

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7559532号
(P7559532)

(45)発行日 令和6年10月2日(2024.10.2)

(24)登録日 令和6年9月24日(2024.9.24)

(51)国際特許分類	F I			
G 0 6 F 8/60 (2018.01)	G 0 6 F 8/60			
G 0 6 F 9/455(2018.01)	G 0 6 F 9/455	1 5 0		
G 0 6 F 9/50 (2006.01)	G 0 6 F 9/50	1 5 0 Z		
G 0 6 F 11/36 (2006.01)	G 0 6 F 11/36	1 9 6		
G 0 5 B 19/05 (2006.01)	G 0 5 B 19/05	A		
請求項の数 10 (全30頁)				

(21)出願番号	特願2020-204061(P2020-204061)	(73)特許権者	000002945
(22)出願日	令和2年12月9日(2020.12.9)		オムロン株式会社
(65)公開番号	特開2022-91301(P2022-91301A)		京都府京都市下京区塩小路通堀川東入南
(43)公開日	令和4年6月21日(2022.6.21)		不動堂町8 0 1 番地
審査請求日	令和5年10月10日(2023.10.10)	(74)代理人	110001195
			弁理士法人深見特許事務所
		(72)発明者	重森 弓束
			京都府京都市下京区塩小路通堀川東入南
			不動堂町8 0 1 番地 オムロン株式会社内
		審査官	多賀 実
最終頁に続く			

(54)【発明の名称】 制御システムおよび制御方法

(57)【特許請求の範囲】

【請求項1】

1または複数の制御装置と1または複数のサーバとを備える制御システムであって、
制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行
する実行環境とを内包するランタイムのイメージを含むコンテナを生成する生成モジュールと、

前記制御装置および前記サーバの一方または両方に対して前記コンテナをリリースする
オーケストレータとを備え、

前記制御装置および前記サーバは、前記コンテナをデプロイするコンテナエンジンを有
しており、前記オーケストレータによりリリースされたコンテナをデプロイすることで、
前記ランタイムのインスタンスを生成して前記制御演算を実行する、制御システム。

【請求項2】

前記サーバのコンテナエンジンは、前記ランタイムのインスタンスとともに、前記制御
対象とやり取りされる入出力データをエミュレートするエミュレータのインスタンスも生
成する、請求項1に記載の制御システム。

【請求項3】

前記サーバのコンテナエンジンが前記ランタイムのインスタンスを生成する場合に、前
記制御装置のコンテナエンジンが前記制御装置に接続されたデバイスを介してやり取りさ
れる入出力データを前記サーバに仲介するネットワークサーバのインスタンスを生成する
、請求項1に記載の制御システム。

【請求項 4】

前記サーバのコンテナエンジンは、前記ランタイムのインスタンスを生成する場合に、前記制御装置に接続されたデバイスへアクセスする処理を担当する I / O ドライバをエミュレートする仮想 I / O ドライバのインスタンスを生成する、請求項 1 ～ 3 のいずれか 1 項に記載の制御システム。

【請求項 5】

前記コンテナは、前記制御装置のコンテナエンジンのみがデプロイするイメージと、前記サーバのコンテナエンジンのみがデプロイするイメージとのうち、少なくとも一方を含む、請求項 1 ～ 4 のいずれか 1 項に記載の制御システム。

【請求項 6】

前記サーバのコンテナエンジンが前記ランタイムのインスタンスを生成する場合に、当該コンテナエンジンにアクセスするためのアドレスを管理するリポジトリをさらに備える、請求項 1 ～ 5 のいずれか 1 項に記載の制御システム。

【請求項 7】

前記サーバに対して前記コンテナをリリースする場合に、当該コンテナをデプロイするコンテナエンジンが存在しなければ、コンテナエンジンを含むノードを仮想的に生成するノードマネージャをさらに備える、請求項 1 ～ 6 のいずれか 1 項に記載の制御システム。

【請求項 8】

前記生成モジュールにより生成されるコンテナをリリース可能な制御装置の識別情報と関連付けて登録するコンテナハブをさらに備える、請求項 1 ～ 7 のいずれか 1 項に記載の制御システム。

【請求項 9】

前記オーケストレータは、ユーザ操作に従って、リリースするコンテナを選択するとともに、リリース先を決定する、請求項 1 ～ 8 のいずれか 1 項に記載の制御システム。

【請求項 10】

1 または複数の制御装置と 1 または複数のサーバとを備える制御システムにおいて実行される制御方法であって、

制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行する実行環境とを内包するランタイムのイメージを含むコンテナを生成するステップと、

前記制御装置および前記サーバの一方または両方に対して前記コンテナをリリースするステップとを備え、

前記制御装置および前記サーバは、前記コンテナをデプロイするコンテナエンジンを有しており、前記リリースされたコンテナをデプロイすることで、前記ランタイムのインスタンスを生成して前記制御演算を実行する、制御方法。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、制御装置およびサーバを含む制御システムおよびその制御システムにおける制御方法に関する。

【背景技術】

【0002】

複数の制御装置（典型的には、PLC（プログラマブルロジックコントローラ））からなる制御システムの運用を開始するには、設計、実装、検証などの一連の作業が必要である。これらの一連の作業では、制御装置単体について設計、実装、検証を行った後に、制御装置およびデバイスを組み合わせる。このようなシステムとして結合するためには、各制御装置についての作業が完了している必要があるが、制御装置毎の複雑さや特性などによって、作業進捗を同じにすることは難しい。そのため、作業進捗のばらつきによって、システムとして結合するまでに待ち時間が発生し得る。

【0003】

ところで、制御装置の技術分野ではなく、一般的な IT 技術の分野では、コンテナおよ

10

20

30

40

50

びオーケストレーションといった技術を利用して、コンピュータリソースの動的に拡張および縮退することで、スケーラブルなソフトウェアサービスの提供が実現されている。例えば、特表 2015-512091 号公報（特許文献 1）は、コンピュータクラウドリソースを管理する方法を開示する。

【先行技術文献】

【特許文献】

【0004】

【文献】特表 2015-512091 号公報

【発明の概要】

【発明が解決しようとする課題】

【0005】

上述したような複数の制御装置および複数の制御装置を組み合わせたシステムについての設計、実装、検証などの一連の作業を容易化するとともに、制御装置のリソースをスケーラブルに利用可能にする構成が要望されている。

【0006】

本発明の一つの目的は、このような要望に応える解決手段を提供することである。

【課題を解決するための手段】

【0007】

本発明の一例に従えば、1 または複数の制御装置と 1 または複数のサーバとを備える制御システムが提供される。制御システムは、制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行する実行環境とを内包するランタイムのイメージを含むコンテナを生成する生成モジュールと、制御装置およびサーバの一方または両方に対してコンテナをリリースするオーケストレータとを含む。制御装置およびサーバは、コンテナをデプロイするコンテナエンジンを有しており、オーケストレータによりリリースされたコンテナをデプロイすることで、ランタイムのインスタンスを生成して制御演算を実行する。この構成によれば、ランタイムのイメージを含むコンテナを制御装置およびサーバのいずれでもデプロイして実行できるので、制御装置が存在しない状態であっても、制御演算の実行の検証などを容易化できる。また、サーバで実行されている制御演算を制御装置へ容易に移動させることもできる。

【0008】

サーバのコンテナエンジンは、ランタイムのインスタンスとともに、制御対象とやり取りされる入出力データをエミュレートするエミュレータのインスタンスも生成してもよい。この構成によれば、制御装置に接続されるデバイスが存在しなくても、サーバ上で制御演算の検証などを容易に行うことができる。

【0009】

サーバのコンテナエンジンがランタイムのインスタンスを生成する場合に、制御装置のコンテナエンジンが制御装置に接続されたデバイスを介してやり取りされる入出力データをサーバに仲介するネットワークサーバのインスタンスを生成してもよい。この構成によれば、制御装置に接続されたデバイスとの間でやり取りされる I/O データを利用しつつ、制御演算自体はサーバで実行できるので、サーバが提供するリソースを利用できる。

【0010】

サーバのコンテナエンジンは、ランタイムのインスタンスを生成する場合に、制御装置に接続されたデバイスへアクセスする処理を担当する I/O ドライバをエミュレートする仮想 I/O ドライバのインスタンスを生成してもよい。この構成によれば、制御装置上で制御演算を実行するのと実質的に同じ実行環境を提供できるので、制御演算を実現するためのランタイムをサーバにおいても共通に利用できる。

【0011】

コンテナは、制御装置のコンテナエンジンのみがデプロイするイメージと、サーバのコンテナエンジンのみがデプロイするイメージとのうち、少なくとも一方を含んでもよい。この構成によれば、コンテナに冗長にイメージを含ませることで、デプロイ先に

10

20

30

40

50

依存せず制御演算を実現できる。

【0012】

制御システムは、サーバのコンテナエンジンがランタイムのインスタンスを生成する場合に、当該コンテナエンジンにアクセスするためのアドレスを管理するリポジトリをさらに含んでいてもよい。この構成によれば、サーバ上で制御演算を実行する場合に、実行環境に割り当てられるアドレスの違いをリポジトリの参照により吸収できる。

【0013】

制御システムは、サーバに対してコンテナをリリースする場合に、当該コンテナをデプロイするコンテナエンジンが存在しなければ、コンテナエンジンを含むノードを仮想的に生成するノードマネージャをさらに含んでいてもよい。この構成によれば、要求に応じて、制御演算を実行するノードを任意に生成できる。

【0014】

制御システムは、生成モジュールにより生成されるコンテナをリリース可能な制御装置の識別情報と関連付けて登録するコンテナハブをさらに含んでいてもよい。この構成によれば、一度生成されたコンテナの再利用が容易になる。

【0015】

オーケストレータは、ユーザ操作に従って、リリースするコンテナを選択するとともに、リリース先を決定してもよい。この構成によれば、ユーザは、制御システムの状況に応じて、必要なコンテナを必要な実行環境にリリースできる。

【0016】

本発明の別の一例に従えば、1または複数の制御装置と1または複数のサーバとを備える制御システムにおいて実行される制御方法が提供される。制御方法は、制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行する実行環境とを内包するランタイムのイメージを含むコンテナを生成するステップと、制御装置およびサーバの一方または両方に対してコンテナをリリースするステップとを含む。制御装置およびサーバは、コンテナをデプロイするコンテナエンジンを有しており、リリースされたコンテナをデプロイすることで、ランタイムのインスタンスを生成して制御演算を実行する。

【発明の効果】

【0017】

本発明のある局面によれば、複数の制御装置および複数の制御装置を組み合わせたシステムについての設計、実装、検証などの一連の作業を容易化できる。本発明の別の局面によれば、制御装置のリソースをスケラブルに利用可能にできる。

【図面の簡単な説明】

【0018】

【図1】本実施の形態に係る制御システムの全体構成例を示す模式図である。

【図2】本実施の形態に係る制御システムにおけるコンテナを利用した処理の概要を説明するための図である。

【図3】本実施の形態に係る制御システムにおいて提供されるモードを説明するための図である。

【図4】本実施の形態に係る制御システムを構成するPLCのハードウェア構成例を示すブロック図である。

【図5】本実施の形態に係る制御システムを構成するサーバのハードウェア構成例を示すブロック図である。

【図6】本実施の形態に係る制御システムを構成する開発保守端末のハードウェア構成例を示すブロック図である。

【図7】本実施の形態に係る制御システムにおける機能構成の一例を示す模式図である。

【図8】本実施の形態に係る制御システムにおいて提供されるモードを実現するための機能構成例を示す模式図である。

【図9】本実施の形態に係る制御システムにおいて利用されるPLCランタイムコンテナの構成例を示す模式図である。

10

20

30

40

50

【図１０】本実施の形態に係る制御システムにおけるＰＬＣランタイムコンテナの生成に係る機能構成例を示す模式図である。

【図１１】本実施の形態に係る制御システムにおけるＰＬＣランタイムコンテナのリリースおよび実行管理に係る機能構成例を示す模式図である。

【図１２】本実施の形態に係る制御システムの典型的なユースケースを示す模式図である。

【図１３】図１２（Ａ）に示すユースケースにおける処理手順の一例を概略する模式図である。

【図１４】図１２（Ａ）に示すユースケースにおける別の処理手順の一例を概略する模式図である。

【図１５】図１２（Ｂ）に示すユースケースにおける処理手順の一例を概略する模式図である。

10

【図１６】図１２（Ｂ）に示すユースケースにおける別の処理手順の一例を概略する模式図である。

【図１７】図１２（Ｃ）に示すユースケースにおける処理手順の一例を概略する模式図である。

【図１８】本実施の形態に従う制御システムにおけるＰＬＣランタイムコンテナの生成に関する処理手順の一例を示すフローチャートである。

【図１９】本実施の形態に従う制御システムにおけるＰＬＣランタイムコンテナのリリースおよび実行管理に関する処理手順の一例を示すフローチャートである。

【発明を実施するための形態】

20

【００１９】

本発明の実施の形態について、図面を参照しながら詳細に説明する。なお、図中の同一または相当部分については、同一符号を付してその説明は繰り返さない。

【００２０】

<Ａ．適用例>

まず、本発明が適用される場面の一例について説明する。

【００２１】

図１は、本実施の形態に係る制御システム１の全体構成例を示す模式図である。図１を参照して、本実施の形態に係る制御システム１は、１または複数のＰＬＣシステム２－１，２－２，２－３（以下、「ＰＬＣシステム２」とも総称する。）と、１または複数のサーバ２００と、開発保守端末３００とを含む。以下の説明において、ＰＬＣシステム２－１，２－２，２－３のそれぞれを区別する必要がある場合には、関係する構成要素について、「－１」，「－２」，「－３」といった添え字を付与することもある。

30

【００２２】

ＰＬＣシステム２の各々は、ラインとも称され、工場全体を制御するものであってもよいし、工場に設置された特定の製造設備や製造装置を制御するものであってもよい。ＰＬＣシステム２－１，２－２，２－３は、主たる構成要素として、制御対象を制御する制御装置の典型例であるＰＬＣ（Programmable Logic Controller）１００－１，１００－２，１００－３（以下、「ＰＬＣ１００」とも総称する。）をそれぞれ主体に構成される。

【００２３】

ＰＬＣ１００は、フィールドデバイス１０に接続される。フィールドデバイス１０は、制御対象を制御するために必要な任意の装置を包含する。より具体的には、フィールドデバイス１０は、制御対象（例えば、製造設備、製造装置、製造設備あるいは製造装置に含まれるセンサおよびアクチュエータなど）との間で情報をやり取りするための装置を含む。図１に示す例では、フィールドデバイス１０は、リモートＩ／Ｏ（Input/Output）１２と、リレー群１４と、サーボドライバ１６およびサーボモータ１８とを含む。

40

【００２４】

ＰＬＣ１００は、フィールドデバイス１０から情報を取得し、予め作成されたユーザプログラムに従って制御演算を実行することで、フィールドデバイス１０へ与えられる情報を生成する。以下では、ＰＬＣ１００がフィールドデバイス１０から取得する情報を「入

50

カデータ」とも称し、フィールドデバイス１０へ与えられる情報を「出力データ」とも称す。また、入力データおよび出力データを「入出力データ」あるいは「Ｉ／Ｏデータ」とも総称する。

【００２５】

図１に示す例では、ＰＬＣ１００とフィールドデバイス１０とは、制御系ネットワーク４を介して接続されているが、これに限らず、ハードワイヤーで接続されていてもよい。また、フィールドデバイス１０は、図１に示されるデバイスに限らず、任意のデバイスを包含し得る。

【００２６】

制御系ネットワーク４としては、産業用の通信プロトコルを採用することが好ましい。このような通信プロトコルとしては、EtherCAT（登録商標）、Ethernet／IP（登録商標）、DeviceNet（登録商標）、Component（登録商標）などが知られている。

【００２７】

ＰＬＣシステム２の各々は、情報系ネットワーク６およびゲートウェイ２０を介して、インターネット８上に存在するサーバ２００にアクセス可能になっている。情報系ネットワーク６としては、イーサネット（登録商標）などの汎用的な通信プロトコルを採用してもよい。

【００２８】

サーバ２００は、コンテナをデプロイして実行可能な環境を提供するとともに、コンテナのリリースを管理する。本明細書において、コンテナの「リリース」は、対象のコンテナを実行先に提供する処理および実行先での実行可能化を指示する処理（例えば、デプロイを指示する処理）を包含する。これらの処理および機能の詳細については、後述する。

【００２９】

サーバ２００は、インターネット８を介してクラウド上に配置してもよいし、インターネット８を介することなく、ＰＬＣシステム２が存在するローカルネットワーク内に配置してもよい。サーバ２００の実装形態としては、要求される性能および機能などに応じて、任意に決定できる。

【００３０】

開発保守端末３００は、サーバ２００にアクセスして、後述するような各種処理を実現するためのユーザ操作が可能になっている。

【００３１】

図２は、本実施の形態に係る制御システム１におけるコンテナを利用した処理の概要を説明するための図である。

【００３２】

本明細書においては、ＰＬＣシステム２のＰＬＣ１００で制御演算を実行するために必要なプログラムおよびライブラリなどを「ＰＬＣランタイム」と称す。なお、「ＰＬＣランタイム」との用語は、制御演算の実行に必要なプログラムおよびライブラリなどに加えて、制御演算を実現する実体および実行環境を包含し得る。典型的には、「ＰＬＣランタイム」は、ＰＬＣ１００による制御対象に対する制御演算を定義するユーザプログラム（例えば、IEC 61131-3に従って任意に作成されるプログラム）と、ユーザプログラムを実行する実行環境であるＰＬＣエンジン（ファームウェア）とを含む。

【００３３】

本実施の形態に従う制御システム１においては、ＰＬＣランタイムをＰＬＣ１００だけではなく、サーバ２００が提供する仮想環境においても実現できるようにパッケージ化したものを「ＰＬＣランタイムコンテナ」と称す。「ＰＬＣランタイムコンテナ」をホストＯＳ（Operating System）にデプロイすることで、論理的に独立したアプリケーションの実行環境を実現できる。なお、「ＰＬＣランタイムコンテナ」との用語は、実現されたアプリケーションの実行環境を包含し得る。「ＰＬＣランタイムコンテナ」を構成するプログラム（アプリケーション）およびライブラリなどの詳細、ならびに、「ＰＬＣランタ

10

20

30

40

50

イムコンテナ」の生成方法の詳細などについては、後述する。

【0034】

図2を参照して、PLCランタイムコンテナ50は、PLCシステム2のPLC100およびサーバ200のいずれにもリリース可能である。要求される状況に応じて、PLCランタイムコンテナ50のデプロイ先およびデプロイ内容を最適化することで、少なくとも、(1)ローカル実行モード、(2)クラウド実行モード、(3)シミュレーションモードの3つのモードを実現できる。

【0035】

図3は、本実施の形態に係る制御システム1において提供されるモードを説明するための図である。

【0036】

図3(A)を参照して、(1)ローカル実行モードは、PLC100がフィールドデバイス10を介してやり取りされるI/Oデータを利用して制御演算を実行する状態に相当する。より具体的には、PLC100にPLCランタイム510および物理I/O160が実現される。物理I/O160は、フィールドデバイス10および制御系ネットワーク4(いずれも図1参照)も含む抽象化されたレイヤである。ローカル実行モードは、いわゆる実運用に相当する。

【0037】

図3(B)を参照して、(2)クラウド実行モードは、PLC100に接続されたフィールドデバイス10を介してI/Oデータをやり取りする一方で、制御演算自体は、サーバ200で実行する状態に相当する。より具体的には、PLC100に物理I/O160が実現されるとともに、サーバ200にPLCランタイム510が実現される。クラウド実行モードは、実運用として用いることもできるし、ユーザプログラムの開発やチェックなどに用いることもできる。

【0038】

図3(C)を参照して、(3)シミュレーションモードは、PLC100に接続されたフィールドデバイス10を必要とせず、サーバ200がフィールドデバイス10をエミュレートしつつ、サーバ200が制御演算も実行する状態に相当する。より具体的には、サーバ200にPLCランタイム510および仮想I/Oエミュレータ550が実現される。仮想I/Oエミュレータ550は、物理I/O160の挙動をエミュレートするモジュールである。シミュレーションモードは、ユーザプログラムの開発やチェックに用いることができる。

【0039】

このように、本実施の形態に従う制御システム1においては、PLCランタイムコンテナ50を利用することで、PLCシステム2の一部または全部が存在しない状態であっても、ユーザプログラムの開発やチェックを容易に行うことができるとともに、シームレスで実運用へ移行することもできる。

【0040】

< B. ハードウェア構成 >

次に、本実施の形態に係る制御システムを構成する装置のハードウェア構成の一例について説明する。

【0041】

(b1: PLC100)

図4は、本実施の形態に係る制御システム1を構成するPLC100のハードウェア構成例を示すブロック図である。図4を参照して、PLC100は、CPU(Central Processing Unit)やMPU(Micro-Processing Unit)などのプロセッサ102と、チップセット104と、主メモリ106と、ストレージ110と、情報系ネットワークコントローラ120と、制御系ネットワークコントローラ122と、USB(Universal Serial Bus)コントローラ124と、メモ리카ードインターフェイス126とを含む。

【0042】

10

20

30

40

50

プロセッサ 102 は、ストレージ 110 に格納された各種プログラムを読み出して、主メモリ 106 に展開して実行することで、制御対象を制御するための制御演算を実現する。チップセット 104 は、プロセッサ 102 と各コンポーネントとのデータ伝送などを制御する。

【0043】

ストレージ 110 には、典型的には、基本的な処理を実現するための OS 112 と、PLC ランタイムコンテナのデプロイなどの処理を行うシステムプログラム 114 とが格納される。

【0044】

情報系ネットワークコントローラ 120 は、情報系ネットワーク 6 を介した開発保守端末 300 などとのデータのやり取りを制御する。

【0045】

制御系ネットワークコントローラ 122 と、制御系ネットワーク 4 を介したフィールドデバイス 10 とのデータのやり取りを制御する。

【0046】

USB コントローラ 124 は、USB 接続を介した外部装置（例えば、サポート装置）とのデータのやり取りを制御する。

【0047】

メモリカードインターフェイス 126 は、メモリカード 128 を着脱可能に構成されており、メモリカード 128 に対してデータを書き込み、メモリカード 128 から各種データ（ユーザプログラムやトレースデータなど）を読み出すことが可能になっている。

【0048】

図 4 には、プロセッサ 102 がプログラムを実行することで必要な処理が提供される構成例を示したが、これらの提供される処理の一部または全部を、専用のハードウェア回路（例えば、ASIC（Application Specific Integrated Circuit）または FPGA（Field-Programmable Gate Array）など）を用いて実装してもよい。あるいは、PLC 100 の主要部を、汎用的なアーキテクチャに従うハードウェア（例えば、汎用パソコンをベースとした産業用パソコン）を用いて実現してもよい。この場合には、仮想化技術を用いて、用途の異なる複数の OS を並列的に実行させるとともに、各 OS 上で必要なアプリケーションを実行させるようにしてもよい。

【0049】

（b2：サーバ 200）

図 5 は、本実施の形態に係る制御システム 1 を構成するサーバ 200 のハードウェア構成例を示すブロック図である。図 5 を参照して、サーバ 200 は、CPU や MPU などの 1 または複数のプロセッサ 202 と、主メモリ 206 と、ストレージ 210 と、1 または複数のネットワークコントローラ 220 と、入力部 226 と、表示部 228 とを含む。これらのコンポーネントは、バス 208 を介して接続される。

【0050】

プロセッサ 202 は、ストレージ 210 に格納された各種プログラムを読み出して、主メモリ 206 に展開して実行することで、後述するような各種処理を実現する。

【0051】

ストレージ 210 は、例えば、HDD（Hard Disk Drive）や SSD（Flash Solid State Drive）などで構成される。ストレージ 210 には、典型的には、基本的な処理を実現するための OS 212 と、後述するようなサーバ 200 が提供する各種処理を実現するためのコンテナアプリケーション 214 とが格納される。

【0052】

ネットワークコントローラ 220 は、ネットワークを介した各種装置とのデータのやり取りを制御する。

【0053】

入力部 226 は、キーボード、マウスなどで構成され、ユーザからの指示を受け付ける

10

20

30

40

50

。表示部 228 は、ディスプレイなどで構成され、プロセッサ 202 からの処理結果などを出力する。

【0054】

図 5 には、プロセッサ 202 がプログラムを実行することで必要な処理が提供される構成例を示したが、これらの提供される処理の一部または全部を、専用のハードウェア回路（例えば、ASIC または FPGA など）を用いて実装してもよい。

【0055】

（b3：開発保守端末 300）

図 6 は、本実施の形態に係る制御システム 1 を構成する開発保守端末 300 のハードウェア構成例を示すブロック図である。図 6 を参照して、開発保守端末 300 は、CPU や MPU などのプロセッサ 302 と、主メモリ 306 と、ストレージ 310 と、ネットワークコントローラ 320 と、USB コントローラ 324 と、入力部 326 と、表示部 328 とを含む。これらのコンポーネントは、バス 308 を介して接続される。

10

【0056】

プロセッサ 302 は、ストレージ 310 に格納された各種プログラムを読み出して、主メモリ 306 に展開して実行することで、開発保守端末 300 で必要な処理を実現する。

【0057】

ストレージ 310 は、例えば、HDD や SSD など構成される。ストレージ 310 には、典型的には、OS 312 と、後述するような処理を指示などするための開発プログラム 314 とが格納される。なお、ストレージ 310 には、図 6 に示すプログラム以外の必要なプログラムが格納されてもよい。

20

【0058】

ネットワークコントローラ 320 は、任意の制御系ネットワーク 4 を介した PLC 100 などとのデータのやり取りを制御する。

【0059】

USB コントローラ 324 は、USB 接続を介した外部装置とのデータのやり取りを制御する。

【0060】

入力部 326 は、マウス、キーボード、タッチパネルなどで構成され、ユーザからの指示を受け付ける。表示部 228 は、ディスプレイ、各種インジケータなどで構成され、プロセッサ 202 からの処理結果などを出力する。

30

【0061】

開発保守端末 300 は、光学ドライブ 304 を有していてもよい。光学ドライブ 304 は、コンピュータ読取可能なプログラムを非一過的に格納する記録媒体 305（例えば、DVD（Digital Versatile Disc）などの光学記録媒体）からプログラムを読み取って、ストレージ 310 などに格納する。

【0062】

開発保守端末 300 で実行される各種プログラムは、コンピュータ読取可能な記録媒体 305 を介してインストールされてもよいが、ネットワーク上の任意のサーバからダウンロードする形でインストールするようにしてもよい。

40

【0063】

図 6 には、プロセッサ 302 がプログラムを実行することで必要な処理が提供される構成例を示したが、これらの提供される処理の一部または全部を、専用のハードウェア回路（例えば、ASIC または FPGA など）を用いて実装してもよい。

【0064】

< C. 制御システム 1 の機能構成 >

図 7 は、本実施の形態に係る制御システム 1 における機能構成の一例を示す模式図である。図 7 を参照して、PLC 100 の各々は、コンテナエンジン 150 を有している。コンテナエンジン 150 は、コンテナの実行環境を提供するものであり、PLC ランタイムコンテナ 50 に含まれるマニフェストなどに基づいて、モジュールのイメージをデプロイ

50

して、処理を実行するためのモジュール（インスタンス）を生成する。

【0065】

一方、サーバ200は、コンテナエンジン250と、コンテナ生成モジュール260と、オーケストレータ270とを有している。

【0066】

コンテナエンジン250は、コンテナエンジン150と同様に、PLCランタイムコンテナ50に含まれるマニフェストなどに基づいて、モジュールのイメージをデプロイして、処理を実行するためのモジュール（インスタンス）を生成する。

【0067】

コンテナエンジン150, 250は、CaaS（Containers as a Service）としての基盤である。PLCランタイムコンテナ50をリリースする文脈においては、コンテナエンジン150, 250の各々を「ノード」と称することもある。

【0068】

コンテナ生成モジュール260は、ユーザ操作に応じて、PLCランタイムコンテナ50を生成する。本実施の形態においては、コンテナ生成モジュール260は、PLC100で実行されるべきユーザプログラムおよび関連する設定情報などから、PLCランタイムのイメージを含むPLCランタイムコンテナ50を生成する。すなわち、コンテナ生成モジュール260は、制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行する実行環境とを内包するPLCランタイムのイメージを含むPLCランタイムコンテナ50を生成する。

【0069】

オーケストレータ270は、PLCランタイムコンテナ50を管理する処理を担当する。オーケストレータ270は、コンテナ生成モジュール260により生成されたPLCランタイムコンテナ50をコンテナハブに登録するなどの処理を行うとともに、ユーザ操作などに応じて、コンテナハブに登録されたPLCランタイムコンテナ50をいずれかのホスト（本実施の形態においては、PLC100のコンテナエンジン150あるいはサーバ200のコンテナエンジン250）で実行するのかを管理する処理などを行う。このように、オーケストレータ270は、PLC100およびサーバ200の一方または両方に対してPLCランタイムコンテナ50をリリースする。

【0070】

PLC100およびサーバ200は、PLCランタイムコンテナ50をデプロイするコンテナエンジン150および250をそれぞれ有しており、オーケストレータ270によりリリースされたPLCランタイムコンテナ50をデプロイすることで、PLCランタイムのイメージのインスタンスを生成して制御演算を実行することになる。

【0071】

さらに、オーケストレータ270は、PLCランタイムコンテナ50の起動／停止の管理、ホスト間のネットワーク接続の管理、ストレージの管理、PLCランタイムコンテナ50を実行させるホストのスケジューリング、PLCランタイムコンテナ50のホストでの実行状態の管理などを担当する。

【0072】

なお、オーケストレータ270によるPLCランタイムコンテナ50のリリースは、予め定められたスケジュールに従って行われてもよいし、ユーザ操作に従って、リリースするPLCランタイムコンテナ50の選択およびリリース先の決定を行ってもよい。

【0073】

それぞれの構成要素による処理の詳細については、後述する。

< D. PLCランタイムコンテナの利用 >

次に、PLCランタイムコンテナ50を利用して、図2および図3に示すようなそれぞれのモードを実現する例を説明する。以下の説明では、主として、PLC100のコンテナエンジン150あるいはサーバ200のコンテナエンジン250により提供される機能および処理について説明する。

10

20

30

40

50

【0074】

図8は、本実施の形態に係る制御システム1において提供されるモードを実現するための機能構成例を示す模式図である。

【0075】

図8(A)を参照して、PLC100は、ローカルOS140およびコンテナエンジン150を有している。

【0076】

(1) ローカル実行モードにおいては、コンテナエンジン150上にPLCランタイム510のインスタンスが生成される。また、PLC100には、PLCランタイム510と物理I/O160とのやり取りを実現するための物理I/Oドライバ142も用意される。図8(A)に示すような機能構成によって、PLC100による制御演算が実現される。

【0077】

図8(B)を参照して、(2)クラウド実行モードでは、PLC100のコンテナエンジン150上には、PLCランタイム510に代えて、仮想I/Oネットワークサーバ540のインスタンスが生成される。仮想I/Oネットワークサーバ540は、サーバ200と物理I/O160のやり取りを仲介する処理を担当する。仮想I/Oネットワークサーバ540は、物理I/O160を含むI/O処理の結果を仮想I/Oネットワーククライアント530との間で同期する。

【0078】

このように、サーバ200のコンテナエンジン250がPLCランタイム510のイメージのインスタンスを生成する場合に、PLC100のコンテナエンジン150はPLC100に接続されたフィールドデバイス10を介してやり取りされるI/Oデータをサーバ200に仲介する仮想I/Oネットワークサーバ540のインスタンスを生成する。

【0079】

一方、サーバ200は、サーバOS240およびコンテナエンジン250を有している。(2)クラウド実行モードにおいては、コンテナエンジン250上にPLCランタイム510のインスタンスが生成される。さらに、コンテナエンジン250上には、仮想I/Oドライバ520および仮想I/Oネットワーククライアント530のインスタンスが生成される。

【0080】

このように、(2)クラウド実行モードにおいては、サーバ200において、PLCランタイム510のインスタンスを生成するとともに、PLC100においても、仮想I/Oネットワークサーバ540のインスタンスを生成する。

【0081】

仮想I/Oネットワーククライアント530は、仮想I/Oネットワークサーバ540との間で、I/Oデータのやり取りを行う。

【0082】

仮想I/Oドライバ520は、図8(A)に示すPLC100の物理I/Oドライバ142をエミュレートするものであり、PLCランタイム510からのI/Oデータにアクセスするための内部コマンドに応答して、PLC100の物理I/O160へアクセスすると同様の処理を行う。このように、サーバ200のコンテナエンジン250がPLCランタイム510のイメージのインスタンスを生成する場合に、PLC100に接続されたフィールドデバイス10へアクセスする処理を担当する物理I/Oドライバ142をエミュレートする仮想I/Oドライバ520のインスタンスを生成する。

【0083】

仮想I/Oネットワークサーバ540と仮想I/Oネットワーククライアント530との間のやり取りは、PLCランタイム510に関して透過性を有している。すなわち、PLC100で実行されるPLCランタイム510およびサーバ200で実行されるPLCランタイム510において、物理I/O160へのアクセス命令は同一である。仮想I/O

10

20

30

40

50

Ｏドライバ５２０、仮想Ｉ／Ｏネットワーククライアント５３０および仮想Ｉ／Ｏネットワークサーバ５４０は、当該同一のアクセス命令に応答してＰＬＣ１００の物理Ｉ／Ｏドライバ１４２が実行する処理と実質的に同一の処理を実現する。これらの処理の同一性によって、ＰＬＣランタイム５１０を図８（Ａ）～図８（Ｃ）のように互いに異なる実行環境上で動作させても、処理内容をそれぞれ異ならせる必要がない。

【００８４】

図８（Ｃ）を参照して、（３）シミュレーションモードでは、コンテナエンジン２５０上に、仮想Ｉ／Ｏエミュレータ５５０および仮想Ｉ／Ｏネットワークサーバ５４０のインスタンスが生成される。

【００８５】

仮想Ｉ／Ｏエミュレータ５５０は、ＰＬＣ１００の物理Ｉ／Ｏ１６０を模擬するモジュールであり、ＰＬＣランタイム５１０からの出力データに응答して、入力データを生成するといった処理を実行する。なお、仮想Ｉ／Ｏエミュレータ５５０は、ＰＬＣランタイム５１０を用いたシミュレーションを実現できればよいので、ＰＬＣランタイム５１０から与えられた出力データをそのまま入力データとして戻す処理（いわゆる、折り返し処理）を実行してもよい。あるいは、仮想Ｉ／Ｏエミュレータ５５０は、例えば、サーボドライバなどのモデルを有しており、ＰＬＣランタイム５１０から与えられた出力データを当該モデルに入力して得られる結果を入力データとして응答してもよい。

【００８６】

このように、（３）シミュレーションモードでは、制御装置２００のコンテナエンジン２５０は、ＰＬＣランタイム５１０のインスタンスとともに、制御対象とやり取りされるＩ／Ｏデータをエミュレートする仮想Ｉ／Ｏエミュレータ５５０のインスタンスも生成する。

【００８７】

仮想Ｉ／Ｏネットワークサーバ５４０は、図８（Ｂ）に示すＰＬＣ１００の仮想Ｉ／Ｏネットワークサーバ５４０と同様の処理を実行する。すなわち、仮想Ｉ／Ｏネットワーククライアント５３０と仮想Ｉ／Ｏネットワークサーバ５４０との間でＩ／Ｏデータをやり取りするための通信路を仮想的に形成する。仮想Ｉ／Ｏネットワークサーバ５４０は、仮想Ｉ／Ｏエミュレータ５５０によるＩ／Ｏ処理の結果を仮想Ｉ／Ｏネットワーククライアント５３０との間で同期する。

【００８８】

なお、モードに応じて、仮想Ｉ／Ｏネットワーククライアント５３０の接続先は、ＰＬＣ１００の仮想Ｉ／Ｏネットワークサーバ５４０またはサーバ２００の仮想Ｉ／Ｏネットワークサーバ５４０に適宜変更される。

【００８９】

図９は、本実施の形態に係る制御システム１において利用されるＰＬＣランタイムコンテナ５０の構成例を示す模式図である。図９を参照して、ＰＬＣランタイムコンテナ５０は、ＰＬＣランタイムイメージ５１と、仮想Ｉ／Ｏドライバイメージ５２と、仮想Ｉ／Ｏネットワーククライアントイメージ５３と、仮想Ｉ／Ｏネットワークサーバイメージ５４と、仮想Ｉ／Ｏエミュレータイメージ５５とを含む。これらのイメージは、デプロイされることで必要な処理を実行可能形式のデータである。

【００９０】

ＰＬＣランタイムコンテナ５０は、マニフェスト５６をさらに含む。マニフェスト５６は、各実行環境および各モードにおいて、いずれのイメージを有効化するのか、および、どのようなリソースを利用するのかといった定義を含む。

【００９１】

コンテナエンジン１５０，２５０は、マニフェスト５６を参照して、ＰＬＣランタイムコンテナ５０に含まれる各モジュールのイメージのうち、必要なイメージを選択する。典型的には、マニフェスト５６は、図８（Ａ）～図８（Ｃ）に示されるそれぞれのモードで利用される機能の定義を含む。

10

20

30

40

50

【0092】

図9に示すように、PLCランタイムコンテナ50は、PLC100のコンテナエンジン150のみがデプロイするイメージ（例えば、仮想I/Oネットワークサーバイメージ54）を含んでいてもよい。また、PLCランタイムコンテナ50は、サーバ200のコンテナエンジン250のみがデプロイするイメージ（例えば、仮想I/Oドライバイメージ52、仮想I/Oネットワーククライアントイメージ53、仮想I/Oエミュレータイメージ55など）を含んでいてもよい。

【0093】

なお、図9には、同一のPLCランタイムコンテナ50を利用する構成例を示すが、リリース先および選択されるモードに応じて、PLCランタイムコンテナ50の構成を異ならせてもよい。すなわち、選択されたモードにおいて必要なイメージおよび対応するマニフェスト56を含むPLCランタイムコンテナ50を生成およびリリースするようにしてもよい。

【0094】

< E. PLCランタイムコンテナの生成 >

次に、PLCランタイムコンテナ50の生成に係る機能および処理について説明する。

【0095】

図10は、本実施の形態に係る制御システム1におけるPLCランタイムコンテナ50の生成に係る機能構成例を示す模式図である。図10を参照して、サーバ200のコンテナ生成モジュール260（図7）は、開発環境262と、システム構成マネージャ264とを含む。

【0096】

開発環境262は、ユーザ操作に従って、PLCシステム2で実行されるユーザプログラム266、および、PLCシステム2の構成を定義するPLCシステム構成情報268を生成および出力する。典型的には、開発環境262は、エディタ機能、デバッグ機能、シミュレーション機能、整合性チェック機能などを含む。

【0097】

開発環境262は、サーバ200上に実現されてもよいし、開発保守端末300が開発プログラムを実行することで実現されてもよい。

【0098】

ユーザは、開発環境262上で、制御システム1に含まれるPLCシステム2の構成を定義する。定義される内容としては、PLC100の台数およびサーバ200上で利用するリソースなどを含む。また、ユーザは、開発環境262上で、制御対象に応じたユーザプログラム266を作成する。

【0099】

また、ユーザは、開発環境262を介して、PLCランタイムコンテナ50のリリースやデプロイの指示を行うこともできる。

【0100】

システム構成マネージャ264は、ユーザ操作に従って、ユーザプログラム266およびPLCシステム構成情報268に基づいて、PLCランタイムコンテナ50を生成する。PLCランタイムコンテナ50は、各モジュールのイメージに加えて、マニフェスト56を含む。

【0101】

システム構成マネージャ264は、PLCランタイムコンテナ50の生成に加えて、システム構成プロファイル60を生成および出力する。システム構成プロファイル60は、PLCランタイムコンテナ50のリリース先およびPLC100の識別情報などを含む。

【0102】

また、システム構成プロファイル60は、利用するコンテナハブ276の識別情報、利用するPLCランタイムコンテナ50の識別情報、仮想ノードの使用可否を示す情報、利用する仮想ノードの最大数などを含んでいてもよい。このとき、システム構成プロファイ

10

20

30

40

50

ル 6 0 は、P L C ランタイムコンテナ 5 0 に関する情報が登録されるリポジトリの情報も含まれる。

【 0 1 0 3 】

なお、システム構成プロファイル 6 0 は、制御システム 1 毎に生成されてもよいし、制御システム 1 に含まれる P L C システム 2 毎に生成されてもよい。

【 0 1 0 4 】

システム構成プロファイル 6 0 は、コンテナ構成プロファイルを含んでいてもよい。コンテナ構成プロファイルは、P L C ランタイムコンテナ 5 0 を特定するための識別情報と、P L C ランタイムコンテナ 5 0 が対象とする P L C 1 0 0 および各 P L C 1 0 0 に接続されるデバイス（モータやロボットなど）などの情報を含む。また、コンテナ構成プロファイルは、デバイスを制御するにあたって、開発環境上で追加の必要機能（以下、「機能モジュール」とも称す。）の要否などの情報を含んでいてもよい。

【 0 1 0 5 】

< F . P L C ランタイムコンテナのリリースおよび実行管理 >

次に、P L C ランタイムコンテナ 5 0 のリリースおよび実行管理に係る機能および処理について説明する。

【 0 1 0 6 】

図 1 1 は、本実施の形態に係る制御システム 1 における P L C ランタイムコンテナ 5 0 のリリースおよび実行管理に係る機能構成例を示す模式図である。図 1 1 を参照して、サーバ 2 0 0 のオーケストレータ 2 7 0 （図 7）は、コンテナマネージャ 2 7 2 と、ノードマネージャ 2 7 4 とを含む。

【 0 1 0 7 】

コンテナマネージャ 2 7 2 は、P L C ランタイムコンテナ 5 0 のリリース、デプロイ、実行などを管理する。コンテナマネージャ 2 7 2 は、P L C ランタイムコンテナ 5 0 を登録するコンテナハブ 2 7 6 と、コンテナハブ 2 7 6 に登録された P L C ランタイムコンテナ 5 0 の状態を管理するリポジトリ 2 7 8 とを管理する。

【 0 1 0 8 】

コンテナマネージャ 2 7 2 は、コンテナ生成モジュール 2 6 0 （図 1 0 参照）から P L C ランタイムコンテナ 5 0 が提供されると、付随するシステム構成プロファイル 6 0 を参照して、P L C ランタイムコンテナ 5 0 をコンテナハブ 2 7 6 に登録するとともに、リポジトリ 2 7 8 に対応するエントリを追加する。

【 0 1 0 9 】

コンテナハブ 2 7 6 は、コンテナ生成モジュール 2 6 0 により生成される P L C ランタイムコンテナ 5 0 をリリース可能な P L C 1 0 0 の識別情報と関連付けて登録する。

【 0 1 1 0 】

リポジトリ 2 7 8 の各エントリは、コンテナハブ 2 7 6 に登録されている P L C ランタイムコンテナ 5 0 を特定するためのコンテナ名と、ノード名とノードアドレスとを対応付ける名前空間と、P L C ランタイムコンテナ 5 0 の状態を示すコンテナ状態とを含む。名前空間は、P L C ランタイム 5 1 0 からのノード名の指定と、対応するノードアドレスとをマッピングするものである。このように、リポジトリ 2 7 8 は、サーバ 2 0 0 のコンテナエンジン 2 5 0 が P L C ランタイム 5 1 0 のインスタンスを生成する場合に、コンテナエンジン 2 5 0 にアクセスするためのアドレスを管理する。

【 0 1 1 1 】

ノードマネージャ 2 7 4 は、コンテナマネージャ 2 7 2 からの要求などに応じて、P L C ランタイムコンテナ 5 0 のリリース先のノード（P L C 1 0 0 のコンテナエンジン 1 5 0 あるいはサーバ 2 0 0 のコンテナエンジン 2 5 0）の探索、ノード状態を取得、仮想的なノードの生成などを行う。

【 0 1 1 2 】

仮想的なノードの生成に関して、ノードマネージャ 2 7 4 は、サーバ 2 0 0 に対して P L C ランタイムコンテナ 5 0 をリリースする場合に、P L C ランタイムコンテナ 5 0 をデ

10

20

30

40

50

プロイするコンテナエンジン 250 が存在しなければ、コンテナエンジン 250 を含むノードを仮想的に生成する。

【0113】

＜G. ユースケース＞

次に、いくつかのユースケースについて説明する。

【0114】

図 12 は、本実施の形態に係る制御システム 1 の典型的なユースケースを示す模式図である。図 12 に示すユースケースでは、PLC100-1, 100-2, 100-3 が制御対象を制御する場合を想定する。

【0115】

図 12 (A) に示すユースケース 1 では、PLC100-1, 100-2, 100-3 が実行する制御演算のすべてをサーバ 200 でシミュレーションする例を示す。すなわち、PLC100-1, 100-2, 100-3 のいずれについても、シミュレーションモードが実現される。この場合、PLC100-1, 100-2, 100-3 は必ずしも実在しなくてもよい。

【0116】

図 12 (B) に示すユースケース 2 では、PLC100-1 が実行する制御演算を、PLC100-1 の物理 I/O160-1 を利用しつつ、サーバ 200 上で実行する例を示す。すなわち、PLC100-1 については、クラウド実行モードが実現される。なお、PLC100-2, 100-3 については、図 12 (A) と同様に、シミュレーションモードが実現される。この場合、PLC100-2, 100-3 は必ずしも実在しなくてもよい。

【0117】

図 12 (C) に示すユースケース 3 では、PLC100-2 が自装置の物理 I/O160-2 を利用して実行する例を示す。すなわち、PLC100-1 については、ローカル実行モードが実現される。なお、PLC100-1 については、図 12 (B) と同様に、クラウド実行モードが実現され、PLC100-3 については、図 12 (A) と同様に、シミュレーションモードが実現される。この場合、PLC100-3 は必ずしも実在しなくてもよい。

【0118】

(g1: ユースケース 1)

図 13 は、図 12 (A) に示すユースケース 1 における処理手順の一例を概略する模式図である。図 13 には、ユーザプログラム 266 を含む PLC ランタイム 510 (PLC ランタイムコンテナ 50) を新規作成する場合を示す。また、PLC ランタイムコンテナ 50 は、シミュレーションモードでリリースされる。

【0119】

図 13 を参照して、ユーザは、設計する PLC システム 2 の構成の定義を入力する ((1) 構成定義)。すると、システム構成マネージャ 264 は、入力された PLC システム 2 に対応するシステム構成プロファイル 60 を生成する ((2) システム構成プロファイル生成)。

【0120】

続いて、ユーザは、開発環境 262 を利用してユーザプログラム 266 を作成する ((3) ユーザプログラム作成)。そして、システム構成マネージャ 264 は、PLC ランタイムコンテナ 50 を生成する ((4) PLC ランタイムコンテナ生成)。さらに、コンテナマネージャ 272 は、生成された PLC ランタイムコンテナ 50 をコンテナハブ 276 に登録する ((5) PLC ランタイムコンテナ登録)。以上の処理によって、PLC ランタイムコンテナ 50 の生成および登録が完了する。

【0121】

ユーザがいずれかの PLC ランタイムコンテナ 50 のリリース (この場合には、シミュレーションモード) を指示すると ((6) リリース指示)、ノードマネージャ 274 は、

10

20

30

40

50

リリース先となる仮想ノードを生成する（（７）仮想ノード生成）とともに、生成した仮想ノードのノードアドレスをノード名と対応付けてリポジトリ２７８に登録する（（８）ノード名登録）。そして、選択されたＰＬＣランタイムコンテナ５０がリリース先の仮想ノードへ送信される（（９）ＰＬＣランタイムコンテナリリース）。仮想ノードのコンテナエンジン２５０－１は、ＰＬＣランタイムコンテナ５０をデプロイする（（１０）ＰＬＣランタイムコンテナデプロイ）。仮想ノードのコンテナエンジン２５０－１は、デプロイしたＰＬＣランタイムコンテナ５０のコンテナ名をリポジトリ２７８に登録する（（１１）コンテナ名登録）。仮想ノードのコンテナエンジン２５０－１は、リポジトリ２７８の内容を参照することで、Ｉ／Ｏデータのやり取りにおいて、送信先または受信先のノード名に対応するノードアドレスを解決できる。

10

【０１２２】

また、仮想ノードのコンテナエンジン２５０－１は、デプロイして生成されたＰＬＣランタイムコンテナ５０のインスタンスの内容をコンテナハブ２７６に反映する（（１２）デプロイ内容反映）。このように、コンテナハブ２７６に反映することで、仮想ノードで次回起動する際に再利用が容易になる。

【０１２３】

図１４は、図１２（Ａ）に示すユースケース１における別の処理手順の一例を概略する模式図である。図１４には、図１３に示すような処理手順で作成したＰＬＣランタイムコンテナ５０を再利用する場合を示す。また、ＰＬＣランタイムコンテナ５０は、シミュレーションモードでリリースされる。

20

【０１２４】

図１４を参照して、ユーザは、作成済のＰＬＣシステム２の再利用を指示する（（１）再利用指示）。すると、システム構成マネージャ２６４は、指示されたＰＬＣシステム２に対応するシステム構成プロファイル６０をロードする（（２）システム構成プロファイルロード）。

【０１２５】

続いて、ユーザは、開発環境２６２を利用してユーザプログラム２６６を更新する（（３）ユーザプログラム更新）。そして、システム構成マネージャ２６４は、ＰＬＣランタイムコンテナ５０を更新する（（４）ＰＬＣランタイムコンテナ更新）。さらに、コンテナマネージャ２７２は、更新されたＰＬＣランタイムコンテナ５０をコンテナハブ２７６に登録する（（５）ＰＬＣランタイムコンテナ登録）。以上の処理によって、ＰＬＣランタイムコンテナ５０の更新（再生成）および登録が完了する。

30

【０１２６】

ユーザがいずれかのＰＬＣランタイムコンテナ５０のリリース（この場合には、シミュレーションモード）を指示すると（（６）リリース指示）、選択されたＰＬＣランタイムコンテナ５０がリリース先の仮想ノードへ送信される（（７）ＰＬＣランタイムコンテナリリース）。この例では、既に仮想ノードにおいてＰＬＣランタイムが実現されている場合を想定しているため、リリース先となる仮想ノードを新たに生成する必要はない。

【０１２７】

仮想ノードのコンテナエンジン２５０－１は、ＰＬＣランタイムコンテナ５０をデプロイする（（８）ＰＬＣランタイムコンテナデプロイ）。これによって、更新後のユーザプログラムに対応するＰＬＣランタイムが実現される。なお、多くの場合において、リポジトリ２７８の内容を更新する必要はない。

40

【０１２８】

また、仮想ノードのコンテナエンジン２５０－１は、デプロイして生成されたＰＬＣランタイムコンテナ５０のインスタンスの内容をコンテナハブ２７６に反映する（（９）デプロイ内容反映）。

【０１２９】

（ｇ２：ユースケース２）

図１５は、図１２（Ｂ）に示すユースケース２における処理手順の一例を概略する模式

50

図である。図 15 には、ユーザプログラム 266 を含む PLC ランタイム 510 (PLC ランタイムコンテナ 50) を新規作成する場合を示す。また、PLC ランタイムコンテナ 50 は、クラウド実行モードでリリースされる。

【0130】

図 15 を参照して、ユーザは、設計する PLC システム 2 の構成の定義を入力する ((1) 構成定義)。すると、システム構成マネージャ 264 は、入力された PLC システム 2 に対応するシステム構成プロファイル 60 を生成する ((2) システム構成プロファイル生成)。

【0131】

続いて、ユーザは、開発環境 262 を利用してユーザプログラム 266 を作成する ((3) ユーザプログラム作成)。そして、システム構成マネージャ 264 は、PLC ランタイムコンテナ 50 を生成する ((4) PLC ランタイムコンテナ生成)。さらに、コンテナマネージャ 272 は、生成された PLC ランタイムコンテナ 50 をコンテナハブ 276 に登録する ((5) PLC ランタイムコンテナ登録)。以上の処理によって、PLC ランタイムコンテナ 50 の生成および登録が完了する。

【0132】

ユーザがいずれかの PLC ランタイムコンテナ 50 のリリース (この場合には、クラウド実行モード) を指示すると ((6) リリース指示)、ノードマネージャ 274 は、リリース先となる仮想ノードを生成する ((7) 仮想ノード生成) とともに、生成した仮想ノードのノードアドレスをノード名と対応付けてリポジトリ 278 に登録する ((8) ノード名登録)。

【0133】

そして、選択された PLC ランタイムコンテナ 50 がリリース先の仮想ノードへ送信される ((9) PLC ランタイムコンテナリリース)。仮想ノードのコンテナエンジン 250-1 は、PLC ランタイムコンテナ 50 をデプロイする ((10) PLC ランタイムコンテナデプロイ)。コンテナエンジン 250-1 によるデプロイによって、PLC ランタイム 510、仮想 I/O ドライバ 520 および仮想 I/O ネットワーククライアント 530 のインスタンスが生成される (図 8 (B) 参照)。

【0134】

また、選択された PLC ランタイムコンテナ 50 はリリース先の PLC 100-1 へも送信される ((11) PLC ランタイムコンテナリリース)。PLC 100-1 のコンテナエンジン 150 は、PLC ランタイムコンテナ 50 をデプロイする ((12) PLC ランタイムコンテナデプロイ)。コンテナエンジン 150 によるデプロイによって、仮想 I/O ネットワークサーバ 540 のインスタンスが生成される。

【0135】

仮想ノードのコンテナエンジン 250-1 は、デプロイした PLC ランタイムコンテナ 50 のコンテナ名をリポジトリ 278 に登録する ((13) コンテナ名登録)。また、仮想ノードのコンテナエンジン 250-1 は、デプロイして生成された PLC ランタイムコンテナ 50 のインスタンスの内容をコンテナハブ 276 に反映する ((14) デプロイ内容反映)。

【0136】

図 16 は、図 12 (B) に示すユースケース 2 における別の処理手順の一例を概略する模式図である。図 16 には、PLC ランタイム 510 をローカル実行モードでリリースしたものの、PLC 100 のリソース不足により、クラウド実行モードに変更する場合を示す。

【0137】

図 16 を参照して、ユーザは、設計する PLC システム 2 の構成の定義を入力する ((1) 構成定義)。すると、システム構成マネージャ 264 は、入力された PLC システム 2 に対応するシステム構成プロファイル 60 を生成する ((2) システム構成プロファイル生成)。

10

20

30

40

50

【0138】

続いて、ユーザは、開発環境262を利用してユーザプログラム266を作成する（（3）ユーザプログラム作成）。そして、システム構成マネージャ264は、PLCランタイムコンテナ50を生成する（（4）PLCランタイムコンテナ生成）。さらに、コンテナマネージャ272は、生成されたPLCランタイムコンテナ50をコンテナハブ276に登録する（（5）PLCランタイムコンテナ登録）。以上の処理によって、PLCランタイムコンテナ50の生成および登録が完了する。

【0139】

ユーザがいずれかのPLCランタイムコンテナ50のリリース（この場合には、ローカル実行モード）を指示すると（（6）リリース指示）、選択されたPLCランタイムコンテナ50はリリース先のPLC100-1へ送信される（（7）PLCランタイムコンテナリリース）。PLC100-1のコンテナエンジン150は、PLCランタイムコンテナ50をデプロイする（（8）PLCランタイムコンテナデプロイ）。ここで、PLC100-1の容量オーバーにより、PLCランタイムコンテナ50のデプロイに失敗したとする。

10

【0140】

ここで、システム構成マネージャ264は、ローカル実行モードからクラウド実行モードに処理を変更したとする。すると、ノードマネージャ274は、リリース先となる仮想ノードを生成する（（9）仮想ノード生成）とともに、生成した仮想ノードのノードアドレスをノード名と対応付けてリポジトリ278に登録する（（10）ノード名登録）。

20

【0141】

そして、選択されたPLCランタイムコンテナ50がリリース先の仮想ノードへ送信される（（11）PLCランタイムコンテナリリース）。仮想ノードのコンテナエンジン250-1は、PLCランタイムコンテナ50をデプロイする（（12）PLCランタイムコンテナデプロイ）。

【0142】

また、選択されたPLCランタイムコンテナ50はリリース先のPLC100-1へも再度送信される（（13）PLCランタイムコンテナリリース）。PLC100-1のコンテナエンジン150は、PLCランタイムコンテナ50をクラウド実行モードでデプロイする（（14）PLCランタイムコンテナデプロイ）。コンテナエンジン150によるデプロイによって、仮想I/Oネットワークサーバ540のインスタンスが生成される。

30

【0143】

仮想ノードのコンテナエンジン250-1は、デプロイしたPLCランタイムコンテナ50のコンテナ名をリポジトリ278に登録する（（15）コンテナ名登録）。また、仮想ノードのコンテナエンジン250-1は、デプロイして生成されたPLCランタイムコンテナ50のインスタンスの内容をコンテナハブ276に反映する（（16）デプロイ内容反映）。

【0144】

（g3：ユースケース3）

図17は、図12（C）に示すユースケース3における処理手順の一例を概略する模式図である。図17には、PLCランタイム510をローカル実行モード、クラウド実行モードおよびシミュレーションモードでそれぞれリリースした場合を示す。

40

【0145】

図17を参照して、ユーザは、設計するPLCシステム2の構成の定義を入力する（（1）構成定義）。すると、システム構成マネージャ264は、入力されたPLCシステム2に対応するシステム構成プロファイル60を生成する（（2）システム構成プロファイル生成）。

【0146】

続いて、ユーザは、開発環境262を利用してユーザプログラム266を作成する（（3）ユーザプログラム作成）。そして、システム構成マネージャ264は、PLCランタ

50

イムコンテナ50を生成する((4) PLCランタイムコンテナ生成)。さらに、コンテナマネージャ272は、生成されたPLCランタイムコンテナ50をコンテナハブ276に登録する((5) PLCランタイムコンテナ登録)。以上の処理によって、PLCランタイムコンテナ50の生成および登録が完了する。

【0147】

ユーザは、PLC100-1についてローカル実行モードを選択し、PLC100-2についてクラウド実行モードを選択し、PLC100-3についてシミュレーションモードを選択したとする((6) リリース指示)。

【0148】

ノードマネージャ274は、リリース先となる仮想ノードを生成する((7) 仮想ノード生成)とともに、生成した仮想ノードのノードアドレスをノード名と対応付けてリポジトリ278に登録する((8) ノード名登録)。

【0149】

ローカル実行モードで選択されたPLCランタイムコンテナ50は、リリース先のPLC100-1へ送信され((9) PLCランタイムコンテナリリース)、PLC100-1のコンテナエンジン150がPLCランタイムコンテナ50をデプロイする((10) PLCランタイムコンテナデプロイ)。

【0150】

クラウド実行モードで選択されたPLCランタイムコンテナ50は、リリース先のPLC100-2へ送信され((11) PLCランタイムコンテナリリース)、PLC100-2のコンテナエンジン150がPLCランタイムコンテナ50をデプロイする((12) PLCランタイムコンテナデプロイ)。また、クラウド実行モードで選択されたPLCランタイムコンテナ50は、リリース先の仮想ノードへも送信され((13) PLCランタイムコンテナリリース)、仮想ノードのコンテナエンジン250-2がPLCランタイムコンテナ50をデプロイする((14) PLCランタイムコンテナデプロイ)。

【0151】

シミュレーションモードで選択されたPLCランタイムコンテナ50は、リリース先の仮想ノードへも送信され((15) PLCランタイムコンテナリリース)、仮想ノードのコンテナエンジン250-3がPLCランタイムコンテナ50をデプロイする((16) PLCランタイムコンテナデプロイ)。

【0152】

仮想ノードのコンテナエンジン250-2および250-3は、デプロイしたPLCランタイムコンテナ50のコンテナ名をリポジトリ278に登録する((17) コンテナ名登録)。また、仮想ノードのコンテナエンジン250-2および250-3は、デプロイして生成されたPLCランタイムコンテナ50のインスタンスの内容をコンテナハブ276に反映する((18) デプロイ内容反映)。

【0153】

<H. 処理手順>

次に、本実施の形態に従う制御システム1における処理手順の一例について説明する。

【0154】

図18は、本実施の形態に従う制御システム1におけるPLCランタイムコンテナ50の生成に関する処理手順の一例を示すフローチャートである。図19は、本実施の形態に従う制御システム1におけるPLCランタイムコンテナ50のリリースおよび実行管理に関する処理手順の一例を示すフローチャートである。図18および図19に示す各ステップは、典型的には、サーバ200のプロセッサ202がプログラムを実行することで実現される。

【0155】

図18を参照して、ユーザは、開発環境262にアクセスして、PLCシステム2の構成を定義する。すなわち、サーバ200の開発環境262は、ユーザからPLCシステム2の構成の定義を受け付ける(ステップS100)。より具体的には、ユーザは、対象と

10

20

30

40

50

する P L C 1 0 0 および各 P L C 1 0 0 に接続されるデバイス（モータやロボットなど）などを指定する。

【 0 1 5 6 】

ユーザの定義に応答して、サーバ 2 0 0 の開発環境 2 6 2 は、P L C システム構成情報 2 6 8 を生成および出力する（ステップ S 1 0 2）。続いて、サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、指定された P L C 1 0 0 にそれぞれ対応するシステム構成プロファイル 6 0 を生成および出力する（ステップ S 1 0 4）。すなわち、システム構成プロファイル 6 0 は、指定された P L C 1 0 0 の数だけ生成される。また、システム構成プロファイル 6 0 の各々は、対応する P L C 1 0 0 に接続されるデバイスの情報を含む。

【 0 1 5 7 】

サーバ 2 0 0 の開発環境 2 6 2 は、I E C プログラムを生成するためのエディタ環境を提供し、ユーザにより作成されるユーザプログラム 2 6 6 を受け付ける（ステップ S 1 0 6）。

【 0 1 5 8 】

サーバ 2 0 0 の開発環境 2 6 2 は、システム構成プロファイル 6 0 を参照して、P L C 1 0 0 に接続されるいずれかのデバイスに関連付けられた機能モジュールが存在するか否かを判断する（ステップ S 1 0 8）。いずれかのデバイスに関連付けられた機能モジュールが存在すれば（ステップ S 1 0 8 において Y E S）、サーバ 2 0 0 の開発環境 2 6 2 は、当該関連付けられた機能モジュールに関連するエディタ環境を提供し、ユーザからの各種設定を受け付ける（ステップ S 1 1 0）。

【 0 1 5 9 】

ユーザによる設定操作が完了すると、サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、P L C ランタイムに必要な標準テンプレートをロードする（ステップ S 1 1 2）。サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、システム構成プロファイル 6 0 を参照して、P L C 1 0 0 に接続されるいずれかのデバイスに関連付けられたテンプレートが必要であるか否かを判断する（ステップ S 1 1 4）。いずれかのデバイスに関連付けられたテンプレートが必要であれば（ステップ S 1 1 4 において Y E S）、サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、必要なテンプレートをロードする（ステップ S 1 1 6）。

【 0 1 6 0 】

そして、サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、ユーザが生成したユーザプログラムと、ロードしてテンプレートとに基づいて、P L C ランタイムコンテナ 5 0 を生成する（ステップ S 1 1 8）。

【 0 1 6 1 】

続いて、サーバ 2 0 0 のコンテナマネージャ 2 7 2 は、コンテナ生成モジュール 2 6 0 により生成された P L C ランタイムコンテナ 5 0 をコンテナハブ 2 7 6 に登録する（ステップ S 1 2 0）。

【 0 1 6 2 】

以上の手順により、P L C ランタイムコンテナ 5 0 の生成に関する処理は完了する。すなわち、サーバ 2 0 0 は、制御対象に対する制御演算を定義するユーザプログラムとユーザプログラムを実行する実行環境とを内包する P L C ランタイムのイメージを含むコンテナを生成する。

【 0 1 6 3 】

図 1 9 を参照して、サーバ 2 0 0 の開発環境 2 6 2 は、ユーザ操作に応答して、コンテナハブ 2 7 6 およびリポジトリ 2 7 8 を参照して、予め用意されている P L C ランタイムコンテナ 5 0 の一覧を表示する（ステップ S 2 0 0）。

【 0 1 6 4 】

ユーザがいずれかの P L C ランタイムコンテナ 5 0 およびモードを選択すると、サーバ 2 0 0 のシステム構成マネージャ 2 6 4 は、いずれのモードが選択されたかを判断する（ステップ S 2 0 2）。

【 0 1 6 5 】

10

20

30

40

50

ローカル実行モードが選択されると（ステップS 2 0 2において「ローカル実行」）、サーバ2 0 0のノードマネージャ2 7 4は、システム構成プロファイル6 0を参照して、選択されたP L Cランタイムコンテナ5 0のリリース先となるP L C 1 0 0が有効に存在しているか否かを判断する（ステップS 2 0 4）。

【0 1 6 6】

リリース先となるP L C 1 0 0が有効に存在していれば（ステップS 2 0 4においてY E S）、サーバ2 0 0のコンテナマネージャ2 7 2は、P L Cランタイムコンテナ5 0のリリースに応じて、リポジトリ2 7 8の内容を更新する（ステップS 2 0 6）。続いて、サーバ2 0 0のシステム構成マネージャ2 6 4は、選択されたP L Cランタイムコンテナ5 0をリリース先のP L C 1 0 0へ送信する（ステップS 2 0 8）。送信先のP L C 1 0 0のコンテナエンジン1 5 0は、P L Cランタイムコンテナ5 0に含まれるマニフェスト5 6を参照して、P L Cランタイムコンテナ5 0をデプロイする（ステップS 2 1 0）。これにより、リリース先のP L C 1 0 0では、P L Cランタイム5 1 0が実現される。そして、処理は終了する。

【0 1 6 7】

一方、リリース先となるP L C 1 0 0が有効に存在していれば（ステップS 2 0 4においてN O）、サーバ2 0 0の開発環境2 6 2は、ユーザに対してリリース先のP L C 1 0 0が有効に存在していない旨を通知する（ステップS 2 1 2）。そして、処理は終了する。

【0 1 6 8】

クラウド実行モードが選択されると（ステップS 2 0 2において「クラウド実行モード」）、サーバ2 0 0のノードマネージャ2 7 4は、システム構成プロファイル6 0を参照して、選択されたP L Cランタイムコンテナ5 0のリリース先となるP L C 1 0 0が有効に存在しているか否かを判断する（ステップS 2 1 4）。

【0 1 6 9】

リリース先となるP L C 1 0 0が有効に存在していれば（ステップS 2 1 4においてY E S）、サーバ2 0 0のノードマネージャ2 7 4は、システム構成プロファイル6 0を参照して、選択されたP L Cランタイムコンテナ5 0のリリース先となる仮想ノードがサーバ2 0 0上に有効に存在しているか否かを判断する（ステップS 2 1 6）。

【0 1 7 0】

リリース先となる仮想ノードがサーバ2 0 0上に有効に存在していなければ（ステップS 2 1 6においてN O）、サーバ2 0 0のノードマネージャ2 7 4は、必要な数の仮想ノードを生成する（ステップS 2 1 8）。仮想ノードは、コンテナエンジン2 5 0を含む、P L Cランタイムコンテナ5 0をデプロイ可能な環境を提供するものである。

【0 1 7 1】

リリース先となる仮想ノードがサーバ2 0 0上に有効に存在していれば（ステップS 2 1 6においてY E S）、ステップS 2 1 8の処理はスキップされる。

【0 1 7 2】

サーバ2 0 0のコンテナマネージャ2 7 2は、P L Cランタイムコンテナ5 0のリリースに応じて、リポジトリ2 7 8の内容を更新する（ステップS 2 2 0）。そして、サーバ2 0 0のシステム構成マネージャ2 6 4は、選択されたP L Cランタイムコンテナ5 0をリリース先のP L C 1 0 0およびサーバ2 0 0上の仮想ノードへ送信する（ステップS 2 2 2）。一方、送信先のサーバ2 0 0の仮想ノードに含まれるコンテナエンジン2 5 0も、P L Cランタイムコンテナ5 0に含まれるマニフェスト5 6を参照して、P L Cランタイムコンテナ5 0をデプロイする（ステップS 2 2 4）。これにより、リリース先のP L C 1 0 0では、仮想I / Oネットワークサーバ5 4 0が実現され、サーバ2 0 0上の仮想ノードでは、P L Cランタイム5 1 0が実現される。

【0 1 7 3】

一方、リリース先となるP L C 1 0 0が有効に存在していなければ（ステップS 2 1 4においてN O）、サーバ2 0 0の開発環境2 6 2は、ユーザに対してリリース先のP L C 1 0 0が有効に存在していない旨を通知する（ステップS 2 2 6）。そして、処理は終了

10

20

30

40

50

する。

【0174】

シミュレーションモードが選択されると（ステップS202において「シミュレーションモード」）、サーバ200のノードマネージャ274は、システム構成プロファイル60を参照して、選択されたPLCランタイムコンテナ50のリリース先となる仮想ノードがサーバ200上に有効に存在しているか否かを判断する（ステップS228）。

【0175】

リリース先となる仮想ノードがサーバ200上に有効に存在していなければ（ステップS228においてNO）、サーバ200のノードマネージャ274は、必要な数の仮想ノードを生成する（ステップS230）。 10

【0176】

リリース先となる仮想ノードがサーバ200上に有効に存在していれば（ステップS228においてYES）、ステップS230の処理はスキップされる。

【0177】

サーバ200のコンテナマネージャ272は、PLCランタイムコンテナ50のリリースに応じて、リポジトリ278の内容を更新する（ステップS232）。そして、サーバ200のシステム構成マネージャ264は、選択されたPLCランタイムコンテナ50をサーバ200上の仮想ノードへ送信する（ステップS234）。送信先のサーバ200の仮想ノードに含まれるコンテナエンジン250は、PLCランタイムコンテナ50に含まれるマニフェスト56を参照して、PLCランタイムコンテナ50をデプロイする（ステップS236）。これにより、リリース先のサーバ200上の仮想ノードでは、PLCランタイム510および仮想I/Oエミュレータ550を含む、シミュレーションできる環境が実現される。このように、PLC100およびサーバ200の一方または両方に対してPLCランタイムコンテナ50をリリースする処理が実行される。 20

【0178】

＜I. その他の実施の形態＞

説明の便宜上、サーバ200がコンテナエンジン250と、コンテナ生成モジュール260と、オーケストレータ270（図7参照）とを有しているとしたが、サーバ200の実体がクラウド化および仮想化されている場合には、同一のハードウェアがこれらのモジュールのすべてを実現する必要はなく、1または複数のプロセッサを適宜用いて実現することができる。 30

【0179】

＜J. 付記＞

上述したような本実施の形態は、以下のような技術思想を含む。

【0180】

〔構成1〕

1または複数の制御装置（100）と1または複数のサーバ（200）とを備える制御システム（1）であって、

制御対象に対する制御演算を定義するユーザプログラム（266）と当該ユーザプログラムを実行する実行環境とを内包するランタイム（510）のイメージ（51）を含むコンテナ（50）を生成する生成モジュール（260）と、 40

前記制御装置および前記サーバの一方または両方に対して前記コンテナをリリースするオーケストレータ（270）とを備え、

前記制御装置および前記サーバは、前記コンテナをデプロイするコンテナエンジン（150, 250）を有しており、前記オーケストレータによりリリースされたコンテナをデプロイすることで、前記ランタイム（510）のインスタンスを生成して前記制御演算を実行する、制御システム。

【0181】

〔構成2〕

前記サーバのコンテナエンジン（250）は、前記ランタイムのインスタンスとともに 50

、前記制御対象とやり取りされる入出力データをエミュレートするエミュレータ（５５０）のインスタンスも生成する、構成１に記載の制御システム。

【０１８２】

〔構成３〕

前記サーバのコンテナエンジン（２５０）が前記ランタイムのインスタンスを生成する場合に、前記制御装置のコンテナエンジンが前記制御装置に接続されたデバイス（１０）を介してやり取りされる入出力データを前記サーバに仲介するネットワークサーバ（５４０）のインスタンスを生成する、構成１に記載の制御システム。

【０１８３】

〔構成４〕

前記サーバのコンテナエンジン（２５０）は、前記ランタイムのインスタンスを生成する場合に、前記制御装置に接続されたデバイスへアクセスする処理を担当するＩ／Ｏドライバをエミュレートする仮想Ｉ／Ｏドライバ（５２０）のインスタンスを生成する、構成１～３のいずれか１項に記載の制御システム。

【０１８４】

〔構成５〕

前記コンテナは、前記制御装置のコンテナエンジンのみがデプロイするイメージ（５４）と、前記サーバのコンテナエンジンのみがデプロイするイメージ（５２，５３，５５）とのうち、少なくとも一方を含む、構成１～４のいずれか１項に記載の制御システム。

【０１８５】

〔構成６〕

前記サーバのコンテナエンジンが前記ランタイムのインスタンスを生成する場合に、当該コンテナエンジンにアクセスするためのアドレスを管理するリポジトリ（２７８）をさらに備える、構成１～５のいずれか１項に記載の制御システム。

【０１８６】

〔構成７〕

前記サーバに対して前記コンテナをリリースする場合に、当該コンテナをデプロイするコンテナエンジンが存在しなければ、コンテナエンジンを含むノードを仮想的に生成するノードマネージャ（２７４）をさらに備える、構成１～６のいずれか１項に記載の制御システム。

【０１８７】

〔構成８〕

前記生成モジュールにより生成されるコンテナをリリース可能な制御装置の識別情報と関連付けて登録するコンテナハブ（２７６）をさらに備える、構成１～７のいずれか１項に記載の制御システム。

【０１８８】

〔構成９〕

前記オーケストレータは、ユーザ操作に従って、リリースするコンテナを選択するとともに、リリース先を決定する、構成１～８のいずれか１項に記載の制御システム。

【０１８９】

〔構成１０〕

１または複数の制御装置（１００）と１または複数のサーバ（２００）とを備える制御システム（１）において実行される制御方法であって、

制御対象に対する制御演算を定義するユーザプログラムと当該ユーザプログラムを実行する実行環境とを内包するランタイムのイメージを含むコンテナを生成するステップ（Ｓ１００～Ｓ１２０）と、

前記制御装置および前記サーバの一方または両方に対して前記コンテナをリリースするステップ（Ｓ２００～Ｓ２３４）とを備え、

前記制御装置および前記サーバは、前記コンテナをデプロイするコンテナエンジンを有しており、前記リリースされたコンテナをデプロイすることで、前記ランタイムのインス

10

20

30

40

50

タンスを生成して前記制御演算を実行する、制御方法。

【0190】

＜K．利点＞

本実施の形態によれば、コンテナ技術およびオーケストレーション技術を利用可能な制御システムを実現できる。これらの技術を利用することで、例えば、制御演算のシミュレーションなどを行う場合には、サーバ（クラウド）のリソースをスケーラブルに利用できるので、より複雑および大規模なシミュレーションであっても、実時間で行うことができる。

【0191】

本実施の形態によれば、P L Cに接続されたデバイスを利用しつつ、制御演算自体はサーバで実行することもできるので、フィールド側に配置されるP L Cの小型化および簡素化などを行うこともできる。

【0192】

本実施の形態によれば、複数のP L Cから構成される場合であっても、P L Cの各々について、ローカル実行モード、クラウド実行モード、シミュレーションモードの3つのモードで制御演算を実行できるので、複数のP L Cを組み合わせで検証する必要がある場合でも、そのときの状況で可能なモードでの組み合わせで検証を行うことができる。このような組み合わせでの検証によって、システム結合時の問題を早期に発見および解決などが可能となり、制御システムの運用開始までの時間を短縮できる。

【0193】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は、上記した説明ではなく、特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【符号の説明】

【0194】

1 制御システム、2 P L Cシステム、4 制御系ネットワーク、6 情報系ネットワーク、8 インターネット、10 フィールドデバイス、12 リモートI/O、14 リレー群、16 サーボドライバ、18 サーボモータ、20 ゲートウェイ、50 ランタイムコンテナ、51 P L Cランタイムイメージ、52 仮想I/Oドライバイメージ、53 仮想I/Oネットワーククライアントイメージ、54 仮想I/Oネットワークサーバイメージ、55 仮想I/Oエミュレータイメージ、56 マニフェスト、60 システム構成プロファイル、100 P L C、102, 202, 302 プロセッサ、104 チップセット、106, 206, 306 主メモリ、110, 210, 310 ストレージ、112, 212, 312 OS、114 システムプログラム、120 情報系ネットワークコントローラ、122 制御系ネットワークコントローラ、124, 324 USBコントローラ、126 メモリカードインターフェイス、128 メモリカード、140 ローカルOS、142 物理I/Oドライバ、150, 250 コンテナエンジン、160 物理I/O、200 サーバ、208, 308 バス、220, 320 ネットワークコントローラ、226, 326 入力部、228, 328 表示部、240 サーバOS、260 コンテナ生成モジュール、262 開発環境、264 システム構成マネージャ、266 ユーザプログラム、268 システム構成情報、270 オーケストレータ、272 コンテナマネージャ、274 ノードマネージャ、276 コンテナハブ、278 リポジトリ、300 開発保守端末、304 光学ドライブ、305 記録媒体、314 開発プログラム、510 P L Cランタイム、520 仮想I/Oドライバ、530 仮想I/Oネットワーククライアント、540 仮想I/Oネットワークサーバ、550 仮想I/Oエミュレータ。

10

20

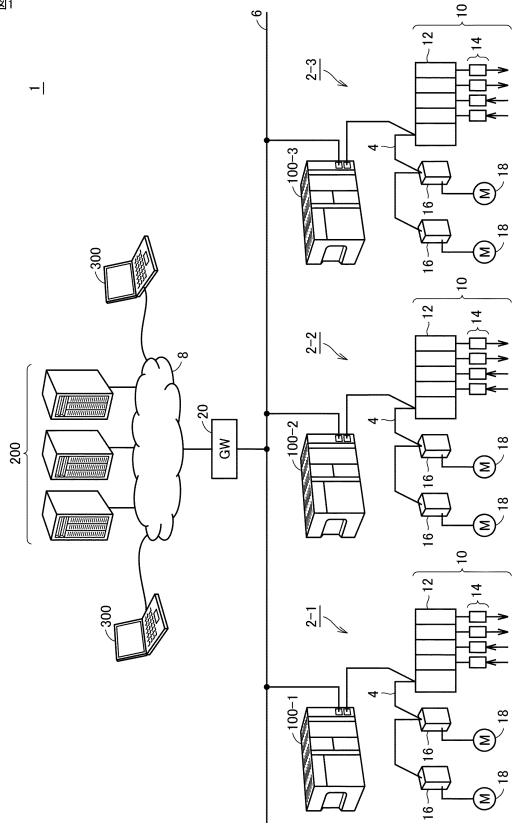
30

40

50

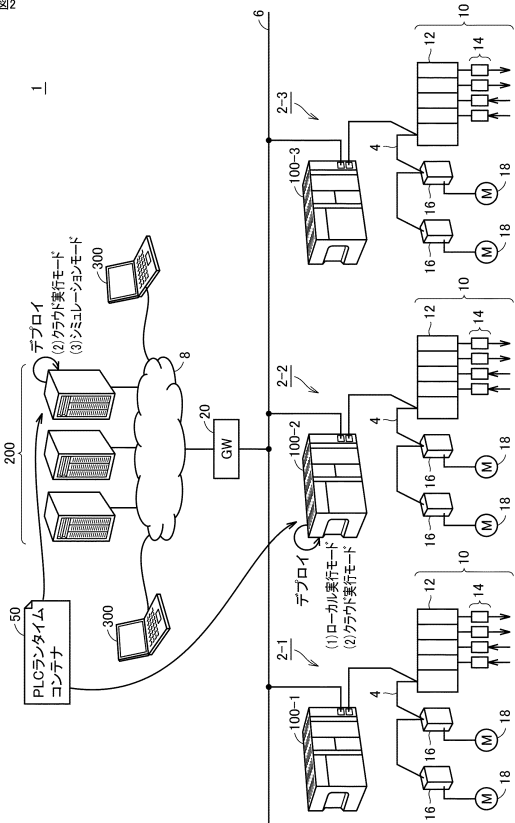
【図面】
【図 1】

図1



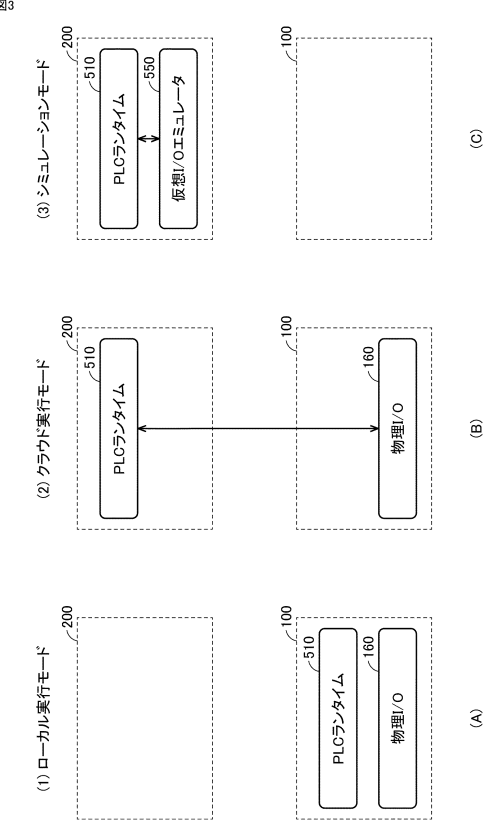
【図 2】

図2



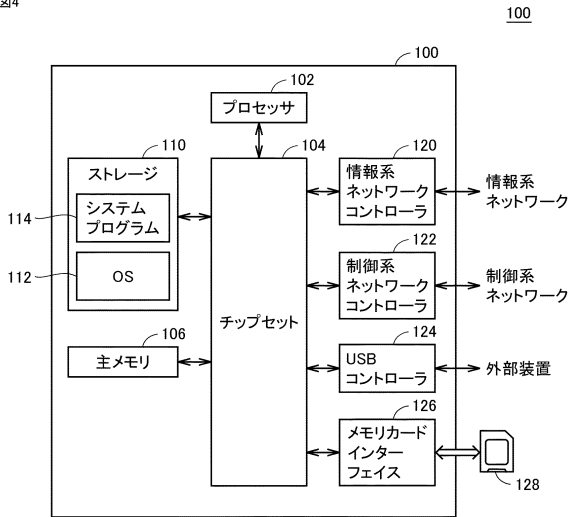
【図 3】

図3



【図 4】

図4



10

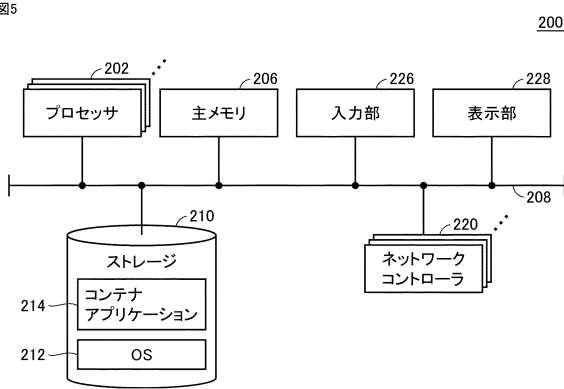
20

30

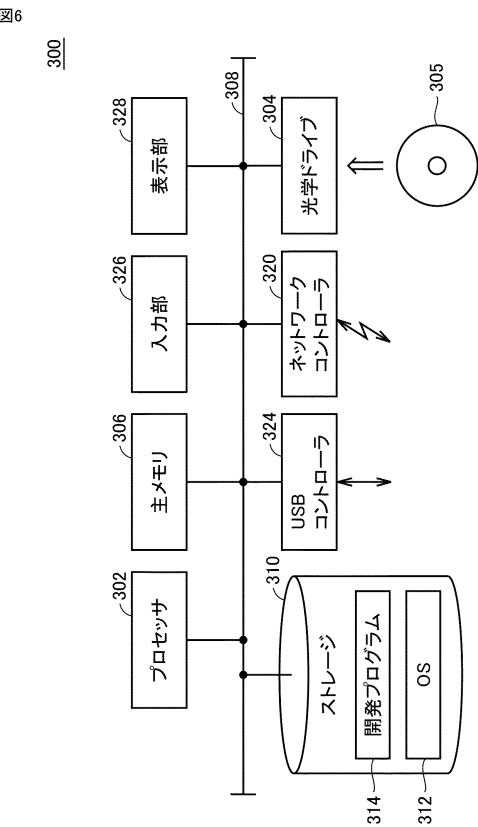
40

50

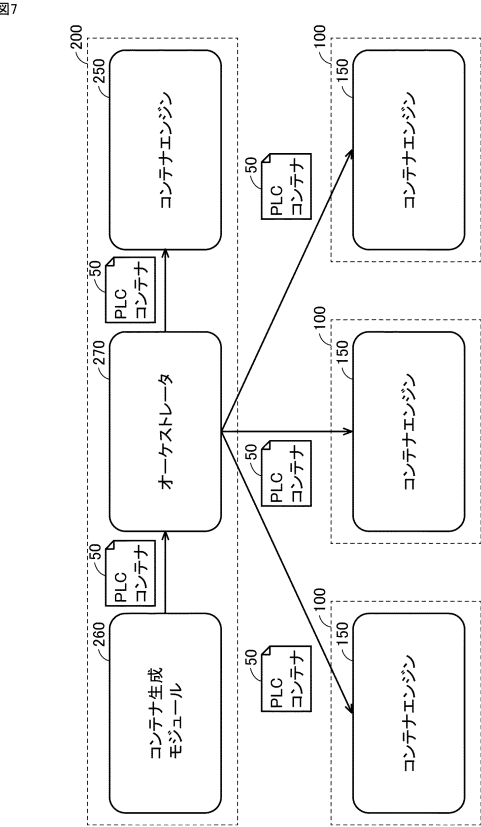
【図 5】



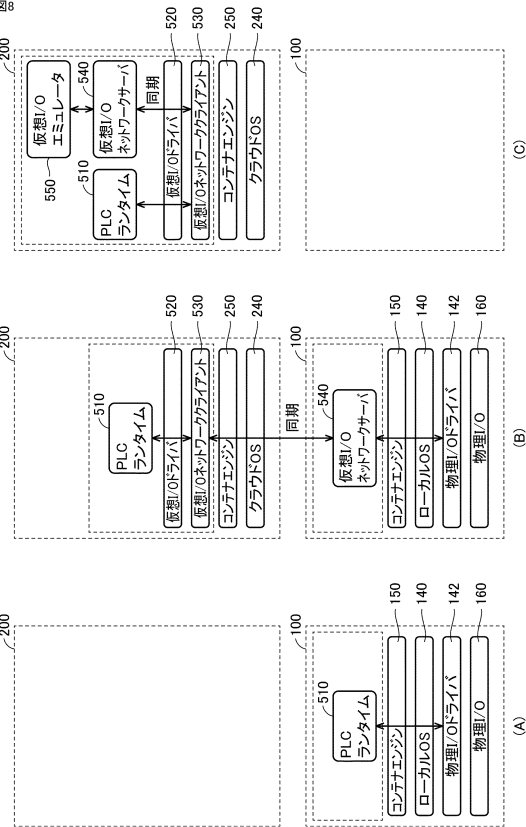
【図 6】



【図 7】



【図 8】



10

20

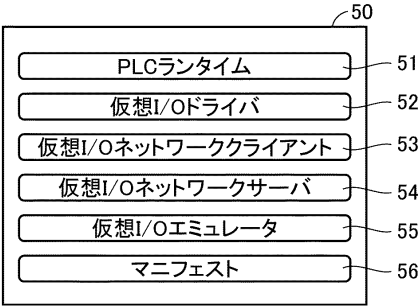
30

40

50

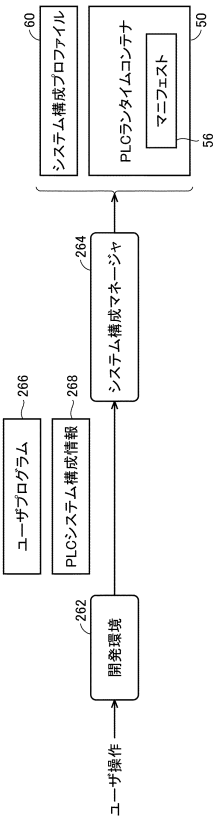
【図 9】

図9



【図 10】

図10



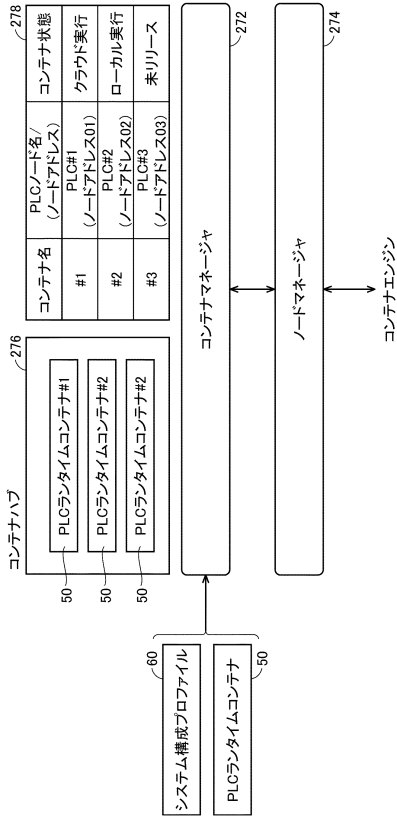
10

20

【図 11】

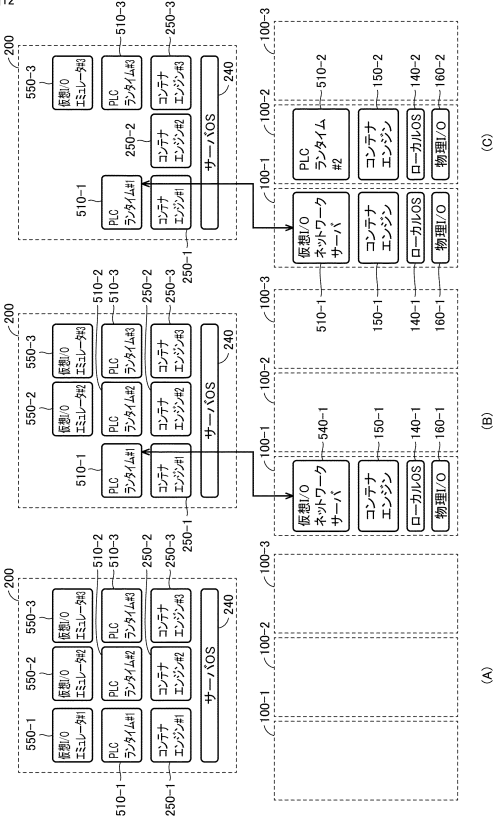
図11

270



【図 12】

図12

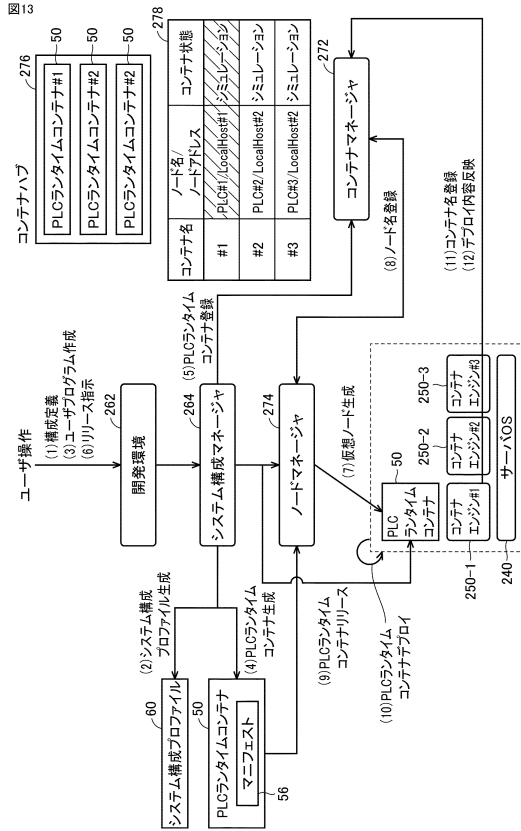


30

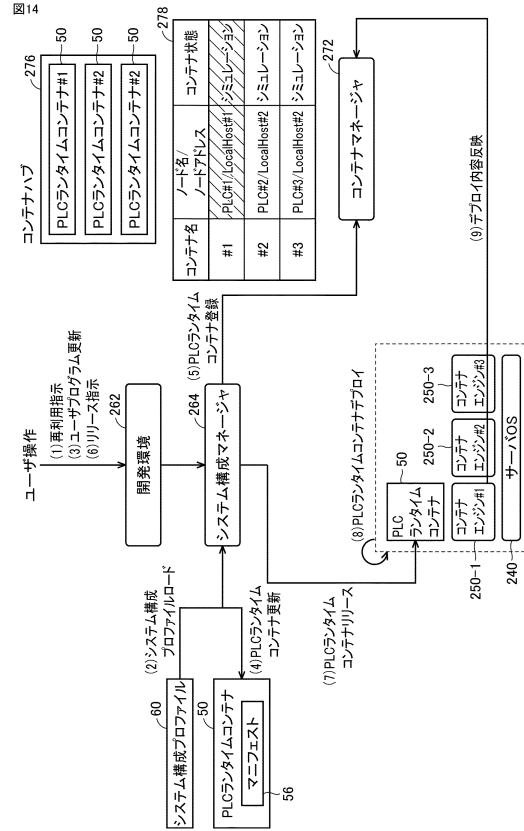
40

50

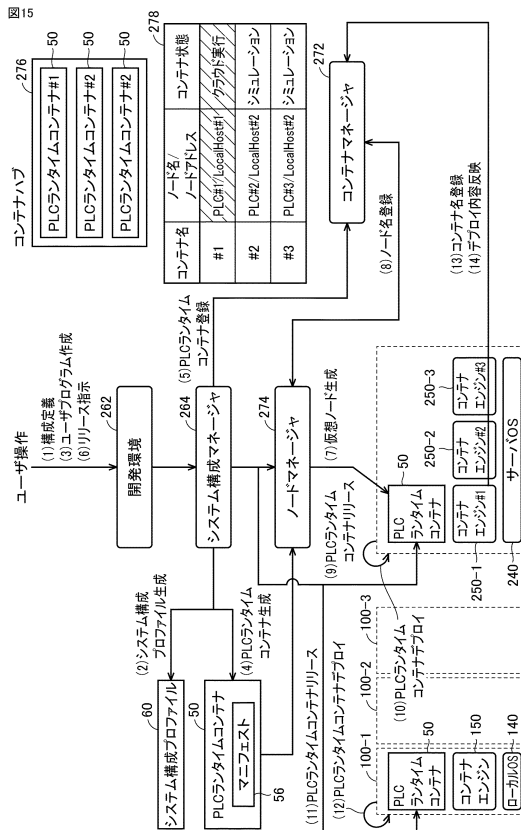
【図 13】



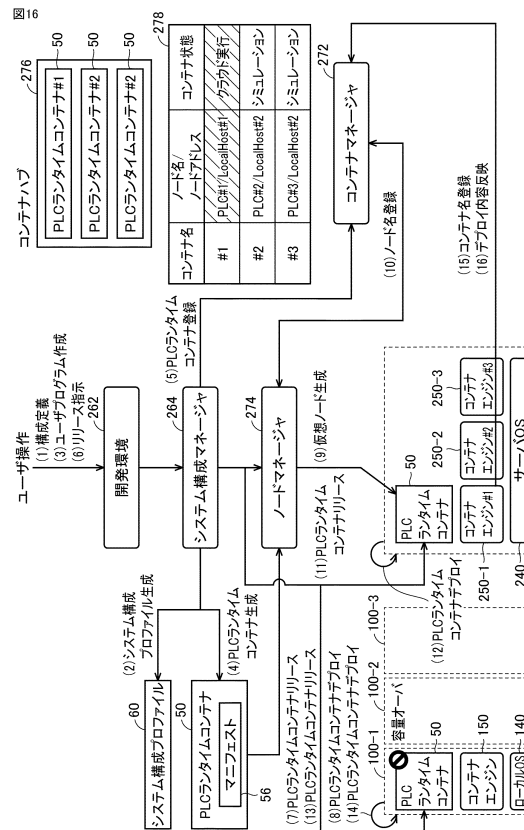
【図 14】



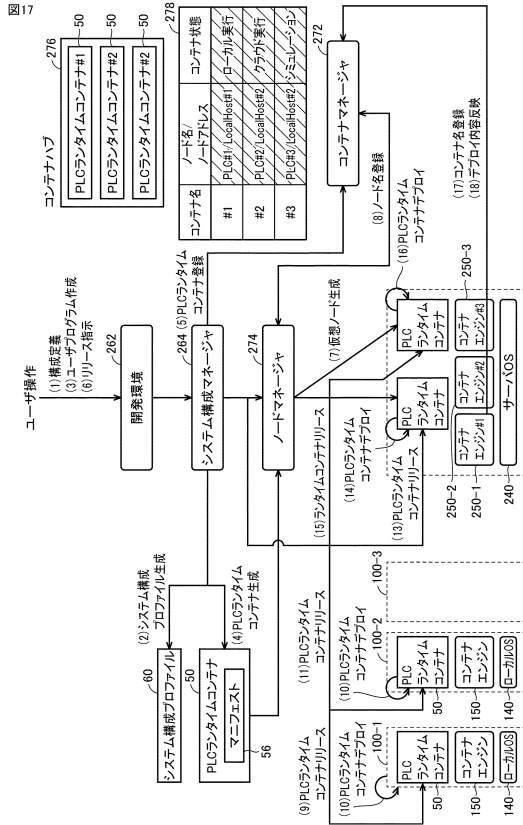
【図 15】



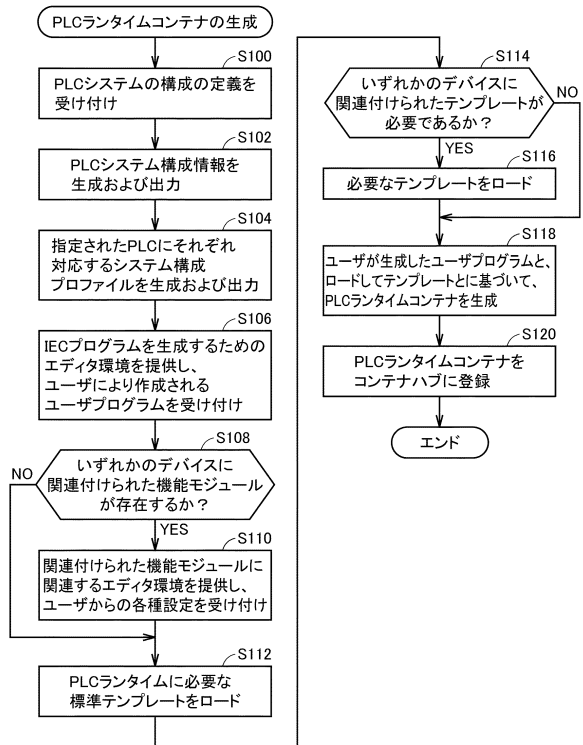
【図 16】



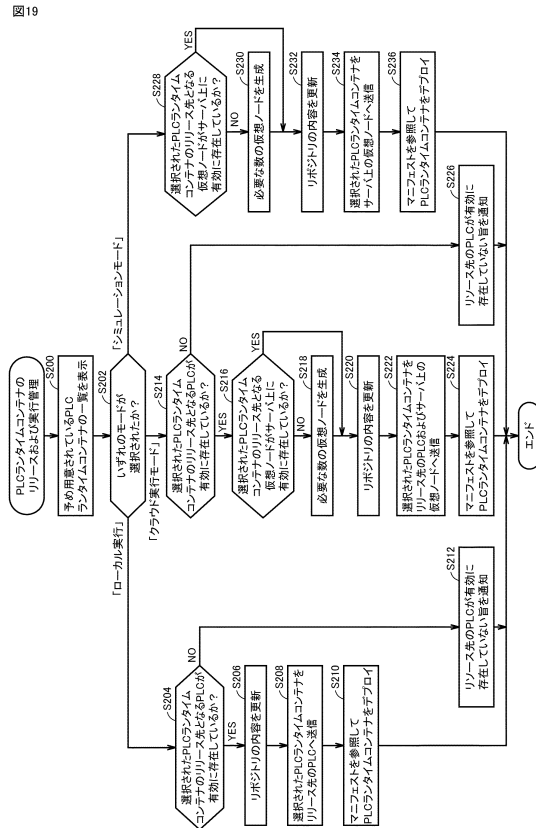
【図 17】



【図 18】



【図 19】



フロントページの続き

- (56)参考文献 特開 2019-128863 (JP, A)
国際公開第 2019/099111 (WO, A1)
欧州特許出願公開第 3722951 (EP, A1)
米国特許出願公開第 2014/0222408 (US, A1)
- (58)調査した分野 (Int.Cl., DB名)
G05B 19/05
G06F 8/00-9/54
G06F 11/36