

(19)日本国特許庁(JP)

(12)公表特許公報(A)

(11)公表番号
特表2023-518198
(P2023-518198A)

(43)公表日 令和5年4月28日(2023.4.28)

(51)国際特許分類

F I

テーマコード (参考)

G 0 6 F 8/60 (2018.01) G 0 6 F 8/60 5 B 3 7 6

G 0 5 B 19/05 (2006.01) G 0 5 B 19/05 A 5 H 2 2 0

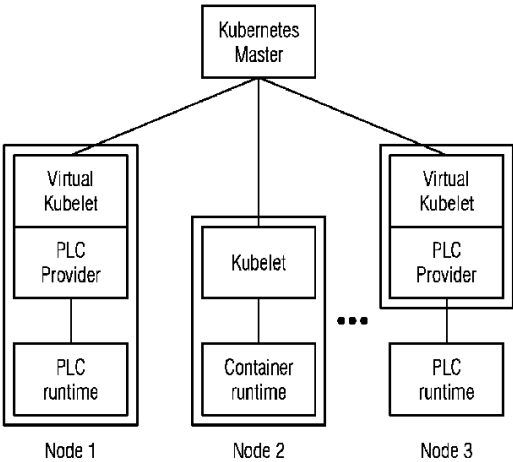
審査請求 有 予備審査請求 有 (全30頁)

(21)出願番号	特願2022-555049(P2022-555049)	(71)出願人	390039413 シーメンス アクチエンゲゼルシャフト Siemens Aktiengesellschaft ドイツ連邦共和国 D-80333 ミュンヘン ヴェアナーフォンシーメンスシュトラッセ 1
(86)(22)出願日	令和3年2月25日(2021.2.25)	(74)代理人	110003317 弁理士法人山口・竹本知的財産事務所
(85)翻訳文提出日	令和4年11月9日(2022.11.9)	(74)代理人	100075166 弁理士 山口 巖
(86)国際出願番号	PCT/EP2021/054633	(74)代理人	100133167 弁理士 山本 浩
(87)国際公開番号	WO2021/185547	(74)代理人	100169627 弁理士 竹本 美奈
(87)国際公開日	令和3年9月23日(2021.9.23)		
(31)優先権主張番号	20164384.8		
(32)優先日	令和2年3月20日(2020.3.20)		
(33)優先権主張国・地域又は機関	欧州特許庁(EP)		
(81)指定国・地域	AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,最終頁に続く		最終頁に続く

(54)【発明の名称】 産業用オートメーションプラットフォームのオートメーションプログラムを管理する方法および装置

(57)【要約】

本発明は、産業用オートメーションプラットフォームのオートメーションプログラムを管理する方法および装置に関し、オートメーションプログラムは、オートメーションプラットフォームへ送信され、オートメーションプログラムの実行は、オートメーションプラットフォーム上で制御される。その際、第1ステップでは、オートメーションプログラムまたはこのオートメーションプログラムへのリファレンスが、Kubernetesマスタから仮想kubele tへ送信され、第2ステップでは、仮想kubele tのプロバイダインタフェースによって、送信されるまたは参照されるオートメーションプログラムが、産業用のオートメーションプラットフォームへ送信され、第3ステップでは、送信されたオートメーションプログラムの実行は、産業用オートメーションプラットフォーム上で制御され、プロバイダインタフェースによって、制御命令が産業用オートメーションプラットフォームへ送信され、産業用オートメーションプラットフォームの肯定応答メッセージが受信され、そして処理されるかまたは制御インスタンスへ、特にK



【特許請求の範囲】

【請求項 1】

産業用のオートメーションプラットフォームのオートメーションプログラムを管理する方法であって、

前記オートメーションプログラムは、前記オートメーションプラットフォームに送信され、前記オートメーションプログラムの実行は、前記オートメーションプラットフォーム上で制御される方法において、

第 1 ステップにて、前記オートメーションプログラムまたは前記オートメーションプログラムへのリファレンスが、K u b e r n e t e s（登録商標）マスタから仮想 k u b e l e t へ、送信され、

10

第 2 ステップにて、前記仮想 k u b e l e t のプロバイダインタフェースによって、送信される又は参照される前記オートメーションプログラムは、産業用の前記オートメーションプラットフォームに送信され、

第 3 ステップにて、送信された前記オートメーションプログラムの実行は、産業用の前記オートメーションプラットフォームで制御され、前記プロバイダインタフェースによって、制御命令が産業用の前記オートメーションプラットフォームに送信され、産業用の前記オートメーションプラットフォームの肯定応答メッセージが受信され、そして処理されるか、または、制御インスタンスへ、特に前記 K u b e r n e t e s マスタへ、転送される、

20

ことを特徴とする方法。

【請求項 2】

請求項 1 に記載の方法において、

前記第 1 ステップにて、前記オートメーションプログラムへの前記リファレンスとして、メモリアドレス、特に、U R L、メモリパス情報、アプリストアのコンテンツに関する仕様、または他のアドレス、が使用されること、

を特徴とする方法。

【請求項 3】

請求項 1 または 2 に記載の方法において、

前記第 2 ステップにて、前記仮想 k u b e l e t へ送信されるかまたはそれに前記リファレンスとして指定される前記オートメーションプログラムは、前記オートメーションプラットフォームへの送信前に、前処理されること、

30

を特徴とする方法。

【請求項 4】

請求項 3 に記載の方法において、

前記第 2 ステップにて、前記オートメーションプログラムは、前記送信前に、コンパイルされること、を特徴とする方法。

【請求項 5】

請求項 3 または 4 に記載の方法において、

前記第 2 ステップにて、送信される前記オートメーションプログラムは、参照される複数のソースから、および／または、前記 K u b e r n e t e s マスタによって送信されるプログラムの一部から、作成されること、

40

を特徴とする方法。

【請求項 6】

請求項 1 ～ 5 の何れか 1 項に記載の方法において、

複数の産業用オートメーションプラットフォームを備える装置で、個別の前記オートメーションプラットフォームの各々に対して、または、同じタイプまたは同様の前記オートメーションプラットフォームのグループに対して、それぞれ固有の前記プロバイダインタフェースが使用され、使用される前記プロバイダインタフェースの各々は、それぞれの前記オートメーションプラットフォームへ送信される前記オートメーションプログラムを、それぞれのターゲットオートメーションプラットフォームのタイプおよび特別なプロパテ

50

ィへの適合のために、実行すること、
を特徴とする方法。

【請求項 7】

請求項 1～6 の何れか 1 項に記載の方法において、

前記仮想 `k u b e l e t` によって、産業用のエンジニアリングシステムまたはその一部が、前記プロバイダインタフェースとして使用され、前記エンジニアリングシステムまたは前記その一部は、前記 `K u b e r n e t e s` マスタから前記仮想 `k u b e l e t` へ送信される命令を用いて、制御されること、
を特徴とする方法。

【請求項 8】

請求項 1～7 の何れか 1 項に記載の方法において、

前記オートメーションプラットフォームのリソースを管理するために、および／または、前記オートメーションプログラムを複数の前記オートメーションプラットフォームのうちの 1 つに割り当てるために、`K u b e r n e t e s` のカスタムリソース定義 (`C R D`) が使用されること、
を特徴とする方法。

【請求項 9】

請求項 1～8 の何れか 1 項に記載の方法において、

前記オートメーションプラットフォームの使用に適合されたスケジューラ (`K u b e r n e t e s C u s t o m S c h e d u l e r`) が、前記オートメーションプラットフォームへの前記オートメーションプログラムへの展開に使用されること、
を特徴とする方法。

【請求項 10】

産業用のオートメーションプラットフォームのオートメーションプログラムを管理する装置であって、

前記オートメーションプログラムまたは前記オートメーションプログラムへのリファレンスを仮想 `K u b e l e t` へ送信する `K u b e r n e t e s` マスタを備え、

前記オートメーションプログラムまたは前記オートメーションプログラムへの前記リファレンスを受信するための、前記オートメーションプラットフォームでの前記オートメーションプログラムのインストールのための命令を受信するための、及び、前記オートメーションプラットフォームでの前記オートメーションプログラムのプロセスを制御するための、前記仮想 `K u b e l e t` を備え、

前記オートメーションプログラムを前記オートメーションプラットフォームへ送信するための、前記オートメーションプログラムのプログラムプロセスを制御するための、及び、前記オートメーションプログラムのプロセスの実行中に、前記オートメーションプラットフォームの肯定応答メッセージを受信及び転送するための、前記仮想 `K u b e l e t` のプロバイダインタフェースを備える、
装置。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、請求項 1 の前提部分に係る産業用オートメーションプラットフォームのオートメーションアプリケーションを管理する方法、および、請求項 10 の前提部分に係る産業用オートメーションプラットフォームのオートメーションアプリケーションを管理する装置、に関する。

【背景技術】

【0002】

オートメーションシステムのオートメーションアプリケーションは、1 つまたは複数の、例えば `I E C 6 1 1 3 1` 規格に準拠しているオートメーションプログラムであって、産業用オートメーションプラットフォームで実行されるオートメーションプログラム、か

10

20

30

40

50

らなる。そのような産業用オートメーションプラットフォームは、例えば、プログラマブルロジックコントローラ（英：Programmable Logic Controller（略してP L C）、独：Speicherprogrammierbare Steuerungen）であってよい。このオートメーションプログラムの作成および管理、例えば、個別のコントローラへのプログラムの割り当て、プログラムのロード／起動／停止、プログラムのアップデート等は、一般に、いわゆるエンジニアリングシステムにおいて、例えばシーメンス株式会社（S i e m e n s A G）のT I Aポータルにおいて、行われる。これらアクティビティは、一般的にエンジニアリングと称され、通常、手動制御で行われる、すなわち、低い自動化レベルで、そして対応的に長い変更時間と高い変更コストで行われる。また、エンジニアリングシステムは、通常、産業用オートメーションプラットフォームにローカル接続する必要もあり、エンジニアリングシステムのバージョンは、使用されるオートメーションプラットフォーム（コントローラ）のそれぞれのバージョンおよびタイプと一致すること等、が必要とされる。

10

【0003】

以下、少なくとも、既に作成されたオートメーションプログラムを目的地又は「ターゲット」へ、つまり産業用オートメーションプラットフォーム（S P S、P L C等）へ、送信するプロセス、オートメーションプログラムのプロセスを開始および終了（スタート／ストップ）するステップ、制御エンティティ（英：control entity、独：Kontrollinstanz）（例えば、H M Iデバイス（H M I：Human Machine Interface））に対するオートメーションプログラムを処理するプロセスの間の産業用オートメーションプラットフォームの肯定応答メッセージ（独：Quittierungsmeldungen、英：acknowledgment messages）を受信及び転送するステップを、まとめて、オートメーションプログラムの「管理」および「制御」と称する。

20

【0004】

I Tベースのモジュラーシステムに対しては、つまり従来のコンピュータ技術によるシステムに対しては、既にしばらく前から、ソフトウェアコンテナ技術が導入されている。特に、複数のコンピューティングノードを備える分散システムにおいては、コンテナベースのプログラムは、いわゆるコンテナオーケストレーション技術（独：Container-Orchestration-Techniken、英：container orchestration technologies）、例えばK u b e r n e t e s（登録商標）（<https://kubernetes.io/de/>）、を用いて、コンピューティングノードに分散され、実行され、監視される。K u b e r n e t e sでは、これらは、中央コンポーネント、いわゆるマスタまたはK u b e r n e t e sマスタ、を介して行われる。これは、実行されるソフトウェアコンポーネントの説明、いわゆるマニフェスト、を取得する。マスタは、適切なノード（独：Knoten、英：Nodes）を選択し、そこにあるソフトウェアコンポーネントを有するコンテナを実行させるが、これは「デプロイメント」と称される。さらに、K u b e r n e t e sマスタは、コンポーネントを監視し、必要に応じて、リカバリするためのアクションを実行する。さらに、ランタイムに、コンポーネントの変更を行うことができる、例えば、アップデートされたバージョンの起動、スケーリング（スループットを増減させるためのインスタンスのスタート／ストップ）等を、行うことができる。

30

40

【0005】

K u b e r n e t e sを使用するために基本的なことは、完全なソフトウェアが1つまたは複数のソフトウェアコンテナ（例えば、D o c k e rコンテナ（<https://www.docker.com/resources/what-container>））に統合されており、このコンテナが、ランタイム環境、いわゆるコンテナランタイムに、コンピュータハードウェア（通常の場合「ノード」と称される）で送信されることである。ここで、「ノード」の構成要素は、いわゆるk u b e l e tであり、当該k u b e l e tはローカルで実行されるソフトウェアであり、当該ソフトウェアは管理を行うK u b e r n e t e sマスタおよびローカルコンテナランタイムと協働し、ソフトウェアを含むコンテナをローカルにダウンロードし、実行し、動作を監視する。

50

【0006】

上述のように、従来、オートメーションプログラムの管理（作成、ロード、スタート、ストップ）は、産業用オートメーションプラットフォーム（産業用コントローラ、SPS、PLC等）にローカル接続される特別なエンジニアリングシステムを用いて、通常の場合大部分が手動（マニュアル）で、行われる。通常の場合、個別のオートメーションプラットフォーム（例えば、PLC（プログラマブルロジックコントローラ））へのオートメーションプログラムの割り当ては、手動で行われる。

【0007】

また、従来のエンジニアリングシステムを使用することは、多くの場合、このエンジニアリングシステムのインスタンスが産業用ネットワークにローカルで存在することと関連付けられており、「遠隔制御」は、特にクラウドベースのアプローチにおいては、遠隔作業場からのローカルエンジニアリングシステムへの専用アクセスを必要とし、これは、システム全体の高コストな構成を必要とする。

【発明の概要】

【発明が解決しようとする課題】

【0008】

従って、本発明の課題は、従来のオートメーションプログラムを高い自動化レベルで管理し、制御することであり、その際、可能な限り確立されており、広く普及しており、簡単に学習可能な方法を使用し、管理される産業用オートメーションプラットフォームに対してのエンジニアリングシステムの依存性を低下させ、エンジニアリングシステムへのローカルな依存性を軽減または除去することである。

【課題を解決するための手段】

【0009】

本発明の課題は、原則的に、Kubernetesマスタとkubeleletとからなる周知のシステムによって、解決され、そのような周知のシステムは、コンテナベースのプログラムをコンテナランタイム環境において管理および制御するように意図されている。本発明によれば、仮想kubelelet (<https://github.com/virtual-kubelet>を参照) が使用され、当該仮想kubeleletは、Kubernetesマスタに対して「本物の」kubeleletのように振る舞う一方、特別なプロバイダインタフェース (<https://virtual-kubelet.io/docs/providers/#provider-interface>を参照) を備えており、当該プロバイダインタフェースは、本発明においては、産業用オートメーションプラットフォームの、特にプログラマブルロジックコントローラの、オートメーションプログラムを管理および制御するように調整されている。1つ以上の仮想kubeleletが、原則的に、周知であり、また、ほぼ任意のプラットフォームにソフトウェアコンテナを導入するために、従来使用されている。当該プラットフォームは、コンテナランタイム環境を、特に、クラウド環境において、すなわちAWS（登録商標）（Amazon Web Service（登録商標））等のようなクラウドサービスプロバイダにおいて、エミュレートされたコンテナランタイム環境を、提供することができる。

【0010】

本発明においては、しかしながら、プロバイダインタフェースまたはプロバイダプラグインが仮想kubeleletのために提供され、当該プロバイダインタフェースまたはプロバイダプラグインは、コンテナではなくオートメーションプログラムを管理し、オートメーションプログラムは、特別な方法によって、管理されて、産業用オートメーションプラットフォームに送信されて、そこで起動（スタート）、停止（ストップ）等される必要がある。これは、本発明によれば、少なくとも、従来の産業用エンジニアリングシステムの以下のような機能性、すなわち、オートメーションプラットフォームで実行するソフトウェアの「デプロイメント」（ソフトウェア展開）および制御を備えて調整されている機能性、が、プロバイダインタフェースを用いて提供される必要があること、を意味する。この場合、そのように設計された仮想kubeleletは、産業用オートメーション装置の領域で実行可能である、つまり、例えば、対応して設けられたプログラマブルロジックコン

トローラまたは一般にオートメーションプラットフォームそれ自体において実行可能である一方、実用的には、別個のサーバでも、特に産業用エッジデバイスでも、実行可能である。代替的に、産業領域外での実行が可能であり、その場合、プロバイダインタフェースは、例えばVPNアクセスを用いての、オートメーションプラットフォームへのデータアクセスを必要とする。

【0011】

本課題は、特に、請求項1に記載の方法および請求項10に記載の装置によって、解決される。

【0012】

これによると、産業用オートメーションプラットフォーム用のオートメーションプログラムを管理する方法が提供され、オートメーションプログラムは、オートメーションプラットフォームへ送信され、オートメーションプログラムの実行は、オートメーションプラットフォーム上で制御される。その際、第1ステップでは、オートメーションプログラムまたはこのオートメーションプログラムへのリファレンスが、Kubernetesマスタから仮想kubele tへ送信され、第2ステップでは、仮想kubele tのプロバイダインタフェースによって、送信されたまたは参照されたオートメーションプログラムが、産業用オートメーションプラットフォームへ送信され、第3ステップでは、送信されたオートメーションプログラムの実行は、産業用オートメーションプラットフォーム上で制御され、プロバイダインタフェースによって、制御命令が産業用オートメーションプラットフォームへ送信され、産業用オートメーションプラットフォームの肯定応答メッセージが受信され、そして処理されるかまたは制御インスタンスへ、特にKubernetesマスタまたは制御監視装置(HMI装置)へ、転送される。本方法により、オートメーションプログラムを、コンテナオーケストレーションシステムの手段を用いて管理し、展開し、実行させることができる。

【0013】

また、本課題は、更に、以下のような産業用オートメーションプラットフォームのオートメーションプログラムを管理する装置によって、解決される、すなわち、オートメーションプログラムまたはオートメーションプログラムへのリファレンスを仮想kubele tへ送信するKubernetesマスタを備え、オートメーションプログラムまたはオートメーションプログラムへのリファレンスを受信するための、そして、オートメーションプラットフォームでのオートメーションプログラムのインストールのための命令を受信するための、オートメーションプログラムのプロセスをオートメーションプラットフォーム上での制御のための、仮想Kubele tを備え、オートメーションプログラムのオートメーションプラットフォームへの送信のための、オートメーションプログラムのプログラムプロセスの制御のための、そして、オートメーションプログラムのプロセスの実行中にオートメーションプラットフォームの肯定応答メッセージを受信及び転送するための、仮想Kubele tのプロバイダインタフェースを備える、装置によって、解決される。

【0014】

本発明に係る方法の有利な構成は、従属請求項に記載されている。そこに記載されている特徴およびそれらの有利な点は、本発明に係る装置にも、同様に適用される。有利な構成は、個別にも、そしてまた明白な組み合わせにおいても、実現することができる。

【0015】

適切に設計されたKubernetesマスタ、及びそれに対応して設計されたカウンターパート、つまり仮想kubele t、を用いて、原理的には、オートメーションプログラムまたはそのソースコードを直接的に供給することができるが、実用的には、Kubernetesマスタによって、参照はオートメーションプログラムへ送信される。リファレンスは、記憶場所への任意の適切な参照または「リンク」にして、そこから仮想kubele tまたはそのプロバイダインタフェースがオートメーションプログラムまたはそのソースコードをロードすることができる参照または「リンク」であってよい、特に、URL、メモリパス情報、インターネットアドレス、FTPアドレス等であってよい。

【0016】

有利な構成においては、送信されるまたは参照されるオートメーションプログラムは、オートメーションプラットフォームへの送信前に、前処理される。これは、特に、異なるプロパティを有する異なるオートメーションプラットフォームをアドレス指定することを可能とする。このようにして、例えば、オートメーションプログラムをそれぞれのターゲットハードウェアに関して適切にコンパイルすることができ、また、異なるオートメーションプラットフォームが、異なる方法でアドレス指定されそしてプログラムされることも、考慮に入れることができる。特に、有利な構成においては、プロバイダインタフェースは、最終的なオートメーションプログラムを複数のソースから「アセンブル」すること、つまり、例えば参照されるオブジェクトまたはライブラリを統合すること、プログラムをまとめてコンパイルすること、個別の要素のアップデートをリロードすること等を行うことができる。

10

【0017】

有利な変形例においては、プロバイダインタフェース、または、統合されたプロバイダインタフェース（プロバイダインタフェースプラグイン）を有する対応して拡張された仮想kubernetes、は、異なるタイプの複数のオートメーションプラットフォームに、オートメーションプログラムを供給し、そのプロセスを制御することができる。その一方、他の有利な変形例においては、異なるタイプまたはタイプシリーズのオートメーションプラットフォームに対して、異なる特化したプロバイダインタフェースを提供するとともに、その際、異なる仮想kubernetesも設けるか、または代替的に複数の異なる特化したプロバイダインタフェースプラグインを共通の仮想kubernetesに統合すること、も可能である。この柔軟性により、対応するシステムは、広範に適用可能であるだけでなく、ほぼ任意にスケーラブルでもある。

20

【0018】

特化したプロバイダインタフェースは、つまり、仮想kubernetes拡張用プラグインは、有利には複数の機能を含んでおり、当該機能は、産業用オートメーションプラットフォーム用の従来のエンジニアリングシステムの機能に対応するか、又は、それどころかそれらから取り出されている。特に有利な変形例においては、プロバイダインタフェースは、実際に利用可能なエンジニアリングシステムの、プログラミングインターフェース、いわゆるAPI（Application Programming Interface）、をアドレス指定して、これにより、その機能性を使用（例えば、ソフトウェアのコンパイル）する、または、オートメーションプログラムをオートメーションネットワークにおいてローカルに展開させそして制御しさえする。

30

【0019】

有利な変形例においては、プログラマブルロジックコントローラのリソースを管理するためにおよび／またはオートメーションプログラムを複数のオートメーションプラットフォームに割り当てるために、Kubernetesのカスタムリソース定義（CRD：Custom Resource Definitions）（<https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/>を参照）が使用される。これにより、コンテナベースのプラットフォームの使用との比較において考慮する必要があるオートメーションプラットフォームの特徴を、データ技術的に定義し自動的に考慮に入れることができる。

40

【0020】

有利には、いわゆるCustom Scheduler（<https://kubernetes.io/docs/concepts/scheduling/kube-scheduler/>を参照）を、つまり、オートメーションプラットフォームの使用に適合されたスケジューラを、そのターゲットでのオートメーションプログラムのデプロイメント（展開）に使用することも可能である。Kubernetesを用いて「ポッド」ごとにスケジューラを指定することが可能であり、当該スケジューラを用いてこのポッドは処理される。PLCプログラムに対して、固有の、つまりオートメーションプラットフォームの要件に特有の、適合されたスケジューラを使用す

50

る場合、次のことを達成できる。一方では、P L Cプログラムを他のポッドから容易に区別できる。これにより、P L Cプログラムの適切なP L Cノードへの割り当てを実行することができる。他方では、カスタムスケジューラでP L Cプログラムをスケジューリングする際、特徴を考慮に入れることができ、特に、オートメーションプラットフォームは、コンテナの意味において、多くの場合、パケットの「ダウンロード」によってではなく、複数の段階からなるシーケンスで充填されること、を考慮に入れることができる。

【0021】

実施バリエーションにおいては、サービス品質パラメータ（Q o Sパラメータ）が、P L Cプログラムを適切なP L Cノードまたは適したオートメーションプラットフォームに割り当てるために、使用される。Q o Sパラメータの一例は、コンピューティング容量、遅延時間（英：latency times、独：Latenzzeiten）、サイクル時間、可用性、プライバシー／セキュリティ等である。これにより、独立した、つまり自動的な、とりわけ機能的に信頼性高い、負荷分散が可能である。

【0022】

本発明に係る方法の実施形態例を、以下において図面を用いて説明する。実施例は、同時に、本発明に係る装置の説明にも、適用される。

【図面の簡単な説明】

【0023】

【図1】図1は、異なるタイプのプログラムを有する3つの異なるノードを備えかつ制御するK u b e r n e t e sマスタを概略図で示す。

【発明を実施するための形態】

【0024】

図には、3つの異なるコンピューティングノードであるノード1、ノード2、ノード3（ノード（英：Node、独：Knoten）；ターゲットハードウェア、「ターゲット」）でのソフトウェアの展開が示されている。このプロセスは、中央管理デバイスであるK u b e r n e t e sマスタによって、制御される。このアプリケーション（図中「K u b e r n e t e s」と記載されている）では、このプロセスのローカル制御は、コンピューティングノードであるノード1、ノード2、ノード3（英：Node、独：Knoten）で、つまり、実行されるプログラムのターゲットハードウェアで、各ノードに存在するエージェント（いわゆるk u b e l e t）によって、行われる。

【0025】

図では、ノード2のローカルで、ソフトウェアコンテナを管理するように設定されている従来のk u b e l e tが実行される。k u b e l e tは、その際、K u b e r n e t e sマスタの命令（「マニフェスト」）によって指定されたソフトウェアコンテナを、ローカル実行環境「コンテナランタイム」で実行できる。ソフトウェアコンテナは、k u b e l e tの命令に従って、指定されたソースからロードされる。

【0026】

また、ノードのk u b e l e tは、そのために、マスタから、例えばマニフェストとともにまたは後ほど、ノードで実行中のソフトウェアコンポーネントを起動、停止、変更するための対応する命令を、取得する。これらは、通常の場合、K u b e r n e t e sにおいては、ソフトウェアコンテナまたはソフトウェアコンテナのグルーピング（いわゆるポッド）である。K u b e r n e t e sマスタとk u b e l e t間のインタフェースは定義されている。k u b e l e tは、ソフトウェアコンテナを起動および操作できるように、ノード（「Node」）での実行環境（「コンテナランタイム」）に対して定義されたインタフェースを備える。

【0027】

K u b e r n e t e sを拡張する場合、いわゆる仮想k u b e l e tが導入され、これらは、図中、ノード1、ノード3により示されている。仮想k u b e l e tは、従来、互換性のないコンテナ環境のソフトウェアコンテナ（例えば、オペレーティングシステムM i c r o s o f t Windows（登録商標）を搭載したコンピュータのWindows（登録商標）コン

10

20

30

40

50

テナ、を管理するために使用される。仮想 `kubelet` の原則は、`Kubernetes` マスタに対するそのインタフェースが、従来の `kubelet` のインタフェースのように、設計されていることである。それは、従って、「上位へは」、従来の `kubelet` と同じインタフェースを有し、そして、マスタは、逆方向に、また従来の `Kubernetes` ノードと、つまり、コンテナ（またはポッド）を管理可能なノードと、通信する、ノード 2 の例を参照。「下位」へは、つまり、「ターゲット」との通信においては、仮想 `kubelet` は、原則的に、任意のインタフェースを有することができる。現行の実装（<https://github.com/virtual-kubelet/virtual-kubelet/>）では、これは、いわゆるプロバイダにより、解決される。従来の方法では、このプロバイダは、より大きなユニット（クラスタまたは `Fargate` のようなサーバレスコンテナ環境）を、単一のノードのように表示させるか、または、（例えば、`Microsoft Azure` コンテナのために）非互換性を解決するかの、何れかの役割を果たす。

10

【0028】

ノード 1、ノード 3 の図示された本発明による解決手段では、仮想 `kubelet` は、オートメーションプログラムのまたはオートメーションプログラムの一部の管理のために、使用される。この目的のために、特別な `PLC` プロバイダが実装されており、当該 `PLC` プロバイダは、オートメーションプログラムを、対応するオートメーションプラットフォーム（以下および図中において「`PLC` ランタイム」と称する）において、起動、停止、監視等を行うことができる。プロバイダが満たす必要がある基本機能は、<https://github.com/virtual-kubelet/virtual-kubelet/>、に説明されている。

20

【0029】

本実施例では、この特別なプロバイダは、それぞれ仮想 `kubelet` 用のプラグインとして実装されているが、他の実装においては、このための別個のコンポーネントを設けてもよい。プラグイン、つまりここでは「`PLC` プロバイダ」、は、それ自体、モジュールで構成されていてもよく、特に、ホストである仮想 `Kubelet` に接続するための汎用コンポーネントを有していてもよく、他方では、それぞれ特別なタイプのオートメーションプラットフォームを管理するための特定のモジュールを有していてもよい。

【0030】

具体的には、プロバイダプラグインである `PLC` プロバイダは、ここで、アドレス指定されたオートメーションプラットフォーム、いわゆる `PLC` ランタイム、の、オートメーションプログラムの実行環境と、通信する。ここでノード 1 を用いて、以下の場合について説明する、すなわち、仮想 `kubelet` が `PLC` プロバイダと共に、管理されるオートメーションプラットフォーム（ノード 1）それ自体にインストールされている場合について、説明する。仮想 `Kubelet` と `PLC` プロバイダは、この場合、ファームウェアの構成部であってよいし、または、別個のプログラムとしてインストールされていてもよい。反対に、ノード 3 を用いて、以下の場合が与えられている、すなわち、仮想 `kubelet` と `PLC` プロバイダが、別個のマシン、例えば、産業用エッジデバイス、で実行される場合が与えられており、当該エッジデバイスは、オートメーションプラットフォームであるノード 3 を含む産業レベルに対するインタフェースと、`Kubernetes` マスタに対するさらなるインタフェースと、を有し、後者はクラウドプロバイダのサーバ上でも実行可能である。

30

40

【0031】

プログラミング準備のため、オートメーションプラットフォーム（`PLC`）のオートメーションプログラムが、例えば、産業用エンジニアリングシステムのプロジェクト（例えば `TEA` ポータルプロジェクト）内で、生成され、「ポッド」にマッピングされ、参照可能なソースに格納される。オートメーションプラットフォーム（`PLC` または仮想 `PLC`）それ自体は、構成済みであり、使用可能であるようにネットワーク技術的に設定済みである。仮想 `PLC` は、この場合、仮想マシンで実行される純粋にソフトウェアベースの `PLC` インスタンスである。仮想 `PLC` の機能と論理インタフェースは、実 `PLC` に対応する。オートメーションプラットフォームであるノード 1、ノード 3 を管理するための `PL`

50

Cプロバイダのいくつかの基本機能のマッピングは、以下のようにして、可能である。

- ・ `CreatePod` → 存在するPLCまたは仮想PLCインスタンスにオートメーションプログラムをロードし、それを起動させる。その際、PLC／PLCインスタンスの対応するリソースが確保され、確保済みとしてマークされる。
- ・ `UpdatePod` → PLC／仮想PLCインスタンスで実行中のオートメーションプログラムを変更する。変更は、ソフトウェアの状態、および／または、オートメーションプログラムの構成または実行パラメータ、に関連し得る。
- ・ `DeletePod` → 実行中のオートメーションプログラムを停止し、PLC／仮想PLCインスタンスにおいて確保されたリソースを解放する。
- ・ `GetContainerLogs` → PLCの診断バッファを転送する、または、それに対応するオートメーションプログラムに関連するPLCの診断バッファの一部を転送する。
- ・ `GetPodStatus` → 関連するオートメーションプログラムの現在の実行詳細、例えば、サイクルタイム、メモリ要件、診断ステータス、表示コンテンツ、を表示する。
- ・ `NodeConditions` → PLC／仮想PLCインスタンスの抽象化状態（実行可能、停止済み、起動中、メモリ不足）を表示する。
- ・ `OperatingSystem` → PLCのバージョンを表示する。

【0032】

このマッピングでは、存在するエンジニアリングツール（例えばTIAポータル、STEP 7）を完全にPLCプロバイダにおいて使用できる。その一方で、モジュールコンポーネントは、PLCプロバイダをスリムに保ち、またコンピューティング容量が小さいノード上で実行可能とするために、有利である。従って、例えば、上記の基本機能は、モジュールまたはプラグインとして、PLCプロバイダに、設けられてよい。また、これらの基本機能を少なくとも部分的に、それに対応するサービスをクラウドまたはその他の稼働中のエンジニアリングシステムで呼び出すことによって、実装することもできる。

【0033】

PLCプロバイダは、従って、ポッドの起動時に、そのために設けられているか参照されるPLCプログラムをロードし、PLCランタイムに転送することができるようにするため、動作プロセスを実装している。これは、参照されるソース（リポジトリ）によって、または、他のソースやディレクトリを介して、行われ得る、代替的に、少なくとも部分的にPLCプログラムそれ自体が動作されるようにKubernetesコンポーネント（Kubernetesマスタ、仮想Kubelet）を変更することもできる。追加的に、PLCプロバイダによって、または、PLCプロバイダによってアドレス指定されたサービスによって、オートメーションプログラムを変更することもでき、特に、PLCランタイムのタイプや現在の状態（ステータス、例えば、空きメモリ容量、ファームウェアのバージョン等）に応じて、つまり実行環境またはオートメーションプラットフォーム自体に応じて、プログラムコードをプラットフォーム固有に適合すること又はコンパイルすることができる。

【0034】

特に、混在システム（異なるPLCタイプ、ソフトウェアコンテナシステムと混在するPLC、標準PC、クラウドでのコンテナランタイム）では、ランタイムの特定のタイプの区別／識別が、可能となる必要がある。これは、例えばKubernetesにおけるラベルを介して、行うことができる。また、PLCプロバイダは、実行環境のタイプと他のパラメータをそれ自体で決定し、オートメーションプログラムおよびアドレス指定または、一般に、実行環境とのデータ交換、を対応的に適合することもできる。

【0035】

PLCプロバイダは、オートメーションプラットフォーム（PLC）に統合可能であり、このことは、特に新しいタイプのオートメーションプラットフォームにおいて、ファームウェアと共にプリインストールすることによって、または、ファームウェアの一部と

10

20

30

40

50

してプリインストールするによって、納品時に既に容易に実行できる。上述したように、このことは、ノード1の例に見ることができる。既存のシステムの場合、PLCプロバイダは、ネットワークを介してオートメーションプラットフォーム／PLCおよびそこでの実行環境と接続するために、既存のコンピューティングノードで、又は、代替的に追加のノードで、起動され得る。この場合、例えば、エッジデバイス（産業用エッジデバイス）またはクラウドノードを、PLCプロバイダのプラットフォームとして、または、対応して設計された仮想kubernetのプラットフォームとして、使用することができる、これについては図のノード3を参照されたい。

【0036】

ソフトウェア展開のルート、すなわち少なくとも1つのオートメーションプログラムを有するポッドの「デプロイメント」のルートについて上述したが、それとは逆方向のルートで、制御情報、ステータスメッセージ、デバッグ値等を、「ターゲット」から関係する仮想kubernetのPLCプロバイダへと、そしてそこから必要に応じて処理された形でさらにKubernetesマスタまたは代替ターゲットへと、送信することができる。そのような代替ターゲットは、特に、オートメーションネットワークの内部または外部の駆動装置及び監視装置（HMI；SCADA；MES）であってよい。クラウドベースのアプリケーションにも到達され得る。ターゲットは、Kubernetesマスタを用いて、設定可能であり、代替的に、Kubernetesマスタは、このデータを、必要に応じて変更した形で、転送することもできる。

【0037】

本発明によれば、このようにして、PLCプログラムは、コンテナオーケストレーションシステムの手段を用いて、管理されそして動作され得る。その際、PLCプログラムをソフトウェアコンテナと共に混在操作することも可能である。

【0038】

これにより、ノード容量（英：node capacity、独：Knotenkapazitaet）、システムの負荷、可用性等に応じて（つまり、一般に、サービスパラメータの品質に応じて）、ノードへのPLCプログラムの（部分的）自動化割り当てが、可能である。これは、PLCプログラムまたはPLCプログラムの一部の自動化された「デプロイメント」（初期インストール、アップデート、トラブルシューティング、テスト等）により、より迅速なコミッショニング時間とアップデート時間を、を可能とする。さらなる有利な点は、現在可能な代替切り換え手法による、すなわち、PLCプログラムの監視、同じまたは他のノードでの再起動による、可用性の向上である。今や、CI／CDシステムまたはDevOpsへの簡潔な統合、つまり、ソースコード変更、変換、テスト、コミッショニングの自動化チェーンへの簡潔な統合、も可能である。

【0039】

産業用オートメーションプラットフォーム用に設定されたプロバイダプラグインを備える仮想kubernetの使用は、例えば、モニタリング、トレース、ロールアウト戦略、バージョニング、ロールバックのための既に存在するツールを用いた、PLCプログラムの管理のためのKubernetes「エコシステム」を開発するPLCプログラムの管理は、従って、オープンソース環境から、確立された標準ツールを用いて、実行可能であり、当該オープンソース環境では広範な能力基盤（英：competence base、独：Kompetenzbasis）が与えられており、使用されるソフトウェアツールが絶えず開発されている。（コンテナと従来のPLCを備えるITベースの）混在システムでの一元的なソフトウェア管理は、一元的なインフラストラクチャによる、そして、ITシステム専門家の追加的な教育訓練の必要性が低いことによる、利点をもたらす。さらに、現場の既存PLCシステムは、ITメカニズムを用いて、PLCシステムを変更することなく、開発され得る。

10

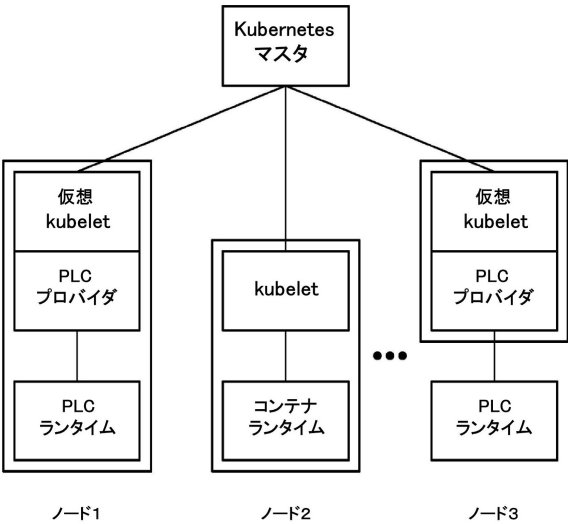
20

30

40

50

【図面】
【図 1】



10

20

30

40

50

【手続補正書】

【提出日】令和4年11月28日(2022.11.28)

【手続補正1】

【補正対象書類名】特許請求の範囲

【補正対象項目名】全文

【補正方法】変更

【補正の内容】

【特許請求の範囲】

【請求項1】

プログラマブルロジックコントローラのオートメーションプログラムを管理する方法であ
って、 10

前記オートメーションプログラムは、前記プログラマブルロジックコントローラに送信
され、前記オートメーションプログラムの実行は、前記プログラマブルロジックコントロ
ーラ上で制御される方法において、

第1ステップにて、前記オートメーションプログラムまたは前記オートメーションプロ
グラムへのリファレンスが、K u b e r n e t e s（登録商標）マスタから仮想k u b e
l e tへ、送信され、

第2ステップにて、リファレンスを送信する場合、前記仮想k u b e l e tのプロバイ
ダイインタフェースによって、参照される前記オートメーションプログラムは、前記プログ
ラマブルロジックコントローラに送信され、 20

第3ステップにて、送信された前記オートメーションプログラムの実行は、前記プログ
ラマブルロジックコントローラで制御され、前記プロバイダイインタフェースによって、制
御命令が前記プログラマブルロジックコントローラに送信され、前記プログラマブルロジ
ックコントローラの肯定応答メッセージが受信され、そして処理されるか、または、制御
インスタンスへ、特に前記K u b e r n e t e sマスタへ、転送される、
ことを特徴とする方法。

【請求項2】

請求項1に記載の方法において、

前記第1ステップにて、前記オートメーションプログラムへの前記リファレンスとして
、メモリアドレス、特に、U R L、メモリパス情報、アプリストアのコンテンツに関す
る仕様、または他のアドレス、が使用されること、 30
を特徴とする方法。

【請求項3】

請求項1または2に記載の方法において、

前記第2ステップにて、前記仮想k u b e l e tへ送信されるかまたはそれに前記リフ
ァレンスとして指定される前記オートメーションプログラムは、前記プログラマブルロジ
ックコントローラへの送信前に、前処理されること、
を特徴とする方法。

【請求項4】

請求項3に記載の方法において、 40

前記第2ステップにて、前記オートメーションプログラムは、前記送信前に、コンパ
イルされること、を特徴とする方法。

【請求項5】

請求項3または4に記載の方法において、

前記第2ステップにて、送信される前記オートメーションプログラムは、参照される複
数のソースから、および／または、前記K u b e r n e t e sマスタによって送信される
プログラムの一部から、作成されること、
を特徴とする方法。

【請求項6】

請求項1～5の何れか1項に記載の方法において、 50

複数の前記プログラマブルロジックコントローラを備える装置で、個別の前記プログラマブルロジックコントローラの各々に対して、または、同じタイプまたは同様の前記プログラマブルロジックコントローラのグループに対して、それぞれ固有の前記プロバイダインタフェースが使用され、使用される前記プロバイダインタフェースの各々は、それぞれの前記プログラマブルロジックコントローラへ送信される前記オートメーションプログラムを、それぞれのターゲットプログラマブルロジックコントローラのタイプおよび特別なプロパティへの適合のために、実行すること、

を特徴とする方法。

【請求項 7】

請求項 1～6 の何れか 1 項に記載の方法において、

10

前記仮想 k u b e l e t によって、産業用のエンジニアリングシステムまたはその一部が、前記プロバイダインタフェースとして使用され、前記エンジニアリングシステムまたは前記その一部は、前記 K u b e r n e t e s マスタから前記仮想 k u b e l e t へ送信される命令を用いて、制御されること、

を特徴とする方法。

【請求項 8】

請求項 1～7 の何れか 1 項に記載の方法において、

前記プログラマブルロジックコントローラのリソースを管理するために、および／または、前記オートメーションプログラムを複数の前記プログラマブルロジックコントローラのうちの 1 つに割り当てるために、K u b e r n e t e s のカスタムリソース定義 (C R D) が使用されること、

を特徴とする方法。

20

【請求項 9】

請求項 1～8 の何れか 1 項に記載の方法において、

前記プログラマブルロジックコントローラの使用に適合されたスケジューラ (K u b e r n e t e s C u s t o m S c h e d u l e r) が、前記プログラマブルロジックコントローラへの前記オートメーションプログラムへの展開に使用されること、

を特徴とする方法。

【請求項 10】

プログラマブルロジックコントローラのオートメーションプログラムを管理する装置であって、

30

前記オートメーションプログラムまたは前記オートメーションプログラムへのリファレンスを仮想 K u b e l e t へ送信する K u b e r n e t e s マスタを備え、

前記オートメーションプログラムまたは前記オートメーションプログラムへの前記リファレンスを受信するための、前記プログラマブルロジックコントローラでの前記オートメーションプログラムのインストールのための命令を受信するための、及び、前記プログラマブルロジックコントローラでの前記オートメーションプログラムのプロセスを制御するための、前記仮想 K u b e l e t を備え、

前記オートメーションプログラムを前記プログラマブルロジックコントローラへ送信するための、前記オートメーションプログラムのプログラムプロセスを制御するための、及び、前記オートメーションプログラムのプロセスの実行中に、前記プログラマブルロジックコントローラの肯定応答メッセージを受信及び転送するための、前記仮想 K u b e l e t のプロバイダインタフェースを備える、

装置。

40

【手続補正 2】

【補正対象書類名】 明細書

【補正対象項目名】 全文

【補正方法】 変更

【補正の内容】

【発明の詳細な説明】

50

【技術分野】

【0001】

本発明は、請求項1の前提部分に係る産業用オートメーションプラットフォームのオートメーションアプリケーションを管理する方法、および、請求項10の前提部分に係る産業用オートメーションプラットフォームのオートメーションアプリケーションを管理する装置、に関する。

【背景技術】

【0002】

オートメーションシステムのオートメーションアプリケーションは、1つまたは複数の、例えばIEC 61131規格に準拠しているオートメーションプログラムであって、産業用オートメーションプラットフォームで実行されるオートメーションプログラム、からなる。そのような産業用オートメーションプラットフォームは、例えば、プログラマブルロジックコントローラ（英：Programmable Logic Controller（略してPLC）、独：Speicherprogrammierbare Steuerungen）であってよい。このオートメーションプログラムの作成および管理、例えば、個別のコントローラへのプログラムの割り当て、プログラムのロード／起動／停止、プログラムのアップデート等は、一般に、いわゆるエンジニアリングシステムにおいて、例えばシーメンス株式会社（Siemens AG）のTIAポータルにおいて、行われる。これらアクティビティは、一般的にエンジニアリングと称され、通常、手動制御で行われる、すなわち、低い自動化レベルで、そして対応的に長い変更時間と高い変更コストで行われる。また、エンジニアリングシステムは、通常、産業用オートメーションプラットフォームにローカル接続する必要もあり、エンジニアリングシステムのバージョンは、使用されるオートメーションプラットフォーム（コントローラ）のそれぞれのバージョンおよびタイプと一致すること等、が必要とされる。

【0003】

以下、少なくとも、既に作成されたオートメーションプログラムを目的地又は「ターゲット」へ、つまり産業用オートメーションプラットフォーム（SPS、PLC等）へ、送信するプロセス、オートメーションプログラムのプロセスを開始および終了（スタート／ストップ）するステップ、制御エンティティ（英：control entity、独：Kontrollinstanz）（例えば、HMIデバイス（HMI：Human Machine Interface））に対するオートメーションプログラムを処理するプロセスの間の産業用オートメーションプラットフォームの肯定応答メッセージ（独：Quittierungsmeldungen、英：acknowledgment messages）を受信及び転送するステップを、まとめて、オートメーションプログラムの「管理」および「制御」と称する。

【0004】

ITベースのモジュラーシステムに対しては、つまり従来のコンピュータ技術によるシステムに対しては、既にしばらく前から、ソフトウェアコンテナ技術が導入されている。特に、複数のコンピューティングノードを備える分散システムにおいては、コンテナベースのプログラムは、いわゆるコンテナオーケストレーション技術（独：Container-Orchestration-Techniken、英：container orchestration technologies）、例えばKubernetes（登録商標）（<https://kubernetes.io/de/>）、を用いて、コンピューティングノードに分散され、実行され、監視される。Kubernetesでは、これらは、中央コンポーネント、いわゆるマスタまたはKubernetesマスタ、を介して行われる。これは、実行されるソフトウェアコンポーネントの説明、いわゆるマニフェスト、を取得する。マスタは、適切なノード（独：Knoten、英：Nodes）を選択し、そこにあるソフトウェアコンポーネントを有するコンテナを実行させるが、これは「デプロイメント」と称される。さらに、Kubernetesマスタは、コンポーネントを監視し、必要に応じて、リカバリするためのアクションを実行する。さらに、ランタイムに、コンポーネントの変更を行うことができる、例えば、アップデートされたバージョンの起動、スケーリング（スループットを増減させるためのインスタンスのスタート

／ストップ)等を、行うことができる。

【0005】

Kubernetesを使用するために基本的なことは、完全なソフトウェアが1つまたは複数のソフトウェアコンテナ(例えば、Dockerコンテナ(<https://www.docker.com/resources/what-container>))に統合されており、このコンテナが、ランタイム環境、いわゆるコンテナランタイムに、コンピュータハードウェア(通常の場合「ノード」と称される)で送信されることである。ここで、「ノード」の構成要素は、いわゆるkubel etであり、当該kubel etはローカルで実行されるソフトウェアであり、当該ソフトウェアは管理を行うKubernetesマスタおよびローカルコンテナランタイムと協働し、ソフトウェアを含むコンテナをローカルにダウンロードし、実行し、動作を監視する。

10

【0006】

上述のように、従来、オートメーションプログラムの管理(作成、ロード、スタート、ストップ)は、産業用オートメーションプラットフォーム(産業用コントローラ、SPS、PLC等)にローカル接続される特別なエンジニアリングシステムを用いて、通常の場合大部分が手動(マニュアル)で、行われる。通常の場合、個別のオートメーションプラットフォーム(例えば、PLC(プログラマブルロジックコントローラ))へのオートメーションプログラムの割り当ては、手動で行われる。

【0007】

また、従来のエンジニアリングシステムを使用することは、多くの場合、このエンジニアリングシステムのインスタンスが産業用ネットワークにローカルで存在することと関連付けられており、「遠隔制御」は、特にはクラウドベースのアプローチにおいては、遠隔作業場からのローカルエンジニアリングシステムへの専用アクセスを必要とし、これは、システム全体の高コストな構成を必要とする。

20

【0008】

非特許文献1は、「オーケストレーション」のための、つまり、リソースが限られているデバイスでのソフトウェアを有するコンテナの展開および稼働のための、仮想kubel etの使用、を記載している。

【0009】

非特許文献2は、プロプライエタリ制御プログラムを実行するためのコンテナでのプロプライエタリPLCハードウェアのエミュレーション、を開示している。

30

【先行技術文献】

【非特許文献】

【0010】

【非特許文献1】Goethals et al. “FLEDGE: Kubernetes Compatible Container Orchestration on Low-Resource Edge Devices” (ISBN:978-3-642-36741-0)

【非特許文献2】Goldschmidt et al. “Container-based architecture for flexible industrial control applications” (Journal of Systems Architecture Bd. 84 (2018), 28-36頁)

【発明の概要】

40

【発明が解決しようとする課題】

【0011】

従って、本発明の課題は、従来のオートメーションプログラムを高い自動化レベルで管理し、制御することであり、その際、可能な限り確立されており、広く普及しており、簡単に学習可能な方法を使用し、管理される産業用オートメーションプラットフォームに対してのエンジニアリングシステムの依存性を低下させ、エンジニアリングシステムへのローカルな依存性を軽減または除去することである。

【課題を解決するための手段】

【0012】

本発明の課題は、原則的に、Kubernetesマスタとkubel etとからなる

50

周知のシステムによって、解決され、そのような周知のシステムは、コンテナベースのプログラムをコンテナランタイム環境において管理および制御するように意図されている。本発明によれば、仮想 `k u b e l e t` (<https://github.com/virtual-kubelet>を参照) が使用され、当該仮想 `k u b e l e t` は、`K u b e r n e t e s` マスタに対して「本物の」 `k u b e l e t` のように振る舞う一方、特別なプロバイダインタフェース (<https://virtual-kubelet.io/docs/providers/#provider-interface>を参照) を備えており、当該プロバイダインタフェースは、本発明においては、産業用オートメーションプラットフォームの、特にプログラマブルロジックコントローラの、オートメーションプログラムを管理および制御するように調整されている。1つ以上の仮想 `k u b e l e t` が、原則的に、周知であり、また、ほぼ任意のプラットフォームにソフトウェアコンテナを導入するために、従来使用されている。当該プラットフォームは、コンテナランタイム環境を、特に、クラウド環境において、すなわち `A W S` (登録商標) (`Amazon Web Service` (登録商標)) 等のようなクラウドサービスプロバイダにおいて、エミュレートされたコンテナランタイム環境を、提供することができる。

10

20

30

【0013】

本発明においては、しかしながら、プロバイダインタフェースまたはプロバイダプラグインが仮想 `k u b e l e t` のために提供され、当該プロバイダインタフェースまたはプロバイダプラグインは、コンテナではなくオートメーションプログラムを管理し、オートメーションプログラムは、特別な方法によって、管理されて、産業用オートメーションプラットフォームに送信されて、そこで起動 (スタート)、停止 (ストップ) 等される必要がある。これは、本発明によれば、少なくとも、従来の産業用エンジニアリングシステムの以下のような機能性、すなわち、オートメーションプラットフォームで実行するソフトウェアの「デプロイメント」 (ソフトウェア展開) および制御を備えて調整されている機能性、が、プロバイダインタフェースを用いて提供される必要があること、を意味する。この場合、そのように設計された仮想 `k u b e l e t` は、産業用オートメーション装置の領域で実行可能である、つまり、例えば、対応して設けられたプログラマブルロジックコントローラまたは一般にオートメーションプラットフォームそれ自体において実行可能である一方、実用的には、別個のサーバでも、特に産業用エッジデバイスでも、実行可能である。代替的に、産業領域外での実行が可能であり、その場合、プロバイダインタフェースは、例えば `V P N` アクセスを用いての、オートメーションプラットフォームへのデータアクセスを必要とする。

【0014】

本課題は、特に、請求項 1 に記載の方法および請求項 10 に記載の装置によって、解決される。

【0015】

これによると、産業用オートメーションプラットフォーム用のオートメーションプログラムを管理する方法が提供され、オートメーションプログラムは、オートメーションプラットフォームへ送信され、オートメーションプログラムの実行は、オートメーションプラットフォーム上で制御される。その際、第 1 ステップでは、オートメーションプログラムまたはこのオートメーションプログラムへのリファレンスが、`K u b e r n e t e s` マスタから仮想 `k u b e l e t` へ送信され、第 2 ステップでは、仮想 `k u b e l e t` のプロバイダインタフェースによって、送信されたまたは参照されたオートメーションプログラムが、産業用オートメーションプラットフォームへ送信され、第 3 ステップでは、送信されたオートメーションプログラムの実行は、産業用オートメーションプラットフォーム上で制御され、プロバイダインタフェースによって、制御命令が産業用オートメーションプラットフォームへ送信され、産業用オートメーションプラットフォームの肯定応答メッセージが受信され、そして処理されるかまたは制御インスタンスへ、特に `K u b e r n e t e s` マスタまたは制御監視装置 (`H M I` 装置) へ、転送される。本方法により、オートメーションプログラムを、コンテナオーケストレーションシステムの手段を用いて管理し、展開し、実行させることができる。

40

50

【0016】

また、本課題は、更に、以下のような産業用オートメーションプラットフォームのオートメーションプログラムを管理する装置によって、解決される、すなわち、オートメーションプログラムまたはオートメーションプログラムへのリファレンスを仮想kubernetへ送信するKubernetesマスタを備え、オートメーションプログラムまたはオートメーションプログラムへのリファレンスを受信するための、そして、オートメーションプラットフォームでのオートメーションプログラムのインストールのための命令を受信するための、オートメーションプログラムのプロセスをオートメーションプラットフォーム上で制御のための、仮想Kubernetを備え、オートメーションプログラムのオートメーションプラットフォームへの送信のための、オートメーションプログラムのプログラムプロセスの制御のための、そして、オートメーションプログラムのプロセスの実行中にオートメーションプラットフォームの肯定応答メッセージを受信及び転送するための、仮想Kubernetのプロバイダインタフェースを備える、装置によって、解決される。

10

【0017】

本発明に係る方法の有利な構成は、従属請求項に記載されている。そこに記載されている特徴およびそれらの有利な点は、本発明に係る装置にも、同様に適用される。有利な構成は、個別にも、そしてまた明白な組み合わせにおいても、実現することができる。

【0018】

適切に設計されたKubernetesマスタ、及びそれに対応して設計されたカウンターパート、つまり仮想kubernet、を用いて、原理的には、オートメーションプログラムまたはそのソースコードを直接的に供給することができるが、実用的には、Kubernetesマスタによって、参照はオートメーションプログラムへ送信される。リファレンスは、記憶場所への任意の適切な参照または「リンク」にして、そこから仮想kubernetまたはそのプロバイダインタフェースがオートメーションプログラムまたはそのソースコードをロードすることができる参照または「リンク」であってよい、特に、URL、メモリパス情報、インターネットアドレス、FTPアドレス等であってよい。

20

【0019】

有利な構成においては、送信されるまたは参照されるオートメーションプログラムは、オートメーションプラットフォームへの送信前に、前処理される。これは、特に、異なるプロパティを有する異なるオートメーションプラットフォームをアドレス指定することを可能とする。このようにして、例えば、オートメーションプログラムをそれぞれのターゲットハードウェアに関して適切にコンパイルすることができ、また、異なるオートメーションプラットフォームが、異なる方法でアドレス指定されそしてプログラムされることも、考慮に入れることができる。特に、有利な構成においては、プロバイダインタフェースは、最終的なオートメーションプログラムを複数のソースから「アSEMBル」すること、つまり、例えば参照されるオブジェクトまたはライブラリを統合すること、プログラムをまとめてコンパイルすること、個別の要素のアップデートをリロードすること等を行うことができる。

30

【0020】

有利な変形例においては、プロバイダインタフェース、または、統合されたプロバイダインタフェース（プロバイダインタフェースプラグイン）を有する対応して拡張された仮想kubernet、は、異なるタイプの複数のオートメーションプラットフォームに、オートメーションプログラムを供給し、そのプロセスを制御することができる。その一方、他の有利な変形例においては、異なるタイプまたはタイプシリーズのオートメーションプラットフォームに対して、異なる特化したプロバイダインタフェースを提供するとともに、その際、異なる仮想kubernetも設けるか、または代替的に複数の異なる特化したプロバイダインタフェースプラグインを共通の仮想kubernetに統合すること、も可能である。この柔軟性により、対応するシステムは、広範に適用可能であるだけでなく、ほぼ任意にスケラブルでもある。

40

【0021】

50

特化したプロバイダインタフェースは、つまり、仮想 `k u b e l e t` 拡張用プラグインは、有利には複数の機能を含んでおり、当該機能は、産業用オートメーションプラットフォーム用の従来のエンジニアリングシステムの機能に対応するか、又は、それどころかそれらから取り出されている。特に有利な変形例においては、プロバイダインタフェースは、実際に利用可能なエンジニアリングシステムの、プログラミングインタフェース、いわゆる `A P I` (Application Programming Interface)、をアドレス指定して、これにより、その機能性を使用（例えば、ソフトウェアのコンパイル）する、または、オートメーションプログラムをオートメーションネットワークにおいてローカルに展開させそして制御しさえする。

【0022】

10

有利な変形例においては、プログラマブルロジックコントローラのリソースを管理するためにおよび／またはオートメーションプログラムを複数のオートメーションプラットフォームに割り当てるために、`K u b e r n e t e s` のカスタムリソース定義 (`CRD`: Custom Resource Definitions) (<https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/> を参照) が使用される。これにより、コンテナベースのプラットフォームの使用との比較において考慮する必要があるオートメーションプラットフォームの特徴を、データ技術的に定義し自動的に考慮に入れることができる。

【0023】

20

有利には、いわゆる `C u s t o m S c h e d u l e r` (<https://kubernetes.io/docs/concepts/scheduling/kube-scheduler/> を参照) を、つまり、オートメーションプラットフォームの使用に適合されたスケジューラを、そのターゲットでのオートメーションプログラムのデプロイメント（展開）に使用することも可能である。`K u b e r n e t e s` を用いて「ポッド」ごとにスケジューラを指定することが可能であり、当該スケジューラを用いてこのポッドは処理される。`P L C` プログラムに対して、固有の、つまりオートメーションプラットフォームの要件に特有の、適合されたスケジューラを使用する場合、次のことを達成できる。一方では、`P L C` プログラムを他のポッドから容易に区別できる。これにより、`P L C` プログラムの適切な `P L C` ノードへの割り当てを実行することができる。他方では、カスタムスケジューラで `P L C` プログラムをスケジューリングする際、特徴を考慮に入れることができ、特に、オートメーションプラットフォームは、コンテナの意味において、多くの場合、パケットの「ダウンロード」によってではなく、複数の段階からなるシーケンスで充填されること、を考慮に入れることができる。

30

【0024】

実施バリエーションにおいては、サービス品質パラメータ (`Q o S` パラメータ) が、`P L C` プログラムを適切な `P L C` ノードまたは適したオートメーションプラットフォームに割り当てるために、使用される。`Q o S` パラメータの一例は、コンピューティング容量、遅延時間（英: latency times、独: Latenzzeiten）、サイクル時間、可用性、プライバシー／セキュリティ等である。これにより、独立した、つまり自動的な、とりわけ機能的に信頼性高い、負荷分散が可能である。

【0025】

40

本発明に係る方法の実施形態例を、以下において図面を用いて説明する。実施例は、同時に、本発明に係る装置の説明にも、適用される。

【図面の簡単な説明】

【0026】

【図1】図1は、異なるタイプのプログラムを有する3つの異なるノードを備えかつ制御する `K u b e r n e t e s` マスタを概略図で示す。

【発明を実施するための形態】

【0027】

図には、3つの異なるコンピューティングノードであるノード1、ノード2、ノード3（ノード（英: Node、独: Knoten）；ターゲットハードウェア、「ターゲット」）での

50

ソフトウェアの展開が示されている。このプロセスは、中央管理デバイスである K u b e r n e t e s マスタによって、制御される。このアプリケーション（図中「K u b e r n e t e s」と記載されている）では、このプロセスのローカル制御は、コンピューティングノードであるノード1、ノード2、ノード3（英：Node、独：Knoten）で、つまり、実行されるプログラムのターゲットハードウェアで、各ノードに存在するエージェント（いわゆる k u b e l e t）によって、行われる。

【0028】

図では、ノード2のローカルで、ソフトウェアコンテナを管理するように設定されている従来の k u b e l e t が実行される。k u b e l e t は、その際、K u b e r n e t e s マスタの命令（「マニフェスト」）によって指定されたソフトウェアコンテナを、ローカル実行環境「コンテナランタイム」で実行できる。ソフトウェアコンテナは、k u b e l e t の命令に従って、指定されたソースからロードされる。

10

【0029】

また、ノードの k u b e l e t は、そのために、マスタから、例えばマニフェストとともにまたは後ほど、ノードで実行中のソフトウェアコンポーネントを起動、停止、変更するための対応する命令を、取得する。これらは、通常の場合、K u b e r n e t e s においては、ソフトウェアコンテナまたはソフトウェアコンテナのグルーピング（いわゆるポッド）である。K u b e r n e t e s マスタと k u b e l e t 間のインタフェースは定義されている。k u b e l e t は、ソフトウェアコンテナを起動および操作できるように、ノード（「Node」）での実行環境（「コンテナランタイム」）に対して定義されたインタフェースを備える。

20

【0030】

K u b e r n e t e s を拡張する場合、いわゆる仮想 k u b e l e t が導入され、これらは、図中、ノード1、ノード3により示されている。仮想 k u b e l e t は、従来、互換性のないコンテナ環境のソフトウェアコンテナ（例えば、オペレーティングシステム M i c r o s o f t W i n d o w s（登録商標）を搭載したコンピュータの W i n d o w s（登録商標）コンテナ、を管理するために使用される。仮想 k u b e l e t の原則は、K u b e r n e t e s マスタに対するそのインタフェースが、従来の k u b e l e t のインタフェースのように、設計されていることである。それは、従って、「上位へは」、従来の k u b e l e t と同じインタフェースを有し、そして、マスタは、逆方向に、また従来の K u b e r n e t e s ノードと、つまり、コンテナ（またはポッド）を管理可能なノードと、通信する、ノード2の例を参照。「下位」へは、つまり、「ターゲット」との通信においては、仮想 k u b e l e t は、原則的に、任意のインタフェースを有することができる。現行の実装（<https://github.com/virtual-kubelet/VirtualKubelet/>）では、これは、いわゆるプロバイダにより、解決される。従来の方法では、このプロバイダは、より大きなユニット（クラスタまたは F a r g a t e のようなサーバレスコンテナ環境）を、単一のノードのように表示させるか、または、（例えば、M i c r o s o f t A z u r e コンテナのために）非互換性を解決するかの、何れかの役割を果たす。

30

【0031】

ノード1、ノード3の図示された本発明による解決手段では、仮想 k u b e l e t は、オートメーションプログラムのまたはオートメーションプログラムの一部の管理のために、使用される。この目的のために、特別な P L C プロバイダが実装されており、当該 P L C プロバイダは、オートメーションプログラムを、対応するオートメーションプラットフォーム（以下および図中において「P L C ランタイム」と称する）において、起動、停止、監視等を行うことができる。プロバイダが満たす必要がある基本機能は、<https://github.com/virtual-kubelet/virtual-kubelet/>、に説明されている。

40

【0032】

本実施例では、この特別なプロバイダは、それぞれ仮想 k u b e l e t 用のプラグインとして実装されているが、他の実装においては、このための別個のコンポーネントを設けてもよい。プラグイン、つまりここでは「P L C プロバイダ」、は、それ自体、モジュラ

50

で構成されていてもよく、特に、ホストである仮想 K u b e l e t に接続するための汎用コンポーネントを有していてもよく、他方では、それぞれ特別なタイプのオートメーションプラットフォームを管理するための特定のモジュールを有していてもよい。

【0033】

具体的には、プロバイダプラグインである P L C プロバイダは、ここで、アドレス指定されたオートメーションプラットフォーム、いわゆる P L C ランタイム、の、オートメーションプログラムの実行環境と、通信する。ここでノード 1 を用いて、以下の場合について説明する、すなわち、仮想 k u b e l e t が P L C プロバイダと共に、管理されるオートメーションプラットフォーム（ノード 1）それ自体にインストールされている場合について、説明する。仮想 K u b e l e t と P L C プロバイダは、この場合、ファームウェアの構成部であってよいし、または、別個のプログラムとしてインストールされていてもよい。反対に、ノード 3 を用いて、以下の場合が与えられている、すなわち、仮想 k u b e l e t と P L C プロバイダが、別個のマシン、例えば、産業用エッジデバイス、で実行される場合が与えられており、当該エッジデバイスは、オートメーションプラットフォームであるノード 3 を含む産業レベルに対するインタフェースと、K u b e r n e t e s マスタに対するさらなるインタフェースと、を有し、後者はクラウドプロバイダのサーバ上でも実行可能である。

【0034】

プログラミング準備のため、オートメーションプラットフォーム（P L C）のオートメーションプログラムが、例えば、産業用エンジニアリングシステムのプロジェクト（例えば T I A ポータルプロジェクト）内で、生成され、「ポッド」にマッピングされ、参照可能なソースに格納される。オートメーションプラットフォーム（P L C または仮想 P L C）それ自体は、構成済みであり、使用可能であるようにネットワーク技術的に設定済みである。仮想 P L C は、この場合、仮想マシンで実行される純粋にソフトウェアベースの P L C インスタンスである。仮想 P L C の機能と論理インタフェースは、実 P L C に対応する。オートメーションプラットフォームであるノード 1、ノード 3 を管理するための P L C プロバイダのいくつかの基本機能のマッピングは、以下のようにして、可能である。

- ・ C r e a t e P o d → 存在する P L C または仮想 P L C インスタンスにオートメーションプログラムをロードし、それを起動させる。その際、P L C / P L C インスタンスの対応するリソースが確保され、確保済みとしてマークされる。
- ・ U p d a t e P o d → P L C / 仮想 P L C インスタンスで実行中のオートメーションプログラムを変更する。変更は、ソフトウェアの状態、および／または、オートメーションプログラムの構成または実行パラメータ、に関連し得る。
- ・ D e l e t e P o d → 実行中のオートメーションプログラムを停止し、P L C / 仮想 P L C インスタンスにおいて確保されたリソースを解放する。
- ・ G e t C o n t a i n e r L o g s → P L C の診断バッファを転送する、または、それに対応するオートメーションプログラムに関連する P L C の診断バッファの一部分を転送する。
- ・ G e t P o d S t a t u s → 関連するオートメーションプログラムの現在の実行詳細、例えば、サイクルタイム、メモリ要件、診断ステータス、表示コンテンツ、を表示する。
- ・ N o d e C o n d i t i o n s → P L C / 仮想 P L C インスタンスの抽象化状態（実行可能、停止済み、起動中、メモリ不足）を表示する。
- ・ O p e r a t i n g S y s t e m → P L C のバージョンを表示する。

【0035】

このマッピングでは、存在するエンジニアリングツール（例えば T I A ポータル、S T E P 7）を完全に P L C プロバイダにおいて使用できる。その一方で、モジュールコンポーネントは、P L C プロバイダをスリムに保ち、またコンピューティング容量が小さいノード上で実行可能とするために、有利である。従って、例えば、上記の基本機能は、モジュールまたはプラグインとして、P L C プロバイダに、設けられてよい。また、これらの

基本機能を少なくとも部分的に、それに対応するサービスをクラウドまたはその他の稼働中のエンジニアリングシステムで呼び出すことによって、実装することもできる。

【0036】

PLCプロバイダは、従って、ポッドの起動時に、そのために設けられているか参照されるPLCプログラムをロードし、PLCランタイムに転送することができるようにするため、動作プロセスを実装している。これは、参照されるソース（リポジトリ）によって、または、他のソースやディレクトリを介して、行われ得る、代替的に、少なくとも部分的にPLCプログラムそれ自体が動作されるようにKubernetesコンポーネント（Kubernetesマスタ、仮想Kubelet）を変更することもできる。追加的に、PLCプロバイダによって、または、PLCプロバイダによってアドレス指定されたサービスによって、オートメーションプログラムを変更することもでき、特に、PLCランタイムのタイプや現在の状態（ステータス、例えば、空きメモリ容量、ファームウェアのバージョン等）に応じて、つまり実行環境またはオートメーションプラットフォーム自体に応じて、プログラムコードをプラットフォーム固有に適合すること又はコンパイルすることができる。

10

【0037】

特に、混在システム（異なるPLCタイプ、ソフトウェアコンテナシステムと混在するPLC、標準PC、クラウドでのコンテナランタイム）では、ランタイムの特定のタイプの区別／識別が、可能となる必要がある。これは、例えばKubernetesにおけるラベルを介して、行うことができる。また、PLCプロバイダは、実行環境のタイプと他のパラメータをそれ自体で決定し、オートメーションプログラムおよびアドレス指定または、一般に、実行環境とのデータ交換、を対応的に適合することもできる。

20

【0038】

PLCプロバイダは、オートメーションプラットフォーム（PLC）に統合可能であり、このことは、特に新しいタイプのオートメーションプラットフォームにおいて、ファームウェアと共にプリインストールすることによって、または、ファームウェアの一部としてプリインストールすることによって、納品時に既に容易に実行できる。上述したように、このことは、ノード1の例に見ることができる。既存のシステムの場合、PLCプロバイダは、ネットワークを介してオートメーションプラットフォーム／PLCおよびそこでの実行環境と接続するために、既存のコンピューティングノードで、又は、代替的に追加のノードで、起動され得る。この場合、例えば、エッジデバイス（産業用エッジデバイス）またはクラウドノードを、PLCプロバイダのプラットフォームとして、または、対応して設計された仮想kubletのプラットフォームとして、使用することができる、これについては図のノード3を参照されたい。

30

【0039】

ソフトウェア展開のルート、すなわち少なくとも1つのオートメーションプログラムを有するポッドの「デプロイメント」のルートについて上述したが、それとは逆方向のルートで、制御情報、ステータスメッセージ、デバッグ値等を、「ターゲット」から関係する仮想kubletのPLCプロバイダへと、そしてそこから必要に応じて処理された形でさらにKubernetesマスタまたは代替ターゲットへと、送信することができる。そのような代替ターゲットは、特に、オートメーションネットワークの内部または外部の駆動装置及び監視装置（HMI；SCADA；MES）であってよい。クラウドベースのアプリケーションにも到達され得る。ターゲットは、Kubernetesマスタを用いて、設定可能であり、代替的に、Kubernetesマスタは、このデータを、必要に応じて変更した形で、転送することもできる。

40

【0040】

本発明によれば、このようにして、PLCプログラムは、コンテナオーケストレーションシステムの手段を用いて、管理されそして動作され得る。その際、PLCプログラムをソフトウェアコンテナと共に混在操作することも可能である。

【0041】

50

これにより、ノード容量（英：node capacity、独：Knotenkapazitaet）、システムの負荷、可用性等に応じて（つまり、一般に、サービスパラメータの品質に応じて）、ノードへのP L Cプログラムの（部分的）自動化割り当てが、可能である。これは、P L CプログラムまたはP L Cプログラムの一部の自動化された「デプロイメント」（初期インストール、アップデート、トラブルシューティング、テスト等）により、より迅速なコミッショニング時間とアップデート時間を、を可能とする。さらなる有利な点は、現在可能な代替切り換え手法による、すなわち、P L Cプログラムの監視、同じまたは他のノードでの再起動による、可用性の向上である。今や、C I / C DシステムまたはD e v O p sへの簡潔な統合、つまり、ソースコード変更、変換、テスト、コミッショニングの自動化チェーンへの簡潔な統合、も可能である。

10

【0042】

産業用オートメーションプラットフォーム用に設定されたプロバイダプラグインを備える仮想k u b e l e tの使用は、例えば、モニタリング、トレース、ロールアウト戦略、バージョニング、ロールバックのための既に存在するツールを用いた、P L Cプログラムの管理のためのK u b e r n e t e s「エコシステム」を開発するP L Cプログラムの管理は、従って、オープンソース環境から、確立された標準ツールを用いて、実行可能であり、当該オープンソース環境では広範な能力基盤（英：competence base、独：Kompetenzbasis）が与えられており、使用されるソフトウェアツールが絶えず開発されている。（コンテナと従来のP L Cを備えるI Tベースの）混在システムでの一元的なソフトウェア管理は、一元的なインフラストラクチャによる、そして、I Tシステム専門家の追加的な教育訓練の必要性が低いことによる、利点をもたらす。さらに、現場の既存P L Cシステムは、I Tメカニズムを用いて、P L Cシステムを変更することなく、開発され得る。

20

30

40

50

【手続補正書】

【提出日】令和4年11月28日(2022.11.28)

【手続補正1】

【補正対象書類名】特許請求の範囲

【補正対象項目名】全文

【補正方法】変更

【補正の内容】

【特許請求の範囲】

【請求項1】

プログラマブルロジックコントローラのオートメーションプログラムを管理する方法であって、 10

前記オートメーションプログラムは、前記プログラマブルロジックコントローラに送信され、前記オートメーションプログラムの実行は、前記プログラマブルロジックコントローラ上で制御される方法において、

第1ステップにて、前記オートメーションプログラムまたは前記オートメーションプログラムへのリファレンスが、Kubernetes（登録商標）マスタから仮想kubele 20

第2ステップにて、リファレンスを送信する場合、前記仮想kubeleのプロバイダインタフェースによって、参照される前記オートメーションプログラムは、前記プログラマブルロジックコントローラに送信され、 20

第3ステップにて、送信された前記オートメーションプログラムの実行は、前記プログラマブルロジックコントローラで制御され、前記プロバイダインタフェースによって、制御命令が前記プログラマブルロジックコントローラに送信され、前記プログラマブルロジックコントローラの肯定応答メッセージが受信され、そして処理されるか、または、制御インスタンスへ、転送される、
ことを特徴とする方法。

【請求項2】

請求項1に記載の方法において、

前記第1ステップにて、前記オートメーションプログラムへの前記リファレンスとして、メモリアドレス、メモリパス情報、アプリストアのコンテンツに関する仕様、または他のアドレス、が使用されること、
を特徴とする方法。 30

【請求項3】

請求項1または2に記載の方法において、

前記第2ステップにて、前記仮想kubeleへ送信されるかまたはそれに前記リファレンスとして指定される前記オートメーションプログラムは、前記プログラマブルロジックコントローラへの送信前に、前処理されること、
を特徴とする方法。

【請求項4】

請求項3に記載の方法において、

前記第2ステップにて、前記オートメーションプログラムは、前記送信前に、コンパイルされること、を特徴とする方法。 40

【請求項5】

請求項3または4に記載の方法において、

前記第2ステップにて、送信される前記オートメーションプログラムは、参照される複数のソースから、および／または、前記Kubernetesマスタによって送信されるプログラムの一部から、作成されること、
を特徴とする方法。

【請求項6】

請求項1～5の何れか1項に記載の方法において、 50

複数の前記プログラマブルロジックコントローラを備える装置で、個別の前記プログラマブルロジックコントローラの各々に対して、または、同じタイプまたは同様の前記プログラマブルロジックコントローラのグループに対して、それぞれ固有の前記プロバイダインタフェースが使用され、使用される前記プロバイダインタフェースの各々は、それぞれの前記プログラマブルロジックコントローラへ送信される前記オートメーションプログラムを、それぞれのターゲットプログラマブルロジックコントローラのタイプおよび特別なプロパティへの適合のために、実行すること、
を特徴とする方法。

【請求項 7】

請求項 1～6 の何れか 1 項に記載の方法において、

前記仮想 `k u b e l e t` によって、産業用のエンジニアリングシステムまたはその一部が、前記プロバイダインタフェースとして使用され、前記エンジニアリングシステムまたは前記その一部は、前記 `K u b e r n e t e s` マスタから前記仮想 `k u b e l e t` へ送信される命令を用いて、制御されること、
を特徴とする方法。

10

【請求項 8】

請求項 1～7 の何れか 1 項に記載の方法において、

前記プログラマブルロジックコントローラのリソースを管理するために、および／または、前記オートメーションプログラムを複数の前記プログラマブルロジックコントローラのうちの 1 つに割り当てるために、`K u b e r n e t e s` のカスタムリソース定義 (`C R D`) が使用されること、
を特徴とする方法。

20

【請求項 9】

請求項 1～8 の何れか 1 項に記載の方法において、

前記プログラマブルロジックコントローラの使用に適合されたスケジューラ (`K u b e r n e t e s C u s t o m S c h e d u l e r`) が、前記プログラマブルロジックコントローラへの前記オートメーションプログラムへの展開に使用されること、
を特徴とする方法。

【請求項 10】

プログラマブルロジックコントローラのオートメーションプログラムを管理する装置であって、

30

前記オートメーションプログラムまたは前記オートメーションプログラムへのリファレンスを仮想 `K u b e l e t` へ送信する `K u b e r n e t e s` マスタを備え、

前記オートメーションプログラムまたは前記オートメーションプログラムへの前記リファレンスを受信するための、前記プログラマブルロジックコントローラでの前記オートメーションプログラムのインストールのための命令を受信するための、及び、前記プログラマブルロジックコントローラでの前記オートメーションプログラムのプロセスを制御するための、前記仮想 `K u b e l e t` を備え、

前記オートメーションプログラムを前記プログラマブルロジックコントローラへ送信するための、前記オートメーションプログラムのプログラムプロセスを制御するための、及び、前記オートメーションプログラムのプロセスの実行中に、前記プログラマブルロジックコントローラの肯定応答メッセージを受信及び転送するための、前記仮想 `K u b e l e t` のプロバイダインタフェースを備える、
装置。

40

【手続補正 2】

【補正対象書類名】 明細書

【補正対象項目名】 0030

【補正方法】 変更

【補正の内容】

【0030】

50

Kubernetesを拡張する場合、いわゆる仮想kubel etが導入され、これらは、図中、ノード1、ノード3により示されている。仮想kubel etは、従来、互換性のないコンテナ環境のソフトウェアコンテナ(例えば、オペレーティングシステムMicrosoft Windows(登録商標)を搭載したコンピュータのWindows(登録商標)コンテナ、を管理するために使用される。仮想kubel etの原則は、Kubernetesマスタに対するそのインタフェースが、従来のkubel etのインタフェースのように、設計されていることである。それは、従って、「上位へは」、従来のkubel etと同じインタフェースを有し、そして、マスタは、逆方向に、また従来のKubernetesノードと、つまり、コンテナ(またはポッド)を管理可能なノードと、通信する、ノード2の例を参照。「下位」へは、つまり、「ターゲット」との通信においては、仮想kubel etは、原則的に、任意のインタフェースを有することができる。現行の実装(<https://github.com/virtual-kubelet/virtual-kubelet>)では、これは、いわゆるプロバイダにより、解決される。従来の方法では、このプロバイダは、より大きなユニット(クラスタまたはFargateのようなサーバレスコンテナ環境)を、単一のノードのように表示させるか、または、(例えば、Microsoft Azureコンテナのために)非互換性を解決するかの、何れかの役割を果たす。

10

20

30

40

50

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.

PCT/EP2021/054633

A. CLASSIFICATION OF SUBJECT MATTER G06F 9/455(2018.01)i According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) G06F Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EPO-Internal, WPI Data		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GOETHALS TOM ET AL. "FLEDGE: Kubernetes Compatible Container Orchestration on Low-Resource Edge Devices" <i>12TH EUROPEAN CONFERENCE ON COMPUTER VISION, ECCV 2012; [LECTURE NOTES IN COMPUTER SCIENCE]</i> , Vol. 16, No. 484262, 19 January 2020 (2020-01-19), pages 174-189 ISSN: 0302-9743, ISBN: 978-3-642-36741-0. XP047552235 abstract page 174, paragraph 1. - page 176, paragraph 2. page 177, paragraph 3. - page 183, paragraph 6. page 186, paragraph 7. - page 189, last line	1-10
A	THOMAS GOLDSCHMIDT ET AL. "Container-based architecture for flexible industrial control applications" <i>JOURNAL OF SYSTEMS ARCHITECTURE</i> , NL, Vol. 84, 06 March 2018 (2018-03-06), pages 28-36 DOI: 10.1016/j.sysarc.2018.03.002 ISSN: 1383-7621, XP055516260 abstract page 28, paragraph 1. - page 32, right-hand column, paragraph 4.	1-10
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family		
Date of the actual completion of the international search 19 May 2021		Date of mailing of the international search report 31 May 2021
Name and mailing address of the ISA/EP European Patent Office p.b. 5818, Patentlaan 2, 2280 HV Rijswijk Netherlands Telephone No. (+31-70)340-2040 Facsimile No. (+31-70)340-3016		Authorized officer Skomorowski, Markus Telephone No.

Form PCT/ISA/210 (second sheet) (January 2015)

10

20

30

40

50

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen

PCT/EP2021/054633

A. KLASIFIZIERUNG DES ANMELDUNGSGEGENSTANDES

INV. G06F9/455

ADD.

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)

G06F

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)

EPO-Internal, WPI Data

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	GOETHALS TOM ET AL: "FLEDGE: Kubernetes Compatible Container Orchestration on Low-Resource Edge Devices", 12TH EUROPEAN CONFERENCE ON COMPUTER VISION, ECCV 2012; [LECTURE NOTES IN COMPUTER SCIENCE], Bd. 16, Nr. 484262, 19. Januar 2020 (2020-01-19), Seiten 174-189, XP047552235, ISSN: 0302-9743 ISBN: 978-3-642-36741-0 Zusammenfassung Seite 174, Absatz 1. - Seite 176, Absatz 2. Seite 177, Absatz 3. - Seite 183, Absatz 6. Seite 186, Absatz 7. - Seite 189, letzter Zeile ----- -/--	1-10

☒ Weitere Veröffentlichungen sind die Fortsetzung von Feld C zu entnehmen
 ☐ Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :

A Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

E frühere Anmeldung oder Patent, die bzw. das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

L Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

O Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

P Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

T Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

X Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfindersicher Tätigkeit beruhend betrachtet werden

Y Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfindersicher Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

Z Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche

19. Mai 2021

Absenddatum des internationalen Recherchenberichts

31/05/2021

Name und Postanschrift der Internationalen Recherchenbehörde

Europäisches Patentamt, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Bevollmächtigter Bediensteter

Skomorowski, Markus

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen

PCT/EP2021/054633

C. (Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN		
Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
A	THOMAS GOLDSCHMIDT ET AL: "Container-based architecture for flexible industrial control applications", JOURNAL OF SYSTEMS ARCHITECTURE, Bd. 84, 6. März 2018 (2018-03-06), Seiten 28-36, XP055516260, NL ISSN: 1383-7621, DOI: 10.1016/j.sysarc.2018.03.002 Zusammenfassung Seite 28, Absatz 1. - Seite 32, rechte Spalte, Absatz 4. -----	1-10

フロントページの続き

MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,N
E,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,
CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IR,IS,IT,JO,JP,K
E,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,N
G,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,TH,TJ,TM,
TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,ZA,ZM,ZW

(72)発明者 ゲッツ, ヤン

ドイツ連邦共和国 9 1 0 5 4 ブッケンホーフ, テネンローアーシュトラッセ 1 0

(72)発明者 ミッターマイアー, ルートヴィヒ アンドレアス

ドイツ連邦共和国 8 1 5 4 1 ミュンヘン, ホーエンヴァルデックシュトラッセ 1 8

(72)発明者 ミュラー, ハラルド

ドイツ連邦共和国 8 2 2 2 9 ゼーフェルトーヘンドルフ, ヴァッサーヴェーク 4

(72)発明者 プレムナド, スリーナス

ドイツ連邦共和国 8 2 2 1 6 マイザッハ, メーリアンシュトラッセ 7 アー

F ターム (参考) 5B376 AA07 AB19

5H220 BB05 CC09 EE01 HH01 JJ12

【要約の続き】

u b e r n e t e s マスタへ、転送される。本方法により、オートメーションプログラムを、コンテナオーケス
トレーションシステムの手段を用いて管理し、展開し、実行させることができる。