャ機能(映像、写真、音声、ロケーション、加速度、自然言語処理による音声、および従来のテキスト入力)を使用することによって応答することができる。

30

【0056】

多くのシステムは、アプリケーションの分散性を明確にプログラミングし、設定し、デプロイすることを強いるが、サービスとしてのプラットフォーム(PaaS)100は、アプリケーションの論理的な定義をその物理的なデプロイメントから分離することによって、これらの動作を簡素化する。開発者ポータル102を使用して、開発者は、アプリケーションをそれが1台のシステムで動作するものであるかのように定義することができる一方で、アプリケーションの構成要素は、バイルール(vail rules)214を使用して自動的にノードにプロビジョニングされる。ランタイムに、サービスとしてのプラットフォーム(PaaS)100の分散型ランタイムノード106は、一緒に動作して、分散型コンピューティング環境216内で1つのリアルタイムビジネスアプリケーションとして機能し、そのアプリケーションに関連するイベントは、イベントブローカー208によって処理される。

40

【0057】

トポロジ

サービスとしてのプラットフォーム(PaaS)100は、分散型および連合型のトポロジの一般的モデルをサポートする。分散型アプリケーション(例えば、イベント駆動型アプリケーション104)は、各ノードが設備(installation)を表す2以上のノードからなってもよい。設備は、1つのサービスインスタンスまたはサービスインスタンスのク

50

ラストを含むことができる。設備がメッセージ交換を望む少なくとも1つの「ピア」ノードを設備が宣言すると、設備は、分散型トポロジにアセンブルされる。

【0058】

設備は、デフォルトでは、独立して管理されと考えられる。別のノードBをピアとして宣言するノードAは、ノードBにアクセスするのにクレデンシャル (credentials) を有していなければならない。このように、サービスとしてのプラットフォーム (PaaS) 100は、自然に連合される。ノードがピアノードで所望の動作を実行する十分な権利を付与されていた場合にのみ、ノードは、別のノードとメッセージを交換できるからである。ピアリング (Peering) は、対称性である。ノードBがノードAとメッセージの交換を望む場合、ノードBは、ノードAをピアとしてプロビジョニングし、ノードAにアクセスする十分な権利をもたなければならない。

10

【0059】

ピアリング関係は、任意の2つのノード間で定義することができるため、サービスとしてのプラットフォーム (PaaS) 100は、任意の分散型トポロジをサポートすることができる。また、認証および許可は、各ノードで独立して管理されるため、トポロジは暗黙的に連合される。

【0060】

一定の使用パターンが、分散型システム内の全ノードを1つの当局によって管理するトポロジを必要としてもよい (または優先してもよい)。このようなシステムは、星型および木型のトポロジに組織化されてもよい。

20

- ・星型—1つの親ノードと任意の数の子ノードとからなる。
- ・木型—根ノードと任意の数の子ノードとからなり、各子ノードが任意の数の子ノードの親として機能してもよい。

【0061】

デプロイされたシステムがより協調的になってくるにつれ、より全体的に連合したピアツウピアネットワークが構築される。このようなネットワークトポロジでは、任意のノードは任意の他のノードとピアリングしてもよいので、ノード間の接続を表す一般グラフ構造 (general graph structure) になる。ネットワークモデルは、非常に複雑になりがちである。グラフのサイクルが起こりうるが、サイクルは、グラフ内の2以上のノードに作用する任意の機能によって扱われなければならないからである。

30

【0062】

また、各ノードは、個別のクレデンシャルを要する独立システムを表すため、サービスとしてのプラットフォーム (PaaS) 100は、協調する組織間の連合に当然一般化する。

【0063】

デプロイメント (DEPLOYMENT)

図3は、一例示的な実施形態による、イベント駆動型アプリケーション104のデプロイメント300を示す図表示である。具体的には、サービスとしてのプラットフォーム (PaaS) 100はデプロイメントマネージャ304を含み、これが、分散型コンピューティング環境216などのターゲット環境へのイベント駆動型アプリケーション104のデプロイメントを操作管理する。図3は、デプロイメントマネージャ304が動作しているノードから直接的または間接的のいずれかで到達可能な多数のノードからなるものとして、分散型コンピューティング環境216を示している。分散型コンピューティング環境216内の物理ノードは、ノードセットに組織化してもよく、そこで、ノードセットについて確立された基準を満たす記述的特性を有することに基づいて、ノードは、特定のノードセットのメンバーとなる。

40

【0064】

分散型コンピューティング環境216内の各ノードは、特定のデバイスまたは構成要素に関連付けられている計算リソースとしてもよい。したがって、分散型コンピューティング環境216内のノードのネットワークを使用して、もののインターネット (IoT) を

50

実装してもよく、その場合、イベント駆動型アプリケーション104は、IoTアプリケーションを備えてもよい。例えば、図3は、対応するカメラ324、カメラ328およびカメラ326にそれぞれ関連付けられたノード312、ノード308およびノード320（これらは、特定のノードセットまたはパーティションを構成してもよい）を示す。同様に、ノード316、ノード314およびノード310は、これらもやはりノード318とともに特定のパーティションを構成してもよく、各RFIDタグ332、RFIDタグ336およびRFIDタグ334に関連付けられている。ノード322は、同様にマイクロホン（microphone）330に関連付けられる。

【0065】

デプロイメントマネージャ304は、特定のノードまたはノードセットにイベント駆動型アプリケーション104の構成要素（またはアーチファクト（artifact））の設定（configurations）338をデプロイする。1つの設定は、1つのノードセットにデプロイするための構成要素のマニフェストを含む。設定338の各々は、対応するパーティション、および特定のパーティションにデプロイされるべきプロジェクトアーチファクトのセット（イベント駆動型アプリケーション104の構成要素を含む）を定義してもよい。一実施例において、パーティションは、特定の設定内で特定されるプロジェクトアーチファクトをデプロイするノードセットを論理的に表す。パーティションは、ターゲット環境（例えば、分散型コンピューティング環境216）内のノードセットから選択される、適格なノードの属性に対する設定の制約によって定義される。設定338は、（図4を参照して以下説明されるように）1つまたは複数のプロジェクト内に含まれてもよい。設定338のセットは、各それぞれのプロジェクト内で定義され、プロジェクトのアーチファクトをデプロイするべきパーティションのセットを定義する。

【0066】

デプロイメントマネージャ304は、環境340を定義するためにも使用され、各環境は、特定の環境内に含まれるノードのリストからなる（例えば、分散型コンピューティング環境216内のノード302～ノード322）。プロジェクトがデプロイメントマネージャ304によって環境にデプロイされるとき、環境内の各ノードは、1つまたは複数のパーティション（例えば、ノードの論理セット）に割り当てられる。各パーティションにアサインされるプロジェクトアーチファクトは、さらに、対応するパーティションの適格なメンバーであるノードにデプロイされる。環境（例えば、分散型コンピューティング環境216）にアサインされるノードセットは、デプロイメントマネージャ304が実行している名前空間（namespace）内に定義されるノードのサブセットとしてもよいことにも留意されたい。

【0067】

図3は、デプロイメントマネージャ304をデプロイメント342にも関連付けられるように示している。デプロイメント342の各々は、プロジェクトとそのプロジェクトで定義される環境との間のバインディング（binding）を定義する。デプロイメントアクションは、デプロイメント342の特定のデプロイメントをその引数として取り、関連付けられているプロジェクトを環境内にデプロイする。

【0068】

デプロイメントパラメータを使用して、特定の環境内のデプロイメントについてプロジェクトアーチファクトをカスタマイズしてもよい。例えば、各パラメータがアーチファクトと、そのアーチファクトのプロパティとを特定してもよい。デプロイメント中、パラメータに関連する値が、当該アーチファクトの定義でそのプロパティのデフォルト値に取って代わる。

【0069】

デプロイメントマネージャ

図4は、いくつかの例示的な実施形態によるデプロイメント環境400のさらなる詳細を示すブロック図である。デプロイメント環境400は、デプロイメントマネージャ304を含み、これは、プロジェクト（例えば、プロジェクト404、プロジェクト408お

10

20

30

40

50

よびプロジェクト 4 1 2) のデプロイメントに集中することにより、開発者の開発タスクを簡素化するように作動する。

【0070】

デプロイメントマネージャ 3 0 4 は、以下の機能を含め、多数の機能を行う。

- ・デフォルトパーティションの自動作成および各パーティションへ開発アーチファクトをアサインすること。
- ・ターゲット環境 3 4 0 で定義されるノードへパーティションを自動的にアサインすること。
- ・ユーザによる設定 3 3 8、環境 3 4 0 およびデプロイメント 3 4 2 のカスタマイズを可能にすること。
- ・プロジェクトをデプロイし、デプロイメント作業のステータスを視覚化すること。
- ・C L I を介して、デプロイメント作業をスクリプトツールおよび自動化ツールに利用できるようにすること。

10

【0071】

デプロイメントマネージャ 3 0 4 は開発者にグラフィカル環境 4 0 2 を与え、そこで開発者は、設定 3 3 8、環境 3 4 0 (例えば、設定を管理するターゲット環境を定義するデータ構造)、環境 3 4 0 (例えば、プロジェクトの設定 3 3 8 がデプロイされるターゲット環境を定義するデータ構造)、およびデプロイメント 3 4 2 (例えば、プロジェクトの設定 3 3 8 と環境 3 4 0 との間のバインディング、およびデプロイメント作業を定義するデータ構造) を管理することができる。特定のプロジェクト (例えば、プロジェクト 4 0 4) は、2 以上の環境 (例えば、デプロイメント環境 4 0 6、テスト環境 4 1 0 および本番環境 4 1 4) にデプロイされてもよく、そのため、複数のこのような環境のタイプにデプロイする必要性を満たす。

20

【0072】

ここで、設定 3 3 8、環境 3 4 0 およびデプロイメント 3 4 2 のそれぞれに関して、さらなる詳細を記載する。

設定 (CONFIGURATIONS)

設定 3 3 8 は、設定の一部であるアーチファクトのマニフェストと、それがデプロイされるパーティションの定義とを含む。設定は、1 つのパーティションとそのパーティションにアサインされるアーチファクトとを定義してもよい。プロジェクトは、1 つまたは複数の設定を含んでもよく、各設定が一意的なパーティションにデプロイされるアーチファクトを記述する。アーチファクトは、2 以上の設定のメンバーとしてもよい。設定は、そのマニフェストに他の設定を含んでもよい。このような場合、子設定 (child configuration) は、パーティションにデプロイされてもよく、さらにその後、子設定は、デプロイメントマネージャを使用して、ターゲットパーティションにアサインされるノードにデプロイされる。

30

【0073】

設定 3 3 8 は、含んでいるプロジェクトのメンバーであるアーチファクトのみを含んでもよい。

設定 3 3 8 に含まれ、パーティションに配置されるアーチファクトは、以下のものを含む。

40

- ・ルール
- ・ソース
- ・タイプ
- ・プロシージャ
- ・トピック
- ・ビジュアルルール
- ・設定
- ・クライアント
- ・R C S 要求

50

- ・ R C S ペイロード
- ・ 協調タイプ

アプリケーションおよび協調は設定 3 3 8 に含まれるが、これらは、より根本的な、分割可能なルールおよびプロシージャから構成される。

【 0 0 7 4 】

環境

環境 3 4 0 の各々は、環境のメンバーであるノードセットを列挙する。ノードは、環境が定義されるプロジェクトのメンバーであってもよい。

【 0 0 7 5 】

環境 3 4 0 は、プロジェクト独立のコンピューティングトポロジを記述し、複数のプロジェクトを 1 つの環境にデプロイすることを可能にする。環境定義内のノードがパーティションにアサインされる必要はないことから、複数の設定およびプロジェクトにわたる環境 3 4 0 の再利用性をさらに改善する。

【 0 0 7 6 】

デプロイメント

デプロイメントは、プロジェクトを環境にデプロイする目的で、開発プロジェクトと環境との間のバインディングを特定する。デプロイメントに適用されるデプロイメント動作の結果は、プロジェクトの設定が環境内で定義されるノードにデプロイされるというアーチファクトである。

【 0 0 7 7 】

デプロイメントパラメータは、プロジェクトおよびその設定を、環境をまたいでポータブルにする (portable)。デプロイメントパラメータは、アーチファクト、そのアーチファクトのプロパティ、およびプロパティにアサインされる値を特定する。デプロイメント時、各デプロイメントパラメータ値は、特定されたアーチファクトのプロパティに代入され、アーチファクトのために当初設定されたデフォルト値に取って代わる。これにより、アーチファクトは、各環境に固有としてもよい物理的リソースにバインドすることが可能となる。

【 0 0 7 8 】

プロジェクト

具体的にプロジェクト (例えば、プロジェクト 4 0 4、プロジェクト 4 0 8、またはプロジェクト 4 1 2) に移ると、各プロジェクトは、名前空間内で定義されるアーチファクトのサブセットを含む。したがって、プロジェクトは、デプロイ可能な機能性の単位を表し、これは、略式にはアプリケーションまたはサービスとして示されることがある。アプリケーションとサービスとの区別に関して、アプリケーションは、そのようなイベントストリームが外部システム (例えば、MQTT キュー) によって発生されるか、またはユーザイベントを介してユーザによって発生されるかに関係なく、インバウンドイベントストリーム (inbound event stream) を作動的に受け入れてもよい。他方で、サービスは、R E S T インターフェースを介して配信される、またはスクリプトによって直接呼び出されることにより配信される呼出し要求に応答する。サービスは、独立してデプロイされ、透明性をもって管理されるため、「マイクロサービス (micro-services)」と考えてもよい。図 4 に図示されるように、プロジェクトは、デプロイメントの単位でもあり、デプロイメントの結果は、インバウンド要求に対して実行するアクティブなアプリケーションまたはサービスとなる。

【 0 0 7 9 】

グラフィカル環境

上で述べたように、デプロイメントマネージャ 3 0 4 は、設定 3 3 8 (設定エディタ 4 1 8 構成要素を使用して)、環境 3 4 0 (環境エディタ 4 2 0 構成要素を使用して)、およびデプロイメント 3 4 2 (デプロイメントエディタ 4 2 2 構成要素を使用して) を視覚化および編集するためのグラフィカル環境 4 0 2 (またはビジュアルエディタ) を開発者に与える。

10

20

30

40

50

【0080】

設定エディタ (configuration editor) 418 は、設定338 / パーティションを、方形エリアが各設定 / パーティションを表す描画パネルに視覚表示する。設定は、パーティションの宣言と考えることができるため、「設定 (configuration)」または「パーティション (partition)」のいずれかの使用は、宣言またはその結果としてのパーティションのどちらを強調するかによって、同様に有効である。

【0081】

各パーティションにアサインされるアーチファクトは、デプロイメントマネージャ304の一部を形成する分割システム416によってそれがアサインされるパーティション内に配置される。設定エディタ418内において、アーチファクトは、プロジェクトのリソースグラフにアーチファクトを表すために使用されるアイコンによって表現され、各アーチファクトアイコンは、そのアーチファクトがアサインされるパーティションを表すエリアに配置される。アーチファクトは、複数のパーティション (または設定338) にアサインしてもよいため、アーチファクトは、視覚化内で何度も表されてもよい。

【0082】

開発者は、アーチファクトを追加することと、異なるパーティションに再アサインしてアーチファクトを設定から取り除くことと、の両方または一方によって、パーティション定義を編集してもよい。追加パーティションを作成することができ、アーチファクトをそれにアサインする。アーチファクトアサインは、分割システム416によって実施される正当性の制約を受ける。設定338を修正する行為は、分割システム416を起動して、開発者のアクションによって要求される任意の再アサインを完了させる。

【0083】

環境エディタ420は、環境のメンバーであるノードを視覚表示する。ユーザは、任意のノードの詳細を見るためにドリルダウンし (drill down) てもよい。ある環境340が、非常に多数のノードを含むこともある。このような場合、すべてのノードを必ずしも図形に列挙する必要はなく、ノードの各クラスを、ノードのクラスのメンバーを定義する制約によって表す。

【0084】

開発者は、これらの制約を使って、名前空間 (例えば、仮想環境の抽象化) で定義され、特定の環境のメンバーとするべきノードを特定する。環境、ノードを特定するための制約をいくつもっていてもよい。ノードは、開発者によって個々に環境にアサインされてもよい。

【0085】

環境の編集は、環境からノードを明確に追加する / 取り除くことによって、または環境に含まれるべきノードセットを特定する制約を追加する / 取り除く / 修正することによって行ってもよい。メンバーシップを規定する制約を開発者が使用している場合、環境エディタ420は、各制約によって特定されるノードセットを開発者が見る仕組みを提供する。

【0086】

環境内のノードの編集は、ノードにドリルダウンすることによって行ってもよい。ノードのプロパティに変更があれば、ノードを異なるノードクラスのメンバーになるようにしてもよい。

【0087】

デプロイメントエディタ422は、デプロイメントで宣言されるバインディングに基づいて、ノードまたはノードクラスへのパーティションのアサインを視覚表示する。デプロイメント環境エディタ420は、各パーティションにアサインされたアーチファクト、およびデプロイメント内の各アーチファクトの定義を修正するデプロイメントパラメータの視覚化も可能にする。開発者は、アーチファクトを選択し、そのアーチファクトにバインドされる環境パラメータを変更することによって、デプロイメントパラメータを編集してもよい。アーチファクトが環境パラメータをサポートしていない場合、編集オプションは使用不可にされる。

10

20

30

40

50

【0088】

図形で見える各ノードは、デプロイメントが進行中か、完了済みか、またはエラーを出したかを示すステータスを表示するため、開発者は、視覚化でデプロイメントのステータスを見ることもできる。開発者は、デプロイメント問題を診断するために、示されたエラーにドリルダウンしてもよい。

【0089】

再び特に図4を参照すると、開発者（または開発チーム）は、デプロイメントマネージャ304を使用して、各ターゲット環境のプロジェクトを定義してもよい（これが今度はノードの集合体を含む）。例えば、プロジェクト404は開発環境406のために開発されてもよく、プロジェクト408はテスト環境410のために開発されてもよく、プロジェクト412は本番環境（production environment）414のために開発されてもよい。このように、デプロイメントマネージャ304は、イベント駆動型アプリケーションの複数のインスタンスの個別環境へのデプロイメントを促しながら、以下の課題に取り組む。アプリケーションは、ターゲット環境（例えば、開発環境406、テスト環境410、および本番環境414）の各々内の異なる物理的リソースにバインドされてもよい。例えば、ターゲットノードは、すべて共通の論理パーティションに属してもよいが、本番ノード（production nodes）は、テストノードとは物理的に別のものとしてもよい。同様に、ソースは、異なる環境内の異なる物理的リソースにバインドされる。図4は、各設定ファイルとしての設定338が、分割システム416から各プロジェクトにデプロイされていることも示す。

【0090】

図5および図6は、ターゲット環境への設定ファイルのデプロイメントに関する各シナリオを示す。特に図5を参照すると、デプロイメント環境500において、ノード502上で動作するデプロイメントマネージャ304は、2つの設定ファイル、すなわち設定ファイル520および設定ファイル522を、ネットワーク524を介してターゲット環境504に出力する。設定ファイル520は、ノード510～ノード516を含むノードセット506の第1設定（またはパーティション）を定義する。設定ファイル522は、ノード516およびノード518を含むノードセット508の第2設定（またはパーティション）を定義する。したがって、特定のノードが複数のノードセット間で共有されてもよいことは認識されるであろう。例えば、ノード516は、ノードセット506およびノードセット508によって共有される。

【0091】

ここで図6を参照すると、設定ファイル618によって表される特定の親設定（parent configuration）がそのマニフェスト内に他の子設定（例えば、設定ファイル620によって表される）を含む別のデプロイメント環境600が示されている。図6は、親設定ファイル618が、ノード602上で実行する第1デプロイメントマネージャ304から、ネットワーク622を介してターゲット環境606にまでデプロイされていることを示す。ターゲット環境606は、ノードセット610およびノードセット614などの複数のノードセットを含む。親設定ファイル618を使用して、ノードセット614のノード604上の別の子デプロイメントマネージャ304をインスタンス化する。こうして、子デプロイメントマネージャ304は、子設定ファイル620によって表される子設定を、ノードセット612およびノードセット616などの別のノードセットを含む別のターゲット環境608にデプロイすることを担う。

【0092】

図7は、いくつかの例示的な実施形態による、分割システム416に関するさらなる詳細を示す図表示である。具体的には、分割システム416は、いくつかのソースコードアナライザ720を含むように示されており、これがイベント駆動型アプリケーション104のソースコードを作動的に分析して、イベント駆動型アプリケーション104の構成要素間の関係を推定するとともに、さらに当該構成要素によりリモート参照（remote references）を検出する。リモート参照は、処理が行われる一組のノードを識別して、指定さ

れた制約を満たす抽象的な組 (abstract set) の計算リソースを表す。いくつかのソースコードアナライザ 720 によって行われるようなソースコード分析に関するさらなる詳細を、図 11 を参照して説明する。

【0093】

イベント駆動型アプリケーション 104 を考えると、当該アプリケーションは、イベント駆動型アプリケーション 104 がデプロイされる分散型コンピューティング環境 216 内のイベントの報告に応答して実行される。このような各イベントは、イベント駆動型アプリケーション 104 にとって関心のある作業の完了を示してもよい。イベントは、通信ネットワーク 722 により多様なソースから特定のイベント駆動型アプリケーション 104 に到着し、例えば、分散型コンピューティング環境 216 内のセンサからの値の読み取りから、イベント駆動型アプリケーション 104 のユーザオペレータによる新たな戦略的イニシアティブ (strategic initiative) の特定まで多岐にわたってもよい。

10

【0094】

図 7 は、イベント駆動型アプリケーション 104 が多数の専用構成要素を含んでもよいことを示しており、主な構成要素クラスには、以下のものが含まれる。

- ・イベント構成要素 704
- ・ソース構成要素 706
- ・ルール構成要素 708
- ・プロシージャ構成要素 710、および
- ・タイプ構成要素 712。

20

【0095】

イベント駆動型アプリケーション 104 は、イベント駆動型アプリケーション 104 の構成要素の各々が分散型コンピューティング環境 216 内の 1 つまたは複数の計算ノード (例えば、ノード 714、ノード 716、またはノード 718) に位置決めされるように、分散型コンピューティング環境 216 内で動作してもよい。

【0096】

ノード 714 ~ ノード 718 として計算ノードのセットを含む分散型コンピューティング環境 216 が図 7 に示されており、各ノードは、コードを実行し、データを格納し、通信ネットワーク 722 経由で他の計算ノードと通信することのできる計算リソースを表す。計算ノードは、パブリッククラウド、プライベートクラウド、データセンターまたはエッジ環境でホストされてもよい。さらに、計算ノードは一意のアドレスを有し、それを使用して他の計算ノードは当該計算ノードと通信することができる。イベント駆動型アプリケーション 104 は、イベント駆動型アプリケーション 104 の構成要素を 1 つまたは複数の利用できる計算ノードに割り当てることにより、分散型コンピューティング環境 216 内で実行するので、実行は、分散型コンピューティング環境 216 内に参加する計算ノードの協調的アクションにより進行する。

30

【0097】

図 8 は、図 7 に図示するルールシステム 800 の図表示であり、設定ファイルと 702 として設定を作動的に出力し、それに従いイベント駆動型アプリケーション 104 の様々な構成要素が分散型コンピューティング環境 216 内の計算ノードにデプロイされる。具体的には、ルールシステム 800 は、イベント駆動型アプリケーション 104 のソースコードを分析してノードセットのアサインを決定するために適用されるルールセットを含む。ノードセットの特定およびこれらのノードセットへのイベント駆動型アプリケーション 104 の構成要素のアサインを、さらに詳細に以下で図 11, 12 を参照して説明する。

40

【0098】

図 8 に図示するように、ルールシステム 800 は、アプリケーション構成要素がアサインされるノードセットとして知られる、計算リソースの抽象集合を定義する多数のルールセットを含む。ルールシステム 800 および関連ルールセットは、アサインルールを分析対象のコードに適用して、ノードセットのアサインを決定する。ルールシステム 800 は、アサインルールを特定のイベントベースのアプリケーションの特化したニーズに対応す

50

るように拡大するためのスキーマも含む。

【0099】

ルールシステム800は、大別して3つのカテゴリのルール、すなわち、一般ルール802、構成要素タイプ固有ルール806、およびカスタムルール804を含む。ルールシステム800は、これらの3つのカテゴリのルールを適用して、イベント駆動型アプリケーション104の構成要素をノードセットにアサインする。これは、すべての構成要素タイプに広く適用できる一般ルール802を再帰的に適用し、次いで1つの構成要素タイプに適用できる構成要素タイプ固有ルール806を適用し、さらにその後にカスタムルール804を適用して行われる。

【0100】

構成要素タイプ固有ルール806は、イベント808、ルール810、タイプ812、プロシージャ814およびソース816に適用可能である。これらのルールの適用に関するさらなる詳細を、図9を参照して以下に述べる。

【0101】

図9は、いくつかの例示的な実施形態による、分散型コンピューティング環境216内でイベント駆動型アプリケーション104を分割、デプロイおよび実行する方法900を示すフローチャートである。

【0102】

方法900と対照的に、分散型のイベントベースのアプリケーションの構築のための一定の方法およびツールは、明確なプログラマーのアクションを介して、計算ノードへの構成要素の手動アサインを必要とする。このようなアサイン作業は、労働集約的で、間違いやすく、多くの場合、計算ノードへのアプリケーション構成要素の最適ではない割当てになる。このようなアサインを自動化するための一定の努力は、オブジェクトの特定に集中しており、これはその後、ノード間の分散型通信の基礎として特定の計算ノードに手動でアサインされるが、各ノードへの追加構成要素の最適な割当てへの集中は限られる。

【0103】

方法900は、いくつかの例示的な実施形態において、分散型コンピューティング環境216の各計算ノードにアサインすることのできる最小コードを正確に決定するためにデプロイされてもよい。このために、方法900は、プログラミング指令が実行されるべき計算ノードのクラスを指定するためのプログラミングディレクティブ (programming directive) を利用してもよい。

【0104】

高レベルでは、方法900は、ノードセットに（および、推定により各ノードセットにアサインされる計算ノードに）イベント駆動型アプリケーション104の構成要素を自動的に割り当てるための分割プロセス916と、その後のイベント駆動型アプリケーション104のデプロイメントプロセス（動作910）と、イベント駆動型アプリケーション104の実行プロセス（動作912）とを含む。分割プロセス916は、プログラミングディレクティブを利用して、プログラミング指令が実行されるべき計算ノードのクラスを指定する。

【0105】

プログラミング記法 (PROGRAMMING NOTATION)

分割システム416で行われるような分割プロセス916は、いくつかの例示的な実施形態において、イベント駆動型アプリケーション104の構成要素が分散型コンピューティング環境216内のノードに割り当てられる記法に基づくので、イベント駆動型アプリケーション104の正当性を確保するとともに、イベント駆動型アプリケーション104のパフォーマンスおよび可用性を最適化する。しかし、アプリケーション開発プロセス中に特定のノードにアサインすることは難しい。その理由は、開発者が、アプリケーション開発中にはターゲット分散型コンピューティング環境216の最終的なトポロジの抽象観 (abstract view) しかもたないことがあるからである。この技術的な課題に対処するため、分割プロセス916および分割システム416は、いくつかの例示的な実施形態では

10

20

30

40

50

、特定のノードセットに関連付けられているプロパティを示す1つまたは複数のノードを表すノードセットを特定することにより、分散型コンピューティング技術の抽象モデルを定義する。開発者が、冷蔵ユニットに関連付けられる計算リソース（例えば、計算ノード）がIoTアプリケーション内に存在することはわかっているが、多数の当該ノード（a number of such nodes）、そのロケーションおよびアイデンティティ（identities）は、デプロイメントプロセス（動作910）の後半まで、さらにイベント駆動型アプリケーション104の構成要素の計算リソースへの割当てが完了してからずっと後になるまで不明なままである場合の例を考えてみる。いくつかの例示的な実施形態による分割システム416および分割プロセス916は、宣言型モデルをサポートすることによりこれらのアサインを抽象化し、そこで開発者は、計算リソースが満たさなければならない論理的な制約を指定することにより、構成要素への参照を指定する。論理的な制約は、その後でノードセットとして正式なものとされ、これは、最終デプロイメントポロジに1つまたは複数のノードを含んでもよい。冷蔵ユニットの例に戻ると、冷蔵ユニットに関連付けられる計算リソースは、例えば、処理制約によって以下のように指定してもよい。

【0106】

PROCESSED BY ManagedEquipment == “refrigeration”（管理対象機器による処理 == 「冷蔵」）

アプリケーション分析

特に分割プロセス916を参照すると、動作1000で、いくつかのソースコードアナライザ720は、検出および構成要素アサイン作業のノードセットの準備のために、イベント駆動型アプリケーション104を分析する。

【0107】

図10は、いくつかのソースコードアナライザ720がアプリケーションのソースコードを分析するために行う動作1000またはさらなる下位動作を示すフローチャートである。具体的には、動作1000は、動作1002から始まり、ここでいくつかのソースコードアナライザ720が特定のイベント駆動型アプリケーション104のソースコードにアクセスする。

【0108】

動作1004で、ソースコードアナライザ720は、アプリケーションのソースコード内で、リモート参照を含むステートメントを特定し、その後、動作1006で、ソースコードアナライザ720は、当該ステートメントで参照される構成要素を特定する。

【0109】

動作1008で、ソースコードアナライザ720は、参照される構成要素をポストするために必要な計算リソースのクラスに対し、論理的な計算リソース制約を決定する。

例えば、ステートメント（statement）：EXECUTE PROCEDURE checkRefrigerationSettings(“temperature”)PROCESSED BY ManagedEquipment == “refrigeration”により、プロシージャ「checkRefrigerationSettings」は、「ManagedEquipment」を表すノードセットに分割される（partitioned）。別の例として、ステートメント：SELECT * FROM Person WHERE age > 21 PROCESSED BY department == “HR”により、タイプ「Person」が複数のHRロケーションをサポートするノードセット内のすべてのノードに分割される。これは、データがすべての「HR」ノードに分散されることを意味する。続いて、SELECT * FROM Person PROCESSED BY department == “HR”等のクエリは、複数のHRロケーションをサポートするノードセット内のすべてのノードのすべてのPersonタイプに対して分散されたクエリを実行して完全な結果セットを作成する。

【0110】

判定動作1010で、ソースコードアナライザ720は、アプリケーションソースコード内に、リモート参照を含む別のステートメントがあるかどうかを判定する。あれば、動作1000は動作1004に戻る。他方で、判定動作1010において、プロセスへのリモート参照を含む別のステートメントがないと判定されれば、動作1000は動作101

10

20

30

40

50

2に進み、ソースコードアナライザ720は、イベント駆動型アプリケーションの構成要素間の依存関係を識別するように作動する。

【0111】

例えば、プロシージャPを参照するルールRがあり、その参照がリモート参照でない場合、プロシージャPはルールRと同じ複数のノードセットに分割されるため、Pに対するRの依存関係(dependency)は、ノードセット内のローカル参照によって満足される。次に、プロシージャPは、リモート参照を含まないステートメントにおいてタイプTを参照する場合がある。これにより、タイプTがRおよびPと同じ複数のノードセットに分割される。

【0112】

動作1014において、ソースコードアナライザ720は、次いで、分析メタデータ(analysis metadata)を生成し、これを使用して分割プロセス916の別の動作を処理する。

【0113】

複数のノードセットの識別(IDENTIFYING NODE SETS) (動作1100)

図9に戻ると、動作1100において、分割システム416のいくつかのソースコードアナライザ720は複数のノードセットの識別に進む。

【0114】

図11は、いくつかのソースコードアナライザ720によって実行され得る動作1100のさらなる複数のサブステップを示すフローチャートである。動作1100は、動作1102において開始され、デフォルトノードセットの割り当ては、イベント駆動型アプリケーションの修飾されていない(unqualified)構成要素のすべての割り当てを保持するように設定されている。

【0115】

動作1104において、ソースコードアナライザ720は、アプリケーションソースコード内の各論理コンピューティングリソースを識別および検査し、判定動作1106および判定動作1112において2つの判定を実行する。

【0116】

具体的には、判定動作1106において、特定の論理コンピューティングリソースについて、その特定の論理コンピューティングリソース制約(specific logical computing resource constraint)がソースコードアナライザ720によって以前に処理されたかどうかについての判定が行われる。判定動作1106において否定的な判定に続いて、動作1108において、ソースコードアナライザ720は、次に特定の論理コンピューティングリソース制約に割り当てられる新しいノードセットを作成する。

【0117】

動作1110において、ソースコードアナライザ720は、参照されるイベント駆動型アプリケーション構成要素の論理コンピューティングリソース制約への依存関係を記録する。言い換えると、(ノードセットにマッピングされる)新しい論理的な制約(logical constraint)が識別されると、ソースコードアナライザ720は、その論理的な制約を用いて参照されるイベント駆動型アプリケーション構成要素のリストを記録するように動作する(これらのイベント駆動型アプリケーション構成要素は、対応するノードセットに分割される)。判定動作1112において、ソースコードアナライザ720は、分析を必要とするさらなる論理コンピューティングリソース制約がアプリケーションソースコード内にあるかどうかを評価する。あれば、動作1100は、判定動作1106に戻る。一方、ない場合、動作1100は、完了した動作1114において終了する。

【0118】

複数のノードセットへの複数の構成要素の割り当て(ASSIGNING COMPONENTS TO NODE SETS) (動作902)

図9に戻ると、動作902において、ルールシステム800は、イベント駆動型アプリケーションの複数の構成要素を、動作1100において識別された複数のノードセットに

10

20

30

40

50

割り当てる。具体的には、ルールシステム 800 は、動作 1200 において、すべての構成要素タイプに適用可能な一組の一般ルール 802 を再帰的に適用し、次いで、動作 904 において、単一の構成要素タイプのみ適用可能なより特殊な一組のルール、すなわち、構成要素タイプ固有ルール 806 を適用することによって、これらの構成要素の割り当てを実行する。最後に、一組のカスタムルール 804 が、動作 914 においてルールシステム 800 によって適用される。これらのカスタムルール 804 は、特定クラスのアプリケーションのためのものであり、そのような特定クラスのアプリケーションのための複数のノードセットにより最適に割り当てられた複数の構成要素に適用され得る。たとえば、プログラマーは、イベント駆動型アプリケーションのすべての電力管理構成要素を、電力消費量の多い機器（たとえば、複数の冷凍ユニット）を管理する複数のノードセットに割り当てたい場合がある。カスタムルール 804 は、そのような追加のパーティショニングセマンティクス（partitioning semantics）の組み込みをサポートする。

10

【0119】

図 12 は、動作 1200 のさらなる複数のサブステップ、すなわち、イベント駆動型アプリケーションの複数の構成要素を複数のノードセットに割り当てるための、すべての構成要素タイプに適用可能な一組の一般ルール 802 の適用を示すフローチャートである。

【0120】

動作 1200 は、動作 1202 で開始し、未割り当て構成要素の識別に続いて、判定動作 1204 において、識別された構成要素が、特定のノードセット（例えば、関連する論理コンピューティングリソース制約によって生成されるノードセット）に関連する論理コンピューティングリソース制約の記法宣言（notational declaration）（例えば、「PROCESSED BY 〈resource〉」節）を含むかどうかについてルールシステム 800 によって判定される。

20

【0121】

判定動作 1204 での肯定的な判定に続いて、ルールシステム 800 は、動作 1206 において、関連する構成要素をその論理コンピューティングリソース制約のために作成されたノードセットに割り当てることに進む。動作 1206 が完了すると、または判定動作 1204 での否定的な判定に続いて、動作 1200 は、判定動作 1208 に進む。判定動作 1208 において、（論理コンピューティングリソースの記法宣言なしの）構成要素が、そのような記法宣言を含む他の構成要素によって参照されるかどうかについての判定が行われる。参照される場合、関連する構成要素は、動作 1210 において参照する構成要素と同じノードセットに割り当てられる。動作 1210 は、すべてのアーチファクトが検査されるまで再帰的に実行される。したがって、構成要素には、その構成要素への参照を含むすべての設定を割り当てることができる。

30

【0122】

次に、判定動作 1208 での否定的な判定の後に割り当てられないままである構成要素は、動作 1212 において、デフォルトの制約のないノードセットに割り当てられる。このデフォルトの制約のないノードセットに割り当てられた各構成要素について、それが参照するすべてのアーチファクトの再帰的トラバース（recursive traversal）が、さらに動作 1212 において実行され、記法宣言（例えば、「PROCESSED BY」節）なしで参照されるアーチファクトも、デフォルトのノードセットに分割される。

40

【0123】

動作 904 での構成要素タイプ固有ルール 806 の適用に戻ると、これらのルールタイプの各々は、以下で別々に考えられる。

複数のイベント 808 イベントは、それが生成されるノードセットに分割される。より具体的には、複数のイベントは、タイプ、ルール、プロシージャ、ソース、および外部要求によって生成され得る。

【0124】

リソース記法宣言（例えば、「PROCESSED BY」節）がプロセス（例えば、配信（publish））要求内で指定されない場合、イベントは、ルールシステム 800 によって、イベ

50

ントを生成するアーチファクトと同じノードセットに分割される。一方、リソース宣言（例えば、「PROCESSED BY」節）がプロセス（例えば、配信）要求内で指定されている場合、イベントは、リソース宣言によって指定されたノードセットに分割される。

【0125】

要求の宛先に関する情報がルールシステム800に知られていないので、外部要求として開始されたプロセス（例えば、配信）要求は、デフォルトのノードセットに分割される。同様に、ユーザ定義の複数のイベントは、イベントに関する他の情報がわからない場合、デフォルトのノードセットに分割される。他の情報は、（PUBLISH（配信）ステートメントなどの）プロセス要求内のイベントへの参照であり得る。このような参照により、上記したイベント分割ルールの完全なセットがイベントに対して評価される。その逆もまた真であり、イベントがノードセットに分割される場合、イベントを生成する構成要素も、同じノードセットに分割される。

【0126】

複数のルール810：複数のルール810は、トリガーするイベント（triggering event）が生成されるのと同じノードセットに分割される。逆もまた真であり、ルールがノードセットに分割されている場合、ルールをトリガーするイベントも同じノードセットに分割され、イベントを生成するアーチファクトも暗黙的に同じノードセットに分割される。

【0127】

複数のタイプ812：構成要素によって参照される複数のタイプ812は、アーチファクトと同じノードセットに配置される。タイプは、異なるノードセットの構成要素によって参照され得、タイプを参照する構成要素を含むすべてのノードセットにプロビジョニングされ得る。構成要素がリソース宣言（例えば、「PROCESSED BY」節）を使用してタイプを参照する場合、タイプは、リソース宣言（例えば、「PROCESSED BY」節）によって定義されたノードセットにプロビジョニングされる。

【0128】

構成要素がリソース宣言なしでタイプを参照する場合、そのタイプは、参照する構成要素と同じノードセットに分割される。

複数のプロシージャ814：複数のプロシージャ814は、プロシージャを参照する構成要素を含むすべてのノードセットに分割される。なお、外部インターフェースを介して呼び出されたプロシージャに利用可能な情報がない場合がある。既知の参照がなく、したがって外部インターフェースを介して呼び出される可能性が高い複数のプロシージャは、デフォルトのノードセットに分割される。

【0129】

構成要素によって参照され、外部インターフェースを介して呼び出されるプロシージャは、複数の外部要求を処理するノードセットに手動で分割されてもよい。

複数のソース816：複数のソース816は、デフォルトでデフォルトのノードセットに分割される。ソースによって生成されたイベントが、明示的に分割されたルールをトリガーする場合、ソースは、トリガーされたルールと同じノードセットに配置される。複数のソース816は、データを取得するための最良の場所を知っている開発者によって手動で分割され得る。この手動配置は、手動配置を固定として使用して自動パーティション化（automatic partitioning）を駆動する。

【0130】

動作902において複数のノードセットへの複数の構成要素の割り当てを完了すると、方法900は動作906に進み、複数の手動オーバーライド（manual overrides）が実行され得る。具体的には、上述した複数のルールがイベント駆動型アプリケーションのすべての構成要素に再帰的に適用されると、分割システム416による分割（partitioning）が完了し、イベント駆動型アプリケーションはデプロイメントの準備が整う（ready for deployment）。この時点で、プログラマーまたはデプロイメントマネージャは、分割された（partitioned）アプリケーションの視覚化を表示することができ、この視覚化は、複数の構成要素および複数の構成要素が割り当てられたノードセットの各々を表示する。

その後、プログラマーは、たとえばドラッグ・アンド・ドロップ・インターフェースを使用して、複数のパーティション定義を手動で変更する。構成要素がノードセットに手動で割り当てられる場合、上記した複数の分割ルールは、分割がルールシステム 800 において具体化された複数の制約を満たすことを保証するために評価され得る。ルールシステム 800 の複数の制約を満たすために必要な複数のノードセットへの複数の構成要素の割り当ての変更は、ルールシステム 800 によって自動的に実行される。

【0131】

次いで、動作 906 を完了すると、方法 900 は、動作 908 において分割プロセスを終了するように進行する。その後、イベント駆動型アプリケーションは、動作 910 においてデプロイメントの準備ができ、動作 912 においてその後の実行の準備ができる。

【0132】

図 13 は、本明細書で説明するデバイスの任意の 1 つまたは複数にインストールすることのできるソフトウェアアーキテクチャ 1304 を示すブロック図 1300 である。ソフトウェアアーキテクチャ 1304 は、プロセッサ 1320、メモリ 1326 および I/O 構成要素 1338 を含む機械 1302 などのハードウェアによってサポートされる。この実施例では、ソフトウェアアーキテクチャ 1304 は、層のスタックとして概念化することができ、各層が特定の機能性を提供する。ソフトウェアアーキテクチャ 1304 は、オペレーティングシステム 1312、ライブラリ 1310、フレームワーク 1308、およびアプリケーション 1306 などの層を含む。作動上、アプリケーション 1306 は、ソフトウェアスタックを通して API コール 1350 を呼び出し、API コール 1350 に応答してメッセージ 1352 を受信する。

【0133】

オペレーティングシステム 1312 は、ハードウェアリソースを管理し、共通のサービスを提供する。オペレーティングシステム 1312 は、例えば、カーネル 1314、サービス 1316 およびドライバ 1322 を含む。カーネル 1314 は、ハードウェアと他のソフトウェア層との間の抽象化層として機能する。例えば、カーネル 1314 は、特に、メモリ管理、プロセッサ管理（例えば、スケジューリング）、構成要素管理、ネットワーキングおよびセキュリティ設定の機能性を提供する。サービス 1316 は、他のソフトウェア層に他の共通のサービスを提供することができる。ドライバ 1322 は、基盤となるハードウェアの制御またはそれとのインターフェース接続を担う。例えば、ドライバ 1322 は、ディスプレイドライバ、カメラドライバ、BLUETOOTH（登録商標）または BLUETOOTH（登録商標）Low Energy ドライバ、フラッシュメモリドライバ、シリアル通信ドライバ（例えば、ユニバーサル・シリアル・バス（USB）ドライバ）、Wi-Fi（登録商標）ドライバ、オーディオドライバ、電力管理ドライバなどを含むことができる。

【0134】

ライブラリ 1310 は、アプリケーション 1306 によって使用される低レベルの共通インフラストラクチャを提供する。ライブラリ 1310 は、メモリ割当て機能、文字列操作機能、数学機能、および同様なものなどの機能を提供するシステムライブラリ 1318（例えば、C 標準ライブラリ）を含むことができる。加えて、ライブラリ 1310 は、メディアライブラリ（例えば、Moving Picture Experts Group - 4（MPEG4）、Advanced Video Coding（H.264 または AVC）、Moving Picture Experts Group Layer - 3（MP3）、Advanced Audio Coding（AAC）、適応マルチレート（AMR: Adaptive Multi-Rate）音声コーデック、ジョイント・フォトグラフィック・エキスパート・グループ（JPEG または JPG）、またはポータブル・ネットワーク・グラフィックス（PNG）などの様々なメディアフォーマットの提示および操作をサポートするライブラリ）、グラフィクスライブラリ（例えば、ディスプレイでグラフィックコンテンツに二次元（2D）および三次元（3D）でレンダリングするために使用される OpenGL フレームワーク）、データベースライブラリ（例えば、様々なリレーショナルデータベ

10

20

30

40

50

ース機能を提供するSQLite)、ウェブライブラリ(例えば、ウェブブラウジング機能性を提供するWebKit)、および同様なものなど、APIライブラリ1324を含むことができる。ライブラリ1310は、アプリケーション1306に多くの他のAPIを提供する、多様な他のライブラリ1328も含むことができる。

【0135】

フレームワーク1308は、アプリケーション1306で使用される高レベルの共通インフラストラクチャを提供する。例えば、フレームワーク1308は、様々なグラフィカルユーザインターフェース(GUI: graphical user interface)機能、高レベルリソース管理、および高レベルロケーションサービスを提供する。フレームワーク1308は、アプリケーション1306によって使用することのできる幅広い範囲の他のAPIを提供することができ、そのうちのいくつかは、特定のオペレーティングシステムまたはプラットフォームに固有のものであってもよい。

【0136】

例示的な実施形態において、アプリケーション1306は、ホームアプリケーション1336、連絡先アプリケーション1330、ブラウザアプリケーション1332、電子書籍リーダーアプリケーション1334、ロケーションアプリケーション1342、メディアアプリケーション1344、メッセージングアプリケーション1346、ゲームアプリケーション1348、およびサードパーティアプリケーション1340などの多種多様な他のアプリケーションを含んでもよい。アプリケーション1306は、プログラムに定義される機能を実行するプログラムである。多様な方法で構造化された1つまたは複数のアプリケーション1306を作成するために、オブジェクト指向型プログラミング言語(例えば、Objective-C、Java、またはC++)または手続き型プログラミング言語(例えば、Cまたはアセンブリ言語)など、様々なプログラミング言語を採用することができる。特定の実施例において、サードパーティアプリケーション1340(例えば、ANDROID(商標)またはIOS(商標)ソフトウェア開発キット(SDK: software development kit)を使用して特定のプラットフォームのベンダ以外のエンティティが開発するアプリケーション)は、IOS(商標)、ANDROID(商標)、WINDOWS(登録商標)Phoneなどのモバイルオペレーティングシステム、または別のモバイルオペレーティングシステム上で動作するモバイルソフトウェアとしてもよい。この例では、サードパーティアプリケーション1340は、本明細書で説明する機能性を促進するために、オペレーティングシステム1312によって提供されるAPIコール1350を呼び出すことができる。

【0137】

図14は、本明細書で述べる任意の1つまたは複数の方法論を機械1400に行わせる命令1408(例えば、ソフトウェア、プログラム、アプリケーション、アプレット、アプリ、または他の実行可能コード)を実行してもよい機械1400の図表示である。例えば、命令1408は機械1400に、本明細書で説明する任意の1つまたは複数の方法を実行させてもよい。命令1408は、汎用のプログラムされていない機械1400を、説明および図示される機能を説明されるように遂行するようにプログラミングされた特定の機械1400に変換する。機械1400は、独立型デバイスとして作動してもよく、または他の機械に結合(例えば、ネットワーク化)してもよい。ネットワーク化されたデプロイメントでは、機械1400は、サーバ・クライアントネットワーク環境内のサーバマシンまたはクライアントマシンの能力で、またはピアツーピア(または分散型)ネットワーク環境内のピアマシンとして作動してもよい。機械1400は、サーバコンピュータ、クライアントコンピュータ、パーソナルコンピュータ(PC)、タブレットコンピュータ、ラップトップコンピュータ、ネットブック、セットトップボックス(STB)、PDA、エンターテインメントメディアシステム、携帯電話、スマートフォン、モバイルデバイス、ウェアラブルデバイス(例えば、スマートウォッチ)、スマート家庭用機器(例えば、スマート家電)、その他スマートデバイス、ウェブアプライアンス、ネットワークルータ、ネットワークスイッチ、ネットワークブリッジ、または機械1400が取るべきアクショ

ンを指定する命令 1 4 0 8 を逐次的にまたはその他の方法で実行することの可能な任意の機械を含んでもよいが、これだけに限定されない。さらに、1 台の機械 1 4 0 0 しか図示していないが、「機械 (machine)」という用語は、本明細書で述べる任意の 1 つまたは複数の方法論を行うために、命令 1 4 0 8 を個別にまたは共同で実行する機械の集合体を含むとも解釈されるものとする。

【0138】

機械 1 4 0 0 は、プロセッサ 1 4 0 2、メモリ 1 4 0 4 および I/O 構成要素 1 4 4 2 を含んでもよく、これらは、バス 1 4 4 4 を介して互いに通信するように構成されてもよい。例示的な実施形態において、プロセッサ 1 4 0 2 (例えば、中央処理装置 (CPU : Central Processing Unit)、縮小命令セットコンピューティング (RISC : Reduced Instruction Set Computing) プロセッサ、複雑な命令セットコンピューティング (CISC : Complex Instruction Set Computing) プロセッサ、グラフィックス・プロセッシング・ユニット (GPU)、デジタル・シグナル・プロセッサ (DSP)、ASIC、無線周波数集積回路 (RFIC : Radio-Frequency Integrated Circuit)、他のプロセッサ、またはこれらの任意の適切な組合せ) は、例えば、命令 1 4 0 8 を実行するプロセッサ 1 4 0 6 とプロセッサ 1 4 1 0 とを含んでもよい。「プロセッサ」という用語は、命令を同時に実行してもよい 2 以上の独立したプロセッサ (「コア」ということもある) を備えてもよいマルチコアプロセッサを含むことが意図されている。図 1 4 は複数のプロセッサ 1 4 0 2 を示しているが、機械 1 4 0 0 は、1 つのコアを有する 1 つのプロセッサ、複数のコアを有する 1 つのプロセッサ (例えば、マルチコアプロセッサ)、1 つのコアを有する複数のプロセッサ、複数のコアを有する複数のプロセッサ、またはこれらの任意の組合せを含んでもよい。

【0139】

メモリ 1 4 0 4 は、メインメモリ 1 4 1 2、静的メモリ 1 4 1 4、および記憶装置 1 4 1 6 を含み、ともにバス 1 4 4 4 を介してプロセッサ 1 4 0 2 にアクセス可能である。メインメモリ 1 4 0 4、静的メモリ 1 4 1 4 および記憶装置 1 4 1 6 は、本明細書で説明する方法論または機能の中の任意の 1 つまたは複数の具現する命令 1 4 0 8 を格納する。命令 1 4 0 8 は、機械 1 4 0 0 によるその実行中、完全にまたは部分的に、メインメモリ 1 4 1 2 の中、静的メモリ 1 4 1 4 の中、記憶装置 1 4 1 6 内の機械可読媒体 1 4 1 8 の中、プロセッサ 1 4 0 2 のうちの少なくとも 1 つ (例えば、プロセッサのキャッシュメモリ) の中、またはこれらの任意の適切な組合せの中にも常駐してもよい。I/O 構成要素 1 4 4 2 は、入力を受信し、出力を提供し、出力を発生し、情報を伝送し、情報を交換し、測定値を取得するなど、多様な構成要素を含んでもよい。特定の機械に含まれる特定の I/O 構成要素 1 4 4 2 は、機械のタイプに依存することになる。例えば、携帯電話などの移動式機械は、タッチ入力デバイスまたはそのような他の入力メカニズムを含むかもしれないが、ヘッドレスサーバマシンは、おそらくこのような入力デバイスを含まないだろう。I/O 構成要素 1 4 4 2 が、図 1 4 に図示していない他の多くの構成要素を含んでもよいことは認識されるであろう。様々な例示的な実施形態において、I/O 構成要素 1 4 4 2 は、出力構成要素 1 4 2 8 および入力構成要素 1 4 3 0 を含んでもよい。出力構成要素 1 4 2 8 は、視覚構成要素 (例えば、プラズマディスプレイパネル (PDP : plasma display panel)、発光ダイオード (LED : light emitting diode) ディスプレイ、液晶ディスプレイ (LCD : liquid crystal display)、プロジェクタ、またはブラウン管 (CRT : cathode ray tube) などのディスプレイ)、音響構成要素 (例えば、スピーカ)、ハプティック構成要素 (haptic components) (例えば、振動モータ、抵抗メカニズム)、その他信号生成器などを含んでもよい。入力構成要素 1 4 3 0 は、英数字入力構成要素 (例えば、キーボード、英数字入力を受信するように構成されたタッチスクリーン、フォトオブティカルキーボード、またはその他英数字入力構成要素)、ポイントベースの入力構成要素 (例えば、マウス、タッチパッド、トラックボール、ジョイスティック、モーションセンサ、または別のポインティング器具)、触覚入力構成要素 (例えば、物理的なボタン、接触もしくは接触ジェスチャの位置と力の両方もしくは一方を提供するタッチスクリ

10

20

30

40

50

ーン、またはその他触覚入力構成要素)、音声入力構成要素(例えば、マイクロホン)、および同様なものを含んでもよい。

【0140】

さらなる例示的な実施形態において、I/O構成要素1442は、多彩な他の構成要素の中でも特に、生体認証構成要素1432、モーション構成要素1434、環境構成要素1436、または位置構成要素1438を含んでもよい。例えば、生体認証構成要素1432は、表現(例えば、手の表現、顔の表情、声の調子、身振り、または視線追跡)を検出し、生体信号(例えば、血圧、心拍数、体温、発汗または脳波)を測定し、人を識別し(例えば、声の識別、網膜の識別、顔の識別、指紋の識別、または脳波図ベースの識別)、および同様なことを行う構成要素を含む。モーション構成要素1434は、加速度センサ構成要素(例えば、加速度計)、重力センサ構成要素、回転センサ構成要素(例えば、ジャイロスコープ)などを含む。環境構成要素1436は、例えば、照明センサ構成要素(例えば、光度計)、温度センサ構成要素(例えば、周囲温度を検出する1つまたは複数の温度計)、湿度センサ構成要素、圧力センサ構成要素(例えば、気圧計)、音響センサ構成要素(例えば、背景騒音を検出する1つまたは複数のマイクロホン)、近接センサ構成要素(例えば、付近の物体を検出する赤外線センサ)、ガスセンサ(例えば、安全のため、または大気中の汚染物質を測定するために、有害ガスの検出濃度に対するガス検出センサ)、または周囲の物理的環境に対応する指標、測定値または信号を提供することのできる他の構成要素を含む。位置構成要素1438は、ロケーションセンサ構成要素(例えば、GPS受信機構成要素)、高度センサ構成要素(例えば、高度計または高度を導ける気圧を検出する気圧計)、方位センサ構成要素(例えば、磁力計)および同様なものを含む。

【0141】

通信は、多様な技術を使用して実施してもよい。I/O構成要素1442は、機械1400をネットワーク1420またはデバイス1422にそれぞれカップリング1424およびカップリング1426を介して結合するように作動可能な通信構成要素1440をさらに含む。例えば、通信構成要素1440は、ネットワークインターフェース構成要素またはネットワーク1420とインターフェース接続するのに適した別のデバイスを含んでもよい。さらなる実施例では、通信構成要素1440は、有線通信構成要素、無線通信構成要素、セルラー通信構成要素、近距離無線通信(NFC: Near Field Communication)構成要素、Bluetooth(登録商標)構成要素(例えば、Bluetooth(登録商標)Low Energy)、Wi-Fi(登録商標)構成要素、および他のモダリティを介して通信を提供する他の通信構成要素を含んでもよい。デバイス1422は、別の機械または多様な周辺デバイス(例えば、USBを介して結合されている周辺デバイス)のいずれであってもよい。

【0142】

また、通信構成要素1440は、識別子を検出してもよく、または識別子を検出するように作動可能な構成要素を含んでもよい。例えば、通信構成要素1440は、無線周波数識別(RFID: Radio Frequency Identification)タグリーダ構成要素、NFCスマートタグ検出構成要素、光学リーダ構成要素(例えば、ユニバーサル製品コード(UPC: Universal Product Code)バーコードなどの一次元のバーコード、クイック・レスポンス(QR)コード、アズテックコード、データマトリックス、データグリフ、マキシコード、PDF417、ウルトラコード、UCC RSS-2Dバーコード、および他の光学コードなどの多次元のバーコードを検出する光学センサ)、または音響検出構成要素(例えば、タグ付けされた音声信号を特定するマイクロホン)を含んでもよい。加えて、インターネットプロトコル(IP)ジオロケーションを介したロケーション、Wi-Fi(登録商標)信号三角測量を介したロケーション、特定のロケーションを示すことのできるNFCビーコン信号の検出を介したロケーションなど、通信構成要素1440を介して多様な情報が導出されてもよい。

【0143】

10

20

30

40

50

様々なメモリ（例えば、メモリ 1 4 0 4、メインメモリ 1 4 1 2、静的メモリ 1 4 1 4、および／またはプロセッサ 1 4 0 2 のメモリ）と、記憶装置 1 4 1 6 と、の両方または一方は、本明細書で説明される方法論または機能の中の任意の 1 つまたは複数を具現する、またはそれによって使用される 1 つまたは複数の命令セットおよびデータ構造（例えば、ソフトウェア）を格納してもよい。これらの命令（例えば、命令 1 4 0 8）は、プロセッサ 1 4 0 2 によって実行されるとき、開示される実施形態を様々な動作に実施させる。

【0 1 4 4】

命令 1 4 0 8 は、伝送媒体を使用して、ネットワークインターフェースデバイス（例えば、通信構成要素 1 4 4 0 に含まれるネットワークインターフェース構成要素）を介して、多数の周知の転送プロトコルのうちの任意のもの（例えば、ハイパーテキスト転送プロトコル（HTTP : hypertext transfer protocol））を使用して、ネットワーク 1 4 2 0 で送受信されてもよい。同様に、命令 1 4 0 8 は、伝送媒体を使用して、カップリング 1 4 2 6（例えば、ピアツーピア結合）を介して、デバイス 1 4 2 2 に送受信されてもよい。

【0 1 4 5】

陳述

1．分散型コンピューティング環境内のイベント駆動型アプリケーションの複数のイベント駆動型アプリケーション構成要素をデプロイする方法であって、

1 つまたは複数のプロセッサを用いて、イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析して複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、

1 つまたは複数のプロセッサを用いて、一組のルールを適用して、

前記自動的に分析することに基づいて、分散型コンピューティング環境内の複数の計算ノードへの複数のイベント駆動型アプリケーション構成要素の割り当てを記録する割り当てデータを生成すること、

分散型コンピューティング環境内で割り当てられた計算ノードでの実行をサポートするために必要な複数のイベント駆動型アプリケーション構成要素の各々の構成要素要件を決定すること、を備える、方法。

【0 1 4 6】

2．複数の計算ノードの各々を複数のノードセットのうちの対応するノードセットに割り当ててを含む、前述の請求項のいずれか 1 つまたは複数に記載の方法。

3．前記割り当てデータを生成することは、複数のノードセットのうちの対応するノードセットの各ノードに、共通の一組のイベントアプリケーション構成要素を割り当ててを含む、前述の請求項のいずれか 1 つまたは複数に記載の方法。

【0 1 4 7】

4．一組のルールは、複数のイベント駆動型アプリケーション構成要素の各々を、イベント駆動型アプリケーションの適切な動作を提供するのに十分である複数のノードセットのうちの最適なノードセットに割り当てるように適用される、前述の請求項のいずれか 1 つまたは複数に記載の方法。

【0 1 4 8】

5．対応するイベント駆動型アプリケーション構成要素が割り当てられる最適なノードセットは、対応するイベント駆動型アプリケーション構成要素に割り当てられる最小コードの決定に基づいて識別される、前述の請求項のいずれか 1 つまたは複数に記載の方法。

【0 1 4 9】

6．前記アプリケーションソースコードの自動的な分析は、複数のイベント駆動型アプリケーション構成要素のうちのイベント駆動型アプリケーション構成要素の各々によるリモート参照の記法宣言を識別することを含む、前述の請求項のいずれか 1 つまたは複数に記載の方法。

【0 1 5 0】

7．リモート参照の記法宣言は、リモート参照の論理コンピューティングリソース制約の宣言を含む、前述の請求項のいずれか 1 つまたは複数に記載の方法。

8. 前記アプリケーションソースコードの分析は、記法宣言を分析して、特定のイベント駆動型アプリケーション構成要素によって指定されるプログラミング指示が実行される計算ノードのクラスを識別すること、を含む、前述の請求項のいずれか1つまたは複数に記載の方法。

【0151】

9. 前記アプリケーションソースコードの分析は、特定のイベント駆動型アプリケーション構成要素の実行を動作的にトリガーするリソースを決定することを含む、前述の請求項のいずれか1つまたは複数に記載の方法。

【0152】

10. 前記アプリケーションソースコードの分析は、特定のイベント駆動型アプリケーション構成要素によって参照されるリソースを決定することを含む、前述の請求項のいずれか1つまたは複数に記載の方法。

【0153】

11. 前記アプリケーションソースコードを分析することは、特定のイベント駆動型アプリケーション構成要素の識別されたコードセグメントの依存関係を識別することを含む、前述の請求項のいずれか1つまたは複数に記載の方法。

【0154】

12. 前記アプリケーションソースコードの自動的な分析は、割り当てデータを生成するために使用される分析メタデータを生成することを含む、前述の請求項のいずれか1つまたは複数に記載の方法。

【0155】

13. コンピューティング装置であって、
プロセッサと、
複数の命令を記憶するメモリと、を備え、
プロセッサによって実行されると、複数の命令は、前記装置を、
1つまたは複数のプロセッサを用いて、イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析して複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、
1つまたは複数のプロセッサを用いて、一組のルールを適用して、
前記自動的に分析することに基づいて、分散型コンピューティング環境内の複数の計算ノードへの複数のイベント駆動型アプリケーション構成要素の割り当てを記録する割り当てデータを生成すること、
分散型コンピューティング環境内で割り当てられた計算ノードでの実行をサポートするために必要な複数のイベント駆動型アプリケーション構成要素の各々の構成要素要件を決定すること、を実行させるように設定する、コンピューティング装置。

【0156】

14. 複数の計算ノードの各々を複数のノードセットのうちの対応するノードセットに割り当てることを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0157】

15. 前記割り当てデータを生成することは、複数のノードセットのうちの対応するノードセットの各ノードに、共通の一組のイベントアプリケーション構成要素を割り当てることを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0158】

16. 一組のルールは、複数のイベント駆動型アプリケーション構成要素の各々を、イベント駆動型アプリケーションの適切な動作を提供するのに十分である複数のノードセットのうちの最適なノードセットに割り当てるように適用される、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0159】

17. 対応するイベント駆動型アプリケーション構成要素が割り当てられる最適なノード

10

20

30

40

50

ドセットは、対応するイベント駆動型アプリケーション構成要素に割り当てられる最小コードの決定に基づいて識別される、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0160】

18. 前記アプリケーションソースコードの自動的な分析は、複数のイベント駆動型アプリケーション構成要素のうちのイベント駆動型アプリケーション構成要素の各々によるリモート参照の記法宣言を識別することを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0161】

19. リモート参照の記法宣言は、リモート参照の論理コンピューティングリソース制約の宣言を含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

10

【0162】

20. 前記アプリケーションソースコードの分析は、記法宣言を分析して、特定のイベント駆動型アプリケーション構成要素によって指定されるプログラミング指示が実行される計算ノードのクラスを識別すること、を含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0163】

21. 前記アプリケーションソースコードの分析は、特定のイベント駆動型アプリケーション構成要素の実行を動作的にトリガーするリソースを決定することを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

20

【0164】

22. 前記アプリケーションソースコードの分析は、特定のイベント駆動型アプリケーション構成要素によって参照されるリソースを決定することを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0165】

23. 前記アプリケーションソースコードを分析することは、特定のイベント駆動型アプリケーション構成要素の識別されたコードセグメントの依存関係を識別することを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0166】

30

24. 前記アプリケーションソースコードの自動的な分析は、割り当てデータを生成するために使用される分析メタデータを生成することを含む、前述の請求項のいずれか1つまたは複数に記載のコンピューティング装置。

【0167】

25. 複数の命令を含む非一時的なコンピュータ可読記憶媒体であって、
複数の命令は、コンピュータによって実行されると、前記コンピュータに、
1つまたは複数のプロセッサを用いて、イベント駆動型アプリケーションのアプリケーションソースコードを自動的に分析して複数のイベント駆動型アプリケーション構成要素間の複数の関係を識別すること、

1つまたは複数のプロセッサを用いて、一組のルールを適用して、
前記自動的に分析することに基づいて、分散型コンピューティング環境内の複数の計算ノードへの複数のイベント駆動型アプリケーション構成要素の割り当てを記録する割り当てデータを生成すること、

40

分散型コンピューティング環境内で割り当てられた計算ノードでの実行をサポートするために必要な複数のイベント駆動型アプリケーション構成要素の各々の構成要素要件を決定すること、を実行させる、非一時的なコンピュータ可読記憶媒体。

【図面】
【図 1】

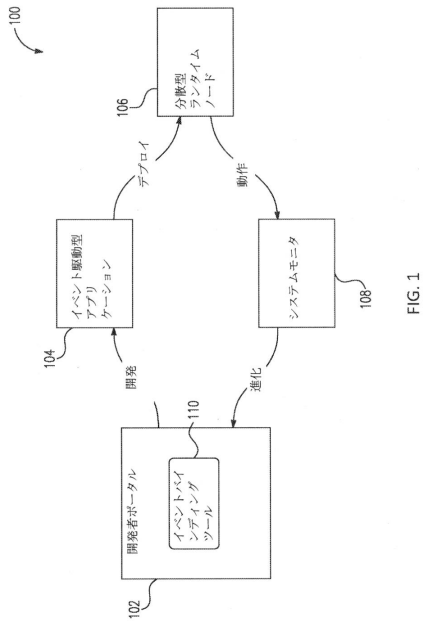


FIG. 1

【図 2】

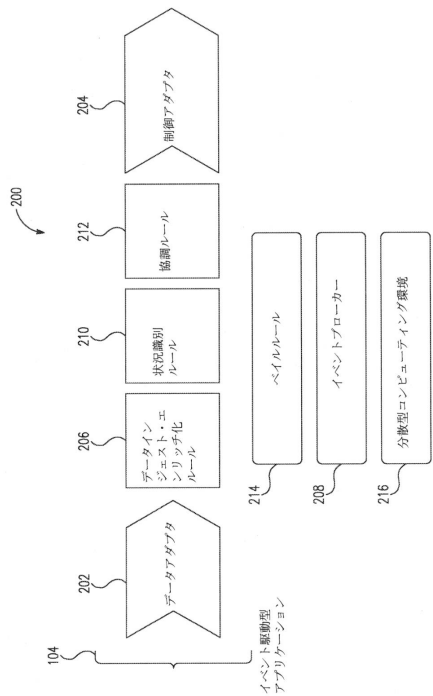


FIG. 2

【図 3】

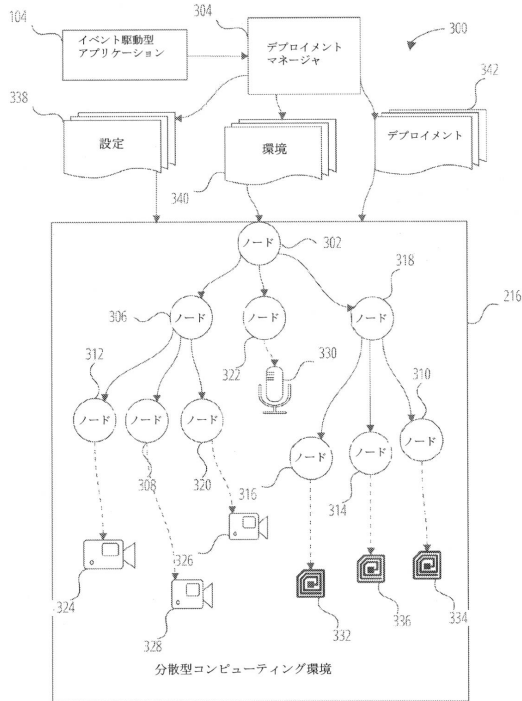


FIG. 3

【図 4】

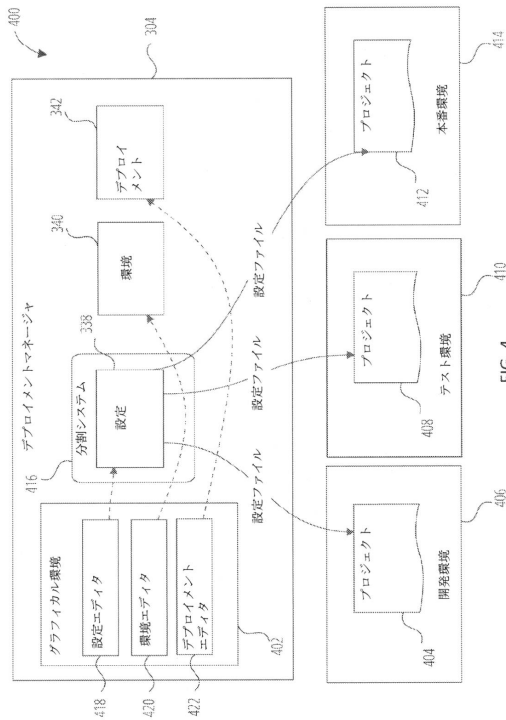


FIG. 4

10

20

30

40

50

【図 5】

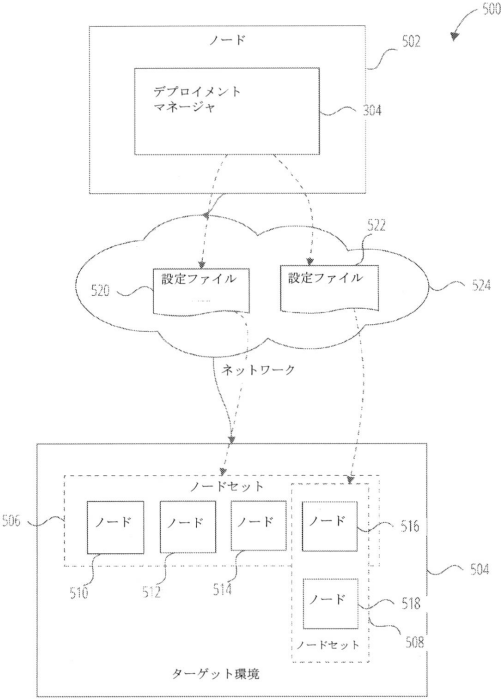


FIG. 5

【図 6】

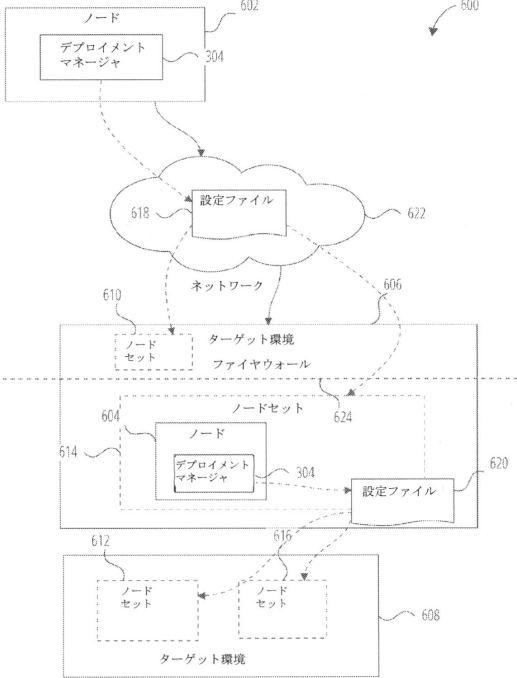


FIG. 6

【図 7】

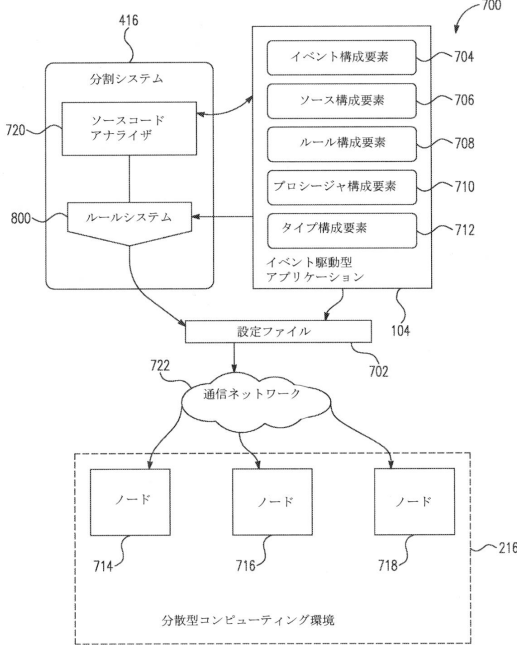


FIG. 7

【図 8】

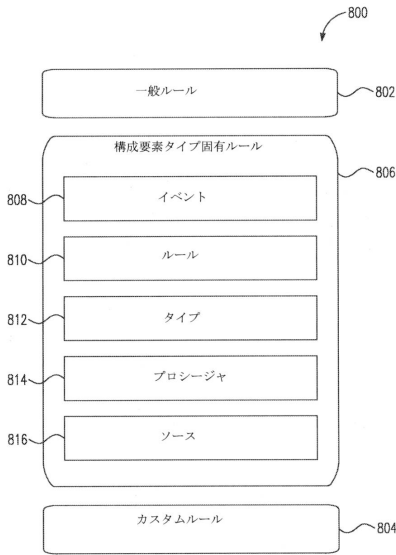


FIG. 8

10

20

30

40

50

【図 9】

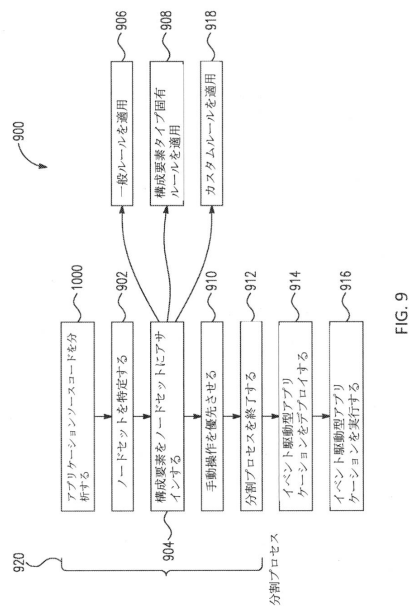


FIG. 9

【図 10】

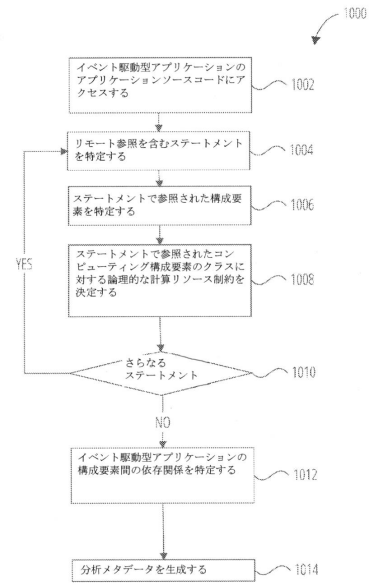


FIG. 10

【図 11】

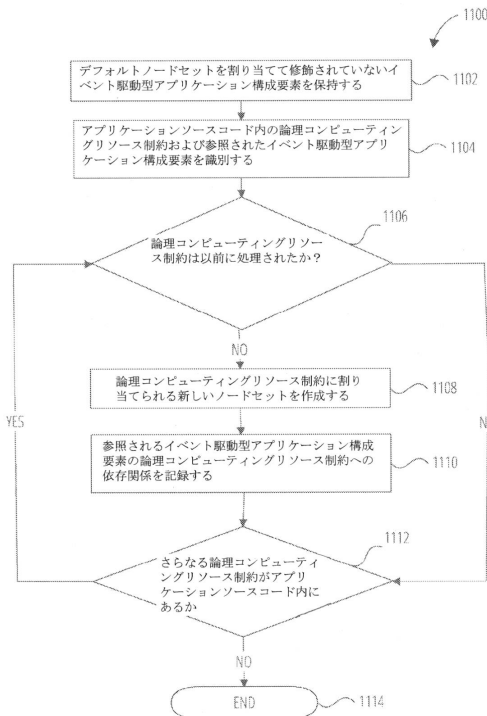


FIG. 11

【図 12】

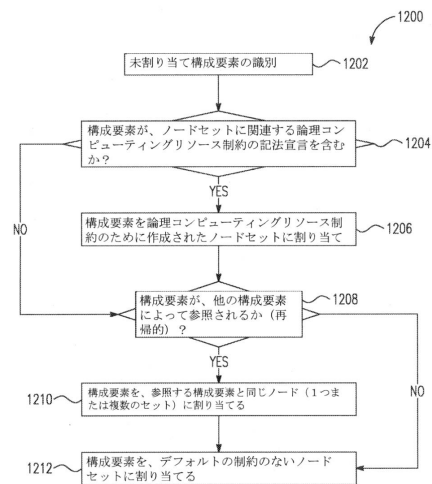


FIG. 12

10

20

30

40

50

【図 13】

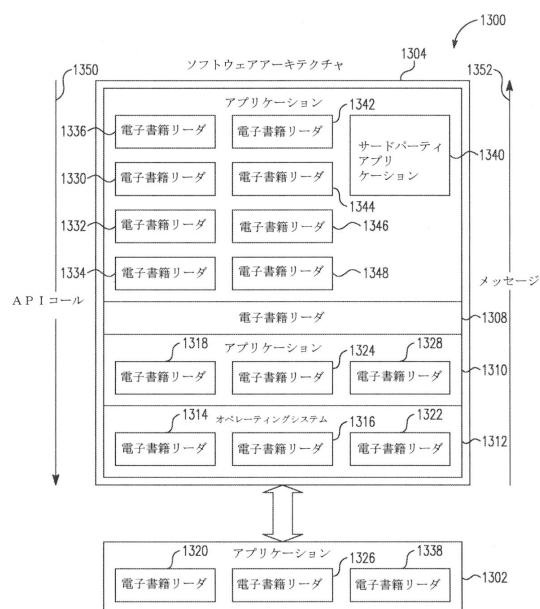


FIG. 13

【図 14】

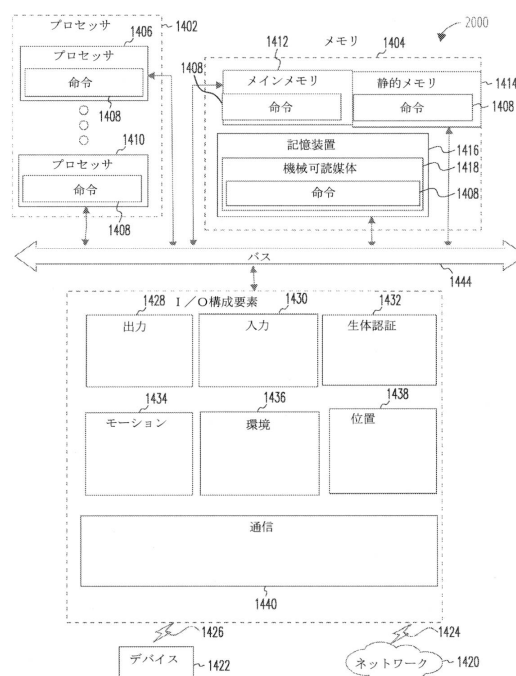


FIG. 14

フロントページの続き

- ニア州 アラモ アラモ グレン ドライブ 2 4 4 5
(72)発明者 チャン、エバン
アメリカ合衆国 9 4 1 3 2 カリフォルニア州 サンフランシスコ ストラトフォード ドライブ
2 3 2
(72)発明者 ラングレー、スティーブ
アメリカ合衆国 9 4 5 3 2 カリフォルニア州 ウォルナット クリーク ブロンソン レーン 8 8 8
審査官 多賀 実
(56)参考文献 特開2009-110311 (JP, A)
特開2008-176364 (JP, A)
米国特許出願公開第2014/0068560 (US, A1)
(58)調査した分野 (Int.Cl., DB名)
G 0 6 F 8 / 0 0 - 8 / 7 7
G 0 6 F 9 / 4 4 - 9 / 5 4
G 1 6 Y 3 0 / 0 0
G 1 6 Y 4 0 / 0 0 - 4 0 / 6 0
H 0 4 L 6 7 / 1 0 - 6 7 / 1 0 9 7
H 0 4 L 6 7 / 1 3 3