

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7670629号
(P7670629)

(45)発行日 令和7年4月30日(2025.4.30)

(24)登録日 令和7年4月21日(2025.4.21)

(51)国際特許分類

F I

G O 6 F 8/60 (2018.01)

G O 6 F 8/60

G O 6 F 21/44 (2013.01)

G O 6 F 21/44

請求項の数 15 (全130頁)

(21)出願番号	特願2021-573500(P2021-573500)	(73)特許権者	520203080
(86)(22)出願日	令和2年6月10日(2020.6.10)		ネッターサンダー, エル・エル・シー
(65)公表番号	特表2022-536706(P2022-536706 A)		Net-Thunder, LLC
(43)公表日	令和4年8月18日(2022.8.18)		アメリカ合衆国 60090 イリノイ
(86)国際出願番号	PCT/US2020/037087		ウィーリング イースト マークワート
(87)国際公開番号	WO2020/252088		ドライブ 111
(87)国際公開日	令和2年12月17日(2020.12.17)		111 E Marquardt Drive, Wheeling, IL 60
審査請求日	令和5年6月9日(2023.6.9)		090, United States
(31)優先権主張番号	62/860,148		of America
(32)優先日	令和1年6月11日(2019.6.11)	(74)代理人	100114890
(33)優先権主張国・地域又は機関	米国(US)		弁理士 アインゼル・フェリックス＝ラ
			インハルト
		(74)代理人	100098501
			弁理士 森田 拓

最終頁に続く

(54)【発明の名称】 セキュリティが強化された自動的に配備される情報技術（IT）システム及び方法

(57)【特許請求の範囲】

【請求項1】

情報技術（IT）コンピュータシステムであって、
複数のシステムルール、前記システムのシステム状態、及び複数のテンプレートに基づいて、前記システムを自動的に管理するように構成されたコントローラと、

前記コントローラに接続するためのリソースであって、前記リソースは、第1のサービス及び第2のサービスを含み、前記第1及び第2のサービスは、互いに依存関係を有し、前記第1のサービスは、前記第2のサービスに対する被依存サービスを含み、前記第2のサービスは、前記第1のサービスに対する依存サービスを含む、前記リソースと、
前記第1及び第2のサービスを互いに、且つ、前記コントローラとインタフェースさせるアプリケーションプログラムインタフェース（API）と、
を含み、

前記テンプレートは、前記第2のサービスのセットアップに必要な依存関係リストを有する前記第2のサービスのテンプレートを含み、列挙された前記依存関係は、前記第1のサービスを含み、

前記コントローラによる自動管理は、前記第2のサービスのテンプレートからの前記依存関係リストに基づいて、前記第2のサービスを構成すること及び前記APIを通して前記第2のサービスに対する前記第1のサービスの相互運用性を管理することを含む、
情報技術（IT）コンピュータシステム。

【請求項2】

前記第 2 のサービスが、前記第 1 のサービスの前記 A P I を通して前記第 1 のサービスに動作を実行するように呼び出しを発するように構成される、請求項 1 に記載のシステム。

【請求項 3】

前記第 1 のサービスが、前記第 2 のサービスのために動作を実行するために、前記第 2 のサービスの妥当性を確認するように前記コントローラに求めるように構成される、請求項 1 ～ 2 のいずれかに記載のシステム。

【請求項 4】

前記コントローラが、相互認証、公開鍵認証、及び／またはネットワークベースの妥当性確認のうちの少なくとも 1 つに基づいて、前記第 2 のサービスの妥当性を確認するようにさらに構成される、請求項 3 に記載のシステム。

10

【請求項 5】

前記第 1 のサービスが、前記コントローラからの許可を条件に、前記第 2 のサービスのために動作を実行するように構成される、請求項 1 ～ 4 のいずれかに記載のシステム。

【請求項 6】

前記コントローラが、前記第 1 及び第 2 のサービスの妥当性確認に使用するために、前記第 1 及び第 2 のサービスに暗号鍵をプロビジョニングするようにさらに構成される、請求項 1 ～ 5 のいずれかに記載のシステム。

【請求項 7】

前記暗号鍵が、前記第 1 及び第 2 のサービスのそれぞれに対して異なる鍵の対を含み、各鍵の対は、公開鍵及びプライベート鍵を含む、請求項 6 に記載のシステム。

20

【請求項 8】

前記コントローラが、前記プライベート鍵を前記第 1 及び第 2 のサービスにプロビジョニングした後、前記プライベート鍵のコピーを削除するようにさらに構成され、前記コントローラが、前記公開鍵に基づいて、前記第 1 及び第 2 のサービスの妥当性確認を管理するように構成される、請求項 7 に記載のシステム。

【請求項 9】

前記第 1 及び／または第 2 のサービスが前記コントローラによって構成中の場合、前記コントローラは、あらゆる外部のネットワークから前記リソースを切断するようにさらに構成される、請求項 1 ～ 8 のいずれかに記載のシステム。

【請求項 10】

前記第 1 及び／または第 2 のサービスが前記コントローラによって構成された後、前記コントローラは、あらゆる切断された外部ネットワークに前記リソースを再接続するようにさらに構成される、請求項 9 に記載のシステム。

30

【請求項 11】

インバンド接続、アウトオブバンド接続、及び／またはストレージエリアネットワーク接続のうちの少なくとも 1 つを、前記リソースと前記コントローラの間にさらに含み、前記コントローラは、前記第 1 及び／または第 2 のサービスが前記コントローラによって構成中の場合、前記インバンド接続、アウトオブバンド接続、及び／またはストレージエリアネットワーク接続から前記リソースを切断するようにさらに構成される、請求項 9 ～ 10 のいずれかに記載のシステム。

40

【請求項 12】

前記第 1 及び／または第 2 のサービスが前記コントローラによって構成された後、前記コントローラは、前記切断されたインバンド接続、アウトオブバンド接続、及び／またはストレージエリアネットワーク接続に前記リソースを再接続するようにさらに構成される、請求項 11 に記載のシステム。

【請求項 13】

前記コントローラが、前記第 1 のサービスと前記第 2 のサービスの間の依存関係を解決するようにさらに構成される、請求項 1 ～ 12 のいずれかに記載のシステム。

【請求項 14】

前記コントローラが、互いに依存関係を有するサービスを含む、前記システムの 1 つま

50

たは複数のコンポーネントのバックアップ動作を管理するようにさらに構成される、請求項 1 ～ 13 のいずれかに記載のシステム。

【請求項 15】

コントローラと、前記コントローラに接続されたリソースであって、前記リソースは、第 1 のサービス及び第 2 のサービスを含み、前記第 1 及び第 2 のサービスは、互いに依存関係を有し、前記第 1 のサービスは、前記第 2 のサービスに対する被依存サービスを含み、前記第 2 のサービスは、前記第 1 のサービスに対する依存サービスを含む、前記リソースとを含むコンピュータシステムで使用される情報技術（IT）方法であって、複数のシステムルール、前記システムのシステム状態、及び複数のテンプレートに基づいて、前記システムを自動的に管理することであって、前記テンプレートは、前記第 2 のサービスのセットアップに必要な依存関係リストを有する前記第 2 のサービスのテンプレートを含み、列挙された前記依存関係は、前記第 1 のサービスを含む、ことと、

10

前記第 1 及び第 2 のサービスを互いに、且つ、前記コントローラとアプリケーションプログラムインタフェース（API）を介してインタフェースさせることと、
を含み、

前記自動的に管理することは、前記コントローラが、前記第 2 のサービスのテンプレートからの前記依存関係リストに基づいて、前記第 2 のサービスを構成すること及び前記 API を通して前記第 2 のサービスに対する前記第 1 のサービスの相互運用性を管理することを含む、

20

方法。

【発明の詳細な説明】

【技術分野】

【0001】

関連出願の相互参照

本特許出願は、2019年6月11日に出願され、「Automatically Deployed Information Technology (IT) System and Method with Enhanced Security」と題された、米国仮特許出願連番 62/860,148 号の優先権を主張し、その開示全体が引用により本明細書に組み込まれる。

【背景技術】

30

【0002】

計算の需要、利用、及びニーズは過去数十年で飛躍的に高まっている。それと共に、より大きなストレージ、速度、計算能力、アプリケーション、アクセシビリティへの需要によって、コンピューティングの分野は急速に変化することとなり、多様なタイプ及びサイズのエンティティにツールが提供されている。その結果、パブリックな仮想計算システム及びクラウドコンピューティングシステムが開発されて、より多くの計算リソースが多数のユーザ及びユーザタイプに提供されるようになった。この指数関数的な成長は続くと思われる。同時に、障害及びセキュリティのリスクが高まると、インフラストラクチャのセットアップ、管理、変更管理、更新はより複雑になり、コストがかかるようになる。スケーラビリティ、すなわち、時間の経過と共にシステムを成長させることは、情報技術の分野でも大きな課題となっている。

40

【0003】

ほとんどの IT システムの問題は、その多くはパフォーマンス及びセキュリティに関連するものであるが、診断及び対処するのが困難な場合がある。システムのセットアップ、構成、及び配備に許される時間及びリソースの制約は、エラーにつながり、将来の IT の問題をもたらし得る。時間の経過と共に、ユーザ、アプリケーション、サービス、セキュリティ、ソフトウェア、及びハードウェアを含む IT システムの変更、パッチ適用、または更新に、多くの異なる管理者が関与する場合がある。しばしば、構成及び変更の文書化及び履歴が不十分であるかまたは失われて、特定のシステムがどのように構成されて機能するかを後で理解するのが困難になる場合がある。これにより、将来の変更またはトラブル

50

ルシューティングが困難になる場合がある。問題または障害が生じた場合、ITの構成及び設定は、復旧及び再現するのが困難な場合がある。また、システム管理者は、例えば誤ったコマンドまたはその他のミスなど、簡単にミスを犯す可能性があり、それによってコンピュータならびにWebデータベース及びサービスがダウンする場合がある。さらに、セキュリティ侵害のリスクの増加は珍しくないが、セキュリティ侵害を回避するための変更、更新、パッチは、望まれないダウンタイムを引き起こす場合がある。

【0004】

重要なインフラストラクチャが定位置に配置され、機能し、稼働すると、コストまたはリスクはしばしば、システムを変更するメリットを上回るように思われる場合がある。稼働中のITシステムまたは環境に変更を加えることに関わる問題は、これらのシステムに依存するユーザまたはエンティティに重大な、時には壊滅的な問題を引き起こす可能性がある。少なくとも、変更管理中に発生する障害または問題のトラブルシューティング及び解決にかかる時間は、かなりの時間、人員、及び金銭のリソースを必要とする場合がある。稼働中の環境に変更が加えられた場合に生じ得る技術的な問題は、連鎖的な影響を及ぼす可能性があり、加えられた変更を元に戻すだけでは解決されない場合がある。これらの問題点の多くは、変更管理中に障害が存在する場合にシステムを迅速に再構築できない一因となる。

【0005】

さらに、ITシステム内のベアメタルクラウドノードまたはリソースは、セキュリティの問題点に対して脆弱な場合があり、または不正なユーザによって危険にさらされ、もしくはアクセスされ得る。ハッカー、攻撃者、または不正なユーザは、そのノードまたはリソースからピボットして、ITシステムの他の部分またはノードに接続されているネットワークにアクセスまたはハッキングし得る。ITシステムのベアメタルクラウドノードまたはコントローラも、システムをセキュリティの脅威にさらすか、または別の方法でシステムを危険にさらし得るアプリケーションネットワークに接続されたリソースを介して脆弱になる場合がある。本明細書で開示する様々な例示的な実施形態によれば、ITシステムは、インターネットとインタフェースするか、または外部ネットワークへの接続の有無によらないアプリケーションネットワークからの、ベアメタルクラウドノードまたはリソースのセキュリティを改善するように構成され得る。

【0006】

例示的な実施形態によれば、ITシステムは、ベアメタルクラウドノードまたは物理リソースを含む。ベアメタルクラウドノードまたは物理リソースが電源投入、セットアップ、管理、または使用されるときに、他の人々または顧客が使用している場合があるノードを有するネットワークに接続され得る場合、インバンド管理はコントローラから省略されてよく、切り替え可能であってよく、切断可能であってよく、またはフィルタリングされてよい。また、システム内のアプリケーションネットワークまたは複数アプリケーションのネットワークは、アプリケーションネットワークがコントローラに結合されるリソース（複数可）を介して、コントローラから切断されてよく、切断可能であってよく、切り替え可能であってよく、またはフィルタリングされてよい。

【0007】

仮想マシンまたはハイパーバイザを含む物理リソースもまた、セキュリティの問題点に対して脆弱な場合があり、ハイパーバイザを使用して、共有リソースである他のハイパーバイザにピボットし得る場合に、不正なユーザによって危険にさらされ、またはアクセスされる場合がある。攻撃者は仮想マシンから抜け出してよく、コントローラを介して管理システム及び／または監督システムにネットワークアクセスできる場合がある。本明細書で開示する様々な例示的な実施形態によれば、ITシステムはセキュリティを改善するように構成されてよく、クラウドプラットフォーム上の仮想リソースを含む1つまたは複数の物理リソースは、インバンド管理接続を介して、コントローラから切断されてよく、切断可能であってよく、フィルタリングされてよく、フィルタリング可能であってよく、またはコントローラに非接続にされてよい。

10

20

30

40

50

【0008】

例示的な実施形態によれば、ITシステムの物理リソースは、1つまたは複数の仮想マシンまたはハイパーバイザを含んでよく、コントローラと物理リソースとの間のインバンド管理接続は、そのリソースから省略されてよく、切断されてよく、切断可能であってよく、またはフィルタリングされて／フィルタリング可能であってよい。

【0009】

例示的な実施形態によると、システムは、本明細書に記載の技術を用いて、システム内に相互に関連したサービスをプロビジョニング及び管理するコントローラを含み得る。例として、他のサービスと相互依存性を有するサービスの削除時に修正の解消の仕方を管理するクリーンアップルールを作成及び維持することができる。

10

【0010】

例示的な実施形態によると、システムは、本明細書に記載の技術を用いて、計算リソースにストレージをプロビジョニングする、及び／または、クラウドインスタンスにリソースをプロビジョニング及び接続するコントローラを含み得る。

【0011】

さらに、例示的な実施形態によると、システムは、本明細書に記載のアーキテクチャを使用して、複数の相互依存サービスを伴うバックアップを含む、効率的なバックアップ動作を支援することができる。

【図面の簡単な説明】

【0012】

20

【図1】例示的な実施形態によるシステムの概略図である。

【図2A】図1のシステム用の例示的なコントローラの概略図である。

【図2B】ストレージ拡張ルールの例示的なセットの動作の例示的なフローを示す。

【図2C】図2Bのステップ210.1及び210.2を実行するための代替例を示す。

【図2D】図2Bのステップ210.1及び210.2を実行するための代替例を示す。

【図2E】例示的なテンプレートを示す。

【図2F】テンプレートの処理に関するコントローラロジックの例示的なプロセスフローを示す。

【図2G】図2Fのステップ205.11、205.12、及び205.13の例示的なプロセスフローを示す。

30

【図2H】図2Fのステップ205.11、205.12、及び205.13の例示的なプロセスフローを示す。

【図2I】他の例示的なテンプレートを示す。

【図2J】テンプレートの処理に関するコントローラロジックの他の例示的なプロセスフローを示す。

【図2K】サービスの依存関係を管理するための例示的なプロセスフローを示す。

【図2L】例示的な実施形態によるテンプレートから導出された例示的なイメージの概略図である。

【図2M】システムルールの例示的なセットを示す。

【図2N】コントローラロジックが図2Mのシステムルールを処理するための例示的なプロセスフローを示す。

40

【図2O】ファイルシステムblobまたは他のファイルのグループからストレージリソースを構成するための例示的なプロセスフローを示す。

【図3A】計算リソースが追加された図2Aのコントローラの概略図である。

【図3B】例示的な実施形態によるテンプレートから導出された例示的なイメージの概略図である。

【図3C】計算リソース、ストレージリソース、及び／またはネットワーキングリソースなどのリソースをシステムに追加するための例示的なプロセスフローを示す。

【図4A】ストレージリソースが追加された図2Aのコントローラの概略図である。

【図4B】例示的な実施形態によるテンプレートから導出された例示的なイメージの概略

50

図である。

【図 5 A】 J B O D 及びストレージリソースが追加された図 2 A のコントローラの概略図である。

【図 5 B】 ストレージリソース及びストレージリソースの直接接続ストレージをシステムに追加するための例示的なプロセスフローを示す。

【図 6 A】 ネットワーキングリソースが追加された図 2 A のコントローラの概略図である。

【図 6 B】 例示的な実施形態によるテンプレートから導出された例示的なイメージの概略図である。

【図 7 A】 例示的な物理的配備における例示的な実施形態によるシステムの概略図である。

【図 7 B】 I T システムにリソースを追加するための例示的なプロセスを示す。

【図 7 C】 複数の計算リソース、複数のサーバ、複数の仮想マシン、及び／または複数のサイトにアプリケーションを配備するための例示的なプロセスフローを示す。

【図 7 D】 複数の計算リソース、複数のサーバ、複数の仮想マシン、及び／または複数のサイトにアプリケーションを配備するための例示的なプロセスフローを示す。

【図 8 A】 例示的な配備における例示的な実施形態によるシステムの概略図である。

【図 8 B】 単一ノードシステムから複数ノードシステムに拡張するための例示的なプロセスフローを示す。

【図 8 C】 ストレージリソースを新たな物理ストレージリソースに移行するための例示的なプロセスフローを示す。

【図 8 D】 マルチテナントシステムの単一ノード上の仮想マシン、コンテナ、及び／またはプロセスを、計算及びストレージ用の別個のハードウェアを有し得るマルチノードシステムに移行するための例示的なプロセスフローを示す。

【図 8 E】 システム内で単一ノードから複数ノードに拡張するための他の例示的なプロセスフローを示す。

【図 9 A】 例示的な物理的配備における例示的な実施形態によるシステムの概略図である。

【図 9 B】 例示的な実施形態によるテンプレートから導出された例示的なイメージの概略図である。

【図 9 C】 N T パッケージからアプリケーションをインストールする一例を示す。

【図 9 D】 例示的な配備における例示的な実施形態によるシステムの概略図である。

【図 9 E】 I T システムに仮想計算リソースホストを追加するための例示的なプロセスフローを示す。

【図 9 F】 クラウド上にインスタンス 3 1 0 a を含むリソースへの追加の接続を有する例示的なシステムを示す。

【図 9 G - 1】 図 9 F のシステムに関する例示的なプロセスフローを示す。

【図 9 G - 2】 図 9 F のシステムに関する例示的なプロセスフローを示す。

【図 9 G - 3】 図 9 F のシステムに関する例示的なプロセスフローを示す。

【図 9 G - 4】 図 9 F のシステムに関する例示的なプロセスフローを示す。

【図 9 H】 クラウド上に追加のインスタンスを有し、追加のインスタンスは、クラウド A P I に接続され、インバンド管理接続によって V P N を通してコントローラにも接続される、図 9 F に示す例示的なシステムを示す。

【図 9 I】 図 9 H のシステムに関する例示的なプロセスフローを示す。

【図 9 J】 この図も、クラウド上に追加のインスタンスを有し、インスタンスは、クラウド A P I に接続され、インバンド管理接続によって V P N を通してコントローラにも接続される、図 9 F に示す他の例示的なシステムを示す。

【図 9 K】 図 9 J のシステムに関する例示的なプロセスフローを示す。

【図 9 L】 システムをクラウドからローカルホスト上に分離する例示的なプロセスフローを示す。

【図 1 0】 例示的な配備における例示的な実施形態によるシステムの概略図である。

【図 1 1 A】 例示的な実施形態のシステム及び方法を示す。

【図 1 1 B】 例示的な実施形態のシステム及び方法を示す。

10

20

30

40

50

【図 1 2】例示的な実施形態のシステム及び方法を示す。

【図 1 3 A】例示的な実施形態によるシステムの概略図である。

【図 1 3 B】例示的な実施形態によるシステムの他の概略図である。

【図 1 3 C】例示的な実施形態によるシステムの例示的なプロセスフローを示す。

【図 1 3 D】例示的な実施形態によるシステムの例示的なプロセスフローを示す。

【図 1 3 E】例示的な実施形態によるシステムの例示的なプロセスフローを示す。

【図 1 4】A は、メインコントローラがコントローラを異なるシステムに配備した例示的なシステムを示す。B は、メインコントローラを有するコントローラをプロビジョニングするための可能なステップを示す例示的なフローを示す。C は、メインコントローラを有するコントローラをプロビジョニングするための可能なステップを示す例示的なフローを示す。

10

【図 1 5 A】メインコントローラが環境を生み出す例示的なシステムを示す。

【図 1 5 B】コントローラが環境をセットアップする例示的なプロセスフローを示す。

【図 1 5 C】コントローラが複数の環境をセットアップする例示的なプロセスフローを示す。

【図 1 6 A】コントローラが 1 つまたは複数のコントローラをセットアップするためのメインコントローラとして動作する例示的な実施形態を示す。

【図 1 6 B】環境が他の環境に書き込むように構成され得る例示的なシステムを示す。

【図 1 6 C】環境が他の環境に書き込むように構成され得る例示的なシステムを示す。

【図 1 6 D】環境が他の環境に書き込むように構成され得る例示的なシステムを示す。

20

【図 1 6 E】コントローラによって生み出される新たな環境をユーザが購入できる例示的なシステムを示す。

【図 1 6 F】コントローラによって生み出される環境にインタフェースするためのユーザインタフェースが設けられた例示的なシステムを示す。

【図 1 7 A】新たな環境に関する変更管理タスクの例を示す。

【図 1 7 B】新たな環境に関する変更管理タスクの例を示す。

【図 1 8 A】新たな環境に関する変更管理タスクの例を示す。

【図 1 8 B】新たな環境に関する変更管理タスクの例を示す。

【図 1 9 A】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

30

【図 1 9 B】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

【図 1 9 C】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

【図 1 9 D】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

【図 1 9 E】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

【図 1 9 F】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

40

【図 1 9 G】システム内に相互に関連したサービスをプロビジョニング及び管理することに関するシステム及びシステムのプロセスフローの例を示す。

【図 2 0】A は、1 つまたは複数の計算リソースが、1 つまたは複数のストレージリソースのストレージを利用する 1 つまたは複数のサービスをホストする例示的なシステム及び関連するプロセスフローを示す。B は、1 つまたは複数の計算リソースが、1 つまたは複数のストレージリソースのストレージを利用する 1 つまたは複数のサービスをホストする例示的なシステム及び関連するプロセスフローを示す。C は、1 つまたは複数の計算リソースが、1 つまたは複数のストレージリソースのストレージを利用する 1 つまたは複数のサービスをホストする例示的なシステム及び関連するプロセスフローを示す。D は、1 つまたは複数の計算リソースが、1 つまたは複数のストレージリソースのストレージを利用

50

する1つまたは複数のサービスをホストする例示的なシステム及び関連するプロセスフローを示す。

【図21A】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21B】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21C-1】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21C-2】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

10

【図21C-3】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21C-4】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21D】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21E】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21F】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

20

【図21G】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21H】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21I】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21J】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21K】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

30

【図21L-1】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図21L-2】システム、サービス、またはシステム内の他のコンポーネントのバックアップに関するシステム及び関連するプロセスフローの例を示す。

【図22A】システム、サービス、またはシステム内の他のコンポーネントの更新に関するシステム及び関連するプロセスフローの例を示す。

【図22B】システム、サービス、またはシステム内の他のコンポーネントの更新に関するシステム及び関連するプロセスフローの例を示す。

【図22C】システム、サービス、またはシステム内の他のコンポーネントの更新に関するシステム及び関連するプロセスフローの例を示す。

40

【発明を実施するための形態】

【0013】

上述のような当技術分野のニーズに対する技術的解決策を提供する取り組みにおいて、本発明者らは、自動化されたITシステムのセットアップ、構成、保守、テスト、変更管理、及び／またはアップグレードを提供する情報技術のためのシステム及び方法に関する多様な発明の実施形態を開示する。例えば、本発明者らは、複数のシステムルール、コンピュータシステムのシステム状態、及び複数のテンプレートに基づいてコンピュータシステムを自動的に管理するように構成されるコントローラを開示する。他の例として、本発明者らは、複数のシステムルール、コンピュータシステムのシステム状態、及び複数のテンプレートに基づいて、コンピュータシステムの物理インフラストラクチャを自動的に管

50

理するように構成されたコントローラを開示する。コントローラによって実行可能な自動管理の例には、アプリケーションまたはサービスを動作させ得るコンピュータ上の設定またはその他の情報へのリモートまたはローカルでのアクセス及び変更、ITシステムの構築、ITシステムの変更、ITシステム内の個々のスタックの構築、サービスまたはアプリケーションの作成、サービスまたはアプリケーションのロード、サービスまたはアプリケーションの構成、サービスまたはアプリケーションの移行、サービスまたはアプリケーションの変更、サービスまたはアプリケーションの削除、異なるネットワーク上の他のスタックへのスタックのクローン、リソースまたはシステムコンポーネントの作成、追加、削除、セットアップ、構成、再構成、及び／または変更、リソース、サービス、アプリケーション、ITシステム、及び／またはITスタックの自動的な追加、削除、及び／または復帰、アプリケーション、サービス、スタック、及び／またはその他のITシステム間のインタラクションの構成、及び／またはITシステムコンポーネントの健全性の監視、が含まれ得る。例示的な実施形態では、コントローラは、リモートまたはローカルとすることができる物理または仮想計算リソースとして具現化することができる。採用できるコントローラの追加の例には、プロセス、仮想マシン、コンテナ、リモート計算リソース、他のコントローラによって配備されるアプリケーション、及び／またはサービスのいずれかまたは任意の組み合わせが含まれるが、これらに限定されない。コントローラは、複数のノード及び／またはリソースにわたって分散されてもよく、他の場所またはネットワークにあってもよい。

【0014】

ITインフラストラクチャは、ほとんどの場合、個別のハードウェアコンポーネント及びソフトウェアコンポーネントから構築される。使用されるハードウェアコンポーネントは、一般的に、サーバ、ラック、電源機器、相互接続、ディスプレイモニタ、及びその他の通信機器を含む。これらの個別のコンポーネントを選択して相互接続する方法及び技法は、種々の度合いの効率、費用対効果、パフォーマンス、及びセキュリティで機能する極めて多数の任意選択の構成によって非常に複雑になる。これらのインフラストラクチャコンポーネントの接続に熟練した個々の技術者／エンジニアは、雇用及びトレーニングするのに費用がかかる。また、ハードウェア及びソフトウェアの極めて多数の繰り返しの可能性によって、そのハードウェア及びソフトウェアの保守及び更新が複雑になる。これにより、ITインフラストラクチャを元々インストールした個人及び／またはエンジニアリング会社が更新を実施できない場合に、追加の課題が生まれている。オペレーティングシステムなどのソフトウェアコンポーネントは、幅広いハードウェアで動作するように汎用的に設計されているか、または特定のコンポーネントに非常に特化している。ほとんどの場合、複雑な計画または青写真が描かれて実行される。変更、成長、スケーリング、及びその他の課題には、複雑な計画を更新することが必要になる。

【0015】

一部のITユーザは、成長しているサプライヤ産業からクラウドコンピューティングサービスを購入するが、これはインフラストラクチャのセットアップの問題及び課題を解決せず、むしろITユーザからクラウドサービスプロバイダに転嫁する。さらに、大規模なクラウドサービスプロバイダは、柔軟性、カスタマイズ性、スケーラビリティ、ならびに新たなハードウェア技術及びソフトウェア技術の迅速な採用を低下させ得る方法で、インフラストラクチャのセットアップの課題及び問題に対処している。また、クラウドコンピューティングサービスは、すぐに使えるベアメタルセットアップ、構成配備及び更新を提供せず、ベアメタル及び仮想ITインフラストラクチャコンポーネントへの移行、それらからの移行、またはそれらの間の移行も可能にしない。クラウドコンピューティングサービスのこれら及びその他の制限により、多くの計算、ストレージ、及びネットワーキングの非効率性が生じ得る。例えば、計算及びネットワーキングにおける速度または遅延の非効率性は、クラウドサービスによって、またはクラウドサービスを利用するアプリケーションもしくはサービスにおいて発生し得る。

【0016】

例示的な実施形態のシステム及び方法は、新規かつ独自のITインフラストラクチャの配備、利用及び管理を提供する。例示的な実施形態によれば、リソースの選択、インストール、相互接続、管理、及び更新の複雑性は、コアコントローラシステムと、そのパラメータファイル、テンプレート、ルール、及びITシステム状態とに根ざしている。このシステムは、技術者に組み立て、接続、及び管理を要求するのではなく、コンポーネントが自己組み立てを行うように構成された自己組み立てルール及び動作ルールのセットを含む。さらに、例示的な実施形態のシステム及び方法は、現行の典型的な外部計画書を必要とせずに、自己組み立てルールを使用して、より高いカスタマイズ性、スケーラビリティ、及び柔軟性を可能にする。それらはまた、効率的なリソースの使用及び再利用も可能にする。

10

【0017】

全体的または部分的に物理的または仮想的であるかを問わず、現在のITシステムの問題点及び問題の多くを改善するシステム及び方法が提供される。例示的な実施形態のシステム及び方法は、柔軟性を可能にし、ばらつき及び人的エラーを低減し、システムセキュリティを向上させ得る構造を提供する。

【0018】

現在のITシステムの問題のうちの1つまたは複数に対していくつかの解決策が個別に存在し得るが、そのような解決策は、本明細書に記載の例示的な実施形態によって解決されるような多数の問題に包括的に対処するものではない。さらに、そのような既存の解決策は、特定の問題に対処し得るが、他の問題を悪化させ得る。

20

【0019】

対処される現在の課題の中には、セットアップ、構成、インフラストラクチャの配備、資産の追跡、セキュリティ、アプリケーションの配備、サービスの配備、保守及びコンプライアンスの文書化、保守、スケーリング、リソースの割り当て、リソース管理、負荷分散、ソフトウェア障害、ソフトウェア及びセキュリティの更新／パッチ適用、テスト、ITシステムの復旧、変更管理、及びハードウェアの更新、に関する問題点があるが、これらに限定されない。

【0020】

本明細書で使用するITシステムは、サーバ、仮想ホスト及び物理ホスト、データベース及びデータベースアプリケーション、例えば、限定はしないが、ITサービス、ビジネスコンピューティングサービス、コンピュータアプリケーション、顧客対応アプリケーション、Webアプリケーション、モバイルアプリケーション、バックエンド、ケース番号管理、顧客追跡、発券、ビジネスツール、デスクトップ管理ツール、会計、電子メール、文書化、コンプライアンス、データストレージ、バックアップ、及び／またはネットワーク管理を含み得るが、これらに限定されない。

30

【0021】

ユーザがITシステムをセットアップする前に直面し得る問題の1つは、インフラストラクチャのニーズの予測である。最初に、または成長中もしくは変化中に時間の経過と共に、どれだけのストレージ、計算能力、またはその他の要求が必要になるかが、ユーザは分からない場合がある。例示的な実施形態によれば、ITシステム及びインフラストラクチャは、システムが変更を必要とする場合、例示的な実施形態の自己配備インフラストラクチャ（物理及び／または仮想の両方）を使用して、後でインフラストラクチャ内で自動的に追加、削除、または再割り当てし得るという点で柔軟性を可能にする。したがって、システムのセットアップ時に提示される将来のニーズを予測するという課題は、グローバルルール、テンプレート、及びシステム状態を使用してシステムに追加する機能を提供し、そのようなルール、テンプレート、及びシステム状態の変更を追跡することによって対処される。

40

【0022】

また、他の課題は、正しい構成、構成の統一性、相互運用性、及び／または相互依存性に関連するものであってよく、これには、例えば、時間の経過に伴う構成済みシステム要

50

素またはその構成の変更による将来の非互換性が含まれ得る。例えば、ITシステムが最初にセットアップされたときに、欠落している要素があるか、または一部の要素の構成に失敗している場合がある。また、例えば、要素またはインフラストラクチャコンポーネントの繰り返しセットアップされる場合、繰り返しの間で統一性が欠如している場合がある。システムに変更が加えられた場合に、構成の見直しが必要になる場合がある。最適な構成と、将来のインフラストラクチャの変更に対する柔軟性との間で、難しい選択が提示されている。例示的な実施形態によれば、最初にシステムを配備するときに、グローバルシステムルールを使用してテンプレートからインフラストラクチャコンポーネントへ構成が自己配備されるので、構成は統一され、再現可能または予測通りになって、最適な構成が可能になる。そのような初期システム配備は物理コンポーネント上に行われ得るが、後続のコンポーネントが追加または変更される場合があり、これらは物理的なものであってもなくてもよい。さらに、そのような初期システム配備は物理コンポーネント上に行われ得るが、後続の環境が物理構造からクローンされる場合があり、これらは物理的なものであってもなくてもよい。これにより、将来問題を起こす変更を最小限に抑えながら、システム構成を最適にすることが可能になる。

【0023】

配備フェーズでは、典型的には、ベアメタル及び／またはソフトウェア定義のインフラストラクチャの相互運用性の課題がある。また、ソフトウェアと、他のアプリケーション、ツール、またはインフラストラクチャとの相互運用性の課題もあり得る。これらには、異なるベンダー製の配備された製品による課題が含まれ得るが、これらに限定されない。本発明者らは、ベアメタル、仮想、またはそれらの任意の組み合わせに関係なく、インフラストラクチャの相互運用性を提供し得るITシステムを開示する。したがって、相互運用性、すなわち、パーツが連携する能力は、インフラストラクチャが自動的に構成及び配備される、開示したインフラストラクチャ配備に組み込まれ得る。例えば、異なるアプリケーションが相互に依存してよく、それらは別々のホストに存在し得る。そのようなアプリケーションが相互に対話できるようにするために、本明細書で説明するコントローラロジック、テンプレート、システム状態、及びシステムルールは、アプリケーションの相互依存関係の構成と、相互依存関係の追跡とに使用される情報及び構成命令を含む。したがって、本明細書で説明するインフラストラクチャの特徴は、各アプリケーションまたはサービスがどのようにして相互にやり取りするかを管理する方法を提供する。例として、電子メールサービスが認証サービスと適切に通信するようにし、及び／またはグループウェアサービスが電子メールサービスと適切に通信するようにする。またさらに、そのような管理は、インフラストラクチャレベルまで下りて、例えば、計算リソースがストレージリソースとどのように通信しているかを追跡可能にすることができる。そうでなければ、ITシステムの複雑性が $O(n^n)$ で上昇する可能性がある。

【0024】

開示しているように、リソースの自動配備は、グローバルシステムルール、テンプレート、及びITシステム状態／システムの自己認識に基づいて配備するコントローラ的能力により、オペレーティングシステムソフトウェアの事前構成を必要としない。例示的な実施形態によれば、ユーザまたはIT専門家は、相互運用性を確保するために、リソースの追加、割り当て、または再割り当てが連携するか否かを知る必要がない場合がある。例示的な実施形態による追加のリソースが、ネットワークに自動的に追加され得る。

【0025】

アプリケーションを使用するには、典型的には計算、ストレージ、及びネットワーキングを含む多くの異なるリソースが必要になる。また、リソースとシステムコンポーネントとの相互運用性も必要になり、これには、定位置に配置されて動作しているものに関する知識、及び他のアプリケーションとの相互運用性が含まれる。アプリケーションは、他のサービスに接続して構成ファイルを取得し、全てのコンポーネントが適切に連携することを確認する必要がある。そのため、アプリケーションの構成は、時間及びリソース集約的となる可能性がある。他のアプリケーションとの相互運用性の問題がある場合、アプ

리케이션の構成は、インフラストラクチャの残りの部分に連鎖的な影響をもたらす可能性がある。これは、停止または侵害につながる可能性がある。本発明者らは、これらの問題点に対処するために自動化されたアプリケーション配備を開示する。したがって、本発明者らが開示するように、システムの現在の状況に関する知識を使用して、ITシステム状態、グローバルシステムルール、及びテンプレートから読み取り、インテリジェントに構成することによって、アプリケーションが自己配備され得る。さらに、例示的な実施形態によれば、本明細書で説明する変更管理機能を使用して、構成の配備前テストが実行され得る。

【0026】

例示的な実施形態によって対処される他の問題点は、異なるベンダーまたは他のツールに切り替えることが望ましい場合に中間構成に関連して生じ得る問題に関するものである。例示的な実施形態の一態様によれば、コントローラのルール及びテンプレートと、特定のベンダーのアプリケーションテンプレートとの間のテンプレート変換が提供される。これにより、システムはソフトウェアまたは他のツールのベンダーを自動的に変更することが可能になる。

【0027】

多くのセキュリティの問題点は、構成ミス、パッチ適用の失敗、及び配備前にパッチ適用をテストできないことから生じる。しばしば、セットアップの構成段階でセキュリティの問題点が発生し得る。例えば、構成ミスによって、機密性の高いアプリケーションがインターネットに公開されたままになったり、電子メールサーバから電子メールの偽造を許したりする場合がある。本発明者らは、自動的に構成されることによって、攻撃者から保護して、攻撃者への不要な公開を回避し、セキュリティエンジニア及びアプリケーションセキュリティアーキテクトにシステムに関するより多くの知識を提供するシステムセットアップを開示する。自動化により、人的ミスまたは構成ミスによるセキュリティの欠陥が減少する。また、開示したインフラストラクチャは、サービス間の内省を提供し、ルールベースのアクセスを可能にし、サービス間の通信を、実際に必要なもののみに制限し得る。本発明者らは、例えば変更管理に関して説明するように、配備前にパッチを安全にテストする能力を有するシステム及び方法を開示する。

【0028】

文書化はしばしば、問題の多いIT管理の領域である。セットアップ及び構成中、主な目標は、典型的にはコンポーネントを連携させることであり得る。通常、これにはトラブルシューティング及び試行錯誤のプロセスが含まれ、実際にシステムを機能させたものを知ることが困難な場合がある。実行された正確なコマンドは通常は文書化されるが、機能するシステムを実現した可能性があるトラブルシューティングまたは試行錯誤のプロセスは、十分に文書化されていないか、まったく文書化されていないことが多い。文書化における問題または不備により、監査証跡及び監査に関する問題が発生し得る。発生した文書化の問題によって、コンプライアンスを示す際に問題が生じ得る。システムまたはそのコンポーネントを構築する場合に、しばしば、コンプライアンスの問題点はよく知られていない場合がある。適用可能なコンプライアンスの判断は、ITシステムをセットアップ及び構成した後にはわからない場合がある。このように、文書化は監査及びコンプライアンスに不可欠である。本発明者らは、自動的に文書化されるセットアップ及び構成を提供する、グローバルシステムルールデータベース、テンプレート、及びITシステム状態データベースを含むシステムを開示する。発生するあらゆる構成はデータベースに記録される。例示的な実施形態によれば、自動的に文書化される構成は監査証跡を提供し、コンプライアンスを示すために使用することができる。インベントリ管理では、自動的に文書化及び追跡される情報を使用し得る。

【0029】

ITシステムのセットアップ、構成、及び運用から生じるもう1つの課題は、ハードウェア及びソフトウェアのインベントリ管理に関するものである。例えば、通常、サーバがいくつあるか、それらが立ち上がっていてまだ機能しているか否か、それらの機能が何で

10

20

30

40

50

あるか、各サーバがどのラックにあるか、どの電源がどのサーバに接続されているか、どのネットワークカード及びどのネットワークポートを各サーバが使用しているか、どのＩＴシステムでコンポーネントが動作しているか、及びその他の多くの重要事項を知ることが重要である。インベントリ情報に加えて、インベントリ管理に使用されるパスワード及びその他の機密情報は、効果的に管理される必要がある。特に大規模なＩＴシステム、データセンタ、または機器が頻繁に変更されるデータセンタでは、この情報の収集及び保持は時間のかかる作業であり、これは多くの場合、手動で、または多様なソフトウェアツールを使用して管理される。安全なパスワードの準拠した保護は、安全なコンピューティング環境を保証する際に重要な問題点になる可能性がある大きなリスク要因である。本発明者らは、全てのサーバ及びその他のコンポーネントのインベントリ及び運転状態の収集及び保守が、ＩＴシステム状態、グローバルシステムルール、テンプレート、及びコントローラのコントローラロジックの一部として、自動的に更新、記憶、及び保護されるＩＴシステムを開示する。

10

【００３０】

ＩＴシステムのセットアップ及び構成に関する問題に加えて、本発明者らは、ＩＴシステムの保守で生じる問題及び問題点にも対処し得るＩＴシステムを開示する。例えば、とりわけ、電源障害、メモリ障害、ネットワーク障害、ネットワークカード障害、及び／またはＣＰＵ障害などの、ハードウェア障害のあるデータセンタが継続的に機能することに伴って、多くの問題が生じる。ハードウェア障害中にホストを移行する場合、さらなる障害が発生する。したがって、本発明者らは動的なリソースの移行、例えば、ホストがダウンした場合に、あるリソースプロバイダから他のリソースプロバイダにリソースを移行することを開示する。そのような状況では、例示的な実施形態によれば、ＩＴシステムは、他のサーバ、ノード、もしくはリソース、または他のＩＴシステムに移行することができる。コントローラは、システムの状態を報告し得る。データの複製は、既知の自動的にセットアップされた構成を有する他のホスト上にある。ハードウェア障害が検出された場合、ハードウェアが提供していた可能性がある任意のリソースは、障害を自動的に検出した後に、自動的に移行され得る。

20

【００３１】

多くのＩＴシステムに関する重要な問題点はスケーラビリティである。成長中の企業または他の組織は、典型的には、それらが成長してニーズが変化すると、ＩＴシステムを追加または再構成する。例えば、ハードドライブ容量、ストレージ容量、ＣＰＵ処理、より多くのネットワークインフラストラクチャ、より多くのエンドポイント、より多くのクライアント、及び／またはより多くのセキュリティの追加など、より多くのリソースが既存のＩＴシステムに必要な場合に、問題が生じる。異なるサービス及びアプリケーション、またはインフラストラクチャの変更が必要になった場合、構成、セットアップ、及び配備でも問題が生じる。例示的な実施形態によれば、データセンタは自動的にスケーリングされ得る。ノードまたはリソースは、動的かつ自動的にリソースのプールに対して追加または削除され得る。リソースプールに対して追加及び削除されたリソースは、自動的に割り当てまたは再割り当てされ得る。サービスがプロビジョニングされ、新たなホストにすばやく移動され得る。コントローラは、より多くのリソースを検出して動的にリソースプールに追加し、リソースを割り当て／再割り当てすべき場所を把握し得る。例示的な実施形態によるシステムは、単一ノードのＩＴシステムから、複数のデータセンタまたはＩＴシステムにわたる多数の物理及び／または仮想ノードまたはリソースを必要とするスケーリングされたシステムにスケーリングし得る。

30

40

【００３２】

本発明者らは、柔軟なリソース割り当て及び管理を可能にするシステムを開示する。このシステムは、リソースプール内にあってよく、動的に割り当てられ得る計算リソース、ストレージリソース、及びネットワーキングリソースを含む。コントローラは、ネットワーク上の新たなノードまたはホストを認識し、次にそれらがリソースプールの一部になれるようにそれらを構成し得る。例えば、新たなサーバがプラグインされると、コントロー

50

ラはそれをリソースプールの一部として構成し、リソースに追加することができ、動的に使用を開始することができる。ノードまたはリソースは、コントローラによって検出され、異なるプールに追加され得る。リソース要求は、例えば、コントローラへのAPI要求を介して行われ得る。次に、コントローラは、ルールに従ってプールから必要なリソースを配備または割り当て得る。これにより、コントローラ及び／またはコントローラを介したアプリケーションは、要求のニーズに基づいてリソースを負荷分散し、動的に分散させることが可能になる。

【0033】

負荷分散の例には、ハードウェアまたはソフトウェアの障害が発生した場合に新たなリソースを配備すること、ユーザ負荷の増加にตอบสนองして、同じアプリケーションの1つまたは複数のインスタンスを配備すること、及びストレージ、計算、またはネットワーキングの要件の不均衡にตอบสนองして、同じアプリケーションの1つまたは複数のインスタンスを配備すること、が含まれるが、これらに限定されない。

【0034】

稼働中のITシステムまたは環境に変更を行うことに関わる問題は、これらのシステムが一貫して立ち上がり、動作することに依存するユーザまたはエンティティにとって重大な、時には壊滅的な問題を発生させる場合がある。これらの停止は、システムの使用における潜在的な損失を表すだけでなく、データの損失、問題を解決するために必要な時間、人員、及び金銭の相当なリソースによる経済的損失を表す。構成の文書に誤りがあるか、システムの理解が不足している場合にシステムの再構築が困難なことによって、問題が深刻化する可能性がある。この問題のため、多くのITシステムユーザは、既知のセキュリティリスクを排除するためにITリソースにパッチを適用することに消極的である。このようにして、セキュリティ侵害に対してより脆弱なままになる。

【0035】

ITシステムの保守で生じる数多くの問題は、構成が必要になり得る変更管理または制御によるソフトウェア障害に関連するものである。そのような障害が発生し得る状況には、新たなソフトウェアバージョンへのアップグレード、異なるソフトウェアへの移行、パスワードまたは認証管理の変更、サービス間またはあるサービスの異なるプロバイダ間での切り替え、が含まれるが、これらに限定されない。

【0036】

手動で構成及び保守されるインフラストラクチャは、通常、再作成が困難である。インフラストラクチャの再作成は、問題のある変更のロールバック、停電、またはその他の災害復旧を含むがこれらに限定されないいくつかの理由で重要であり得る。手動で構成されるシステムの問題を診断することは困難である。手動で構成及び保守されるインフラストラクチャを再作成することは困難である。また、システム管理者は、例えば、コンピュータシステムをダウンさせたことがわかっている誤ったコマンドなど、簡単に間違いを犯す可能性がある。

【0037】

稼働中のITシステムまたは環境に変更を加えると、これらのシステムが一貫して起動し、動作することに依存するユーザまたはエンティティにとって重大な、時には壊滅的な問題が発生する可能性がある。これらの停止は、システムの使用における潜在的な損失を表すだけでなく、そのような停止はまた、データの損失に加え、問題の解決に必要な時間、人員、及び金銭の相当なリソースによる経済的損失を引き起こす可能性がある。構成の文書に誤りがあるか、またはシステムの理解が不足している場合にシステムの再構築が困難なことによって、問題が深刻化する可能性がある。また、多くの場合、大幅な変更または大きな変更の後にシステムを以前の状態に復旧することは非常に困難である。

【0038】

さらに、稼働中の環境に変更が加えられた場合に発生し得る技術的な問題は、連鎖的な影響を及ぼす場合がある。これらの連鎖的な影響により、変更前の状態に戻ることが困難になり、場合によっては不可能になり得る。したがって、実施された変更に関する問題の

10

20

30

40

50

ために変更を戻す必要がある場合でも、システムの状態は既に変更されている。近年、インフラストラクチャ及びシステム管理のエラー、ならびに本番環境への不完全な変更を元に戻すことは解決されない問題であると述べられている。さらに、稼働中の環境への配備前にシステムへの変更をテストすることは問題があることが知られている。

【0039】

したがって、本発明者らは、稼働中のシステムへの変更を変更前の状態に戻すように構成されたシステム及び方法のいくつかの例示的な実施形態を開示する。さらに、本発明者らは、上述の問題の1つまたは複数を防止または改善し得る、稼働中の変更を受けたシステムまたは環境の状態の大幅な復帰を可能にするように構成されたシステム及び方法を開示する。

【0040】

例示的な実施形態の変形によれば、ITシステムは、グローバルシステムルール、テンプレート、及びITシステム状態によってシステムの完全な知識を有する。システムの完全な知識を使用してインフラストラクチャがクローンされ得る。システムまたはシステム環境は、ソフトウェア定義のインフラストラクチャまたは環境としてクローンされ得る。本番環境と呼ばれる、使用中の揮発性データベースを含むシステム環境は、不揮発性読み取り専用データベースに書き込まれて、開発及びテストプロセスにおける開発環境として使用され得る。所望の変更が開発環境に加えられ、そこでテストされ得る。ユーザまたはコントローラロジックは、グローバルルールに変更を加えて、新たなバージョンを作成し得る。ルールのバージョンは追跡され得る。次に、例示的な実施形態の他の態様によれば、新たに開発された環境が自動的に実装され得る。以前の本番環境も保持されてよく、または完全に機能し得るので、以前の状態の本番環境への改正が、データを失わずに可能である。次に、開発環境は、新たな仕様、ルール、及びテンプレートを用いてブートされてよく、データベースまたはシステムは本番データベースと同期され、書き込み可能なデータベースに切り替えられ得る。そして、元の本番データベースは読み取り専用データベースに切り替えられてよく、復旧が必要な場合に、システムはそれに戻され得る。

【0041】

ソフトウェアのアップグレードまたはパッチ適用に関して、アップグレードまたはパッチを必要とするサービスが検出された場合に、新たなホストが配備され得る。アップグレードまたはパッチによる障害が発生した場合に、上述のように変更を戻すことが可能な場合に、新たなサービスが配備され得る。

【0042】

ハードウェアのアップグレードは、特に最新のハードウェアが不可欠な多くの状況で重要である。このタイプの状況の一例は、ミリ秒単位の速度の利点を有するITシステムにより、ユーザが優れた取引結果及び利益を達成することが可能になり得る高頻度取引業界で発生する。特に、新たなハードウェアがプロトコルを用いて通信し、既存のインフラストラクチャと連携する方法を認識するように、現在のインフラストラクチャとの相互運用性を確保する際に、問題が生じる。コンポーネントの相互運用性を確保することに加えて、コンポーネントは既存のセットアップとの統合を必要とする。

【0043】

図1を参照すると、例示的な実施形態のITシステム100が示されている。システム100は、本明細書に記載のものを含むがこれらに限定されない1つまたは複数のタイプのITシステムであり得る。

【0044】

スタンドアロンの物理または仮想サーバ上に存在する場合としない場合があるアプリケーションプログラムインターフェース(API)アプリケーション120を介してコントローラ200に結合されるユーザインターフェース(UI)110が図示されている。コントローラ200は、本明細書で説明する任意の制御動作を実施するために、1つまたは複数のプロセッサ及び1つまたは複数のメモリ上に配備され得る。そのような制御動作を実行するためにプロセッサ(複数可)によって実行される命令は、プロセッサメモリなどの非

10

20

30

40

50

一時的コンピュータ可読記憶媒体上に存在することができる。API 120は1つまたは複数のAPIアプリケーションを含んでよく、これらは冗長であってよく、及び／または並行して動作し得る。APIアプリケーション120は、システムリソースを構成する要求を受信し、要求を解析して、コントローラ200に渡す。APIアプリケーション120は、コントローラから1つまたは複数の応答を受信し、応答（複数可）を解析して、UI（またはアプリケーション）110に渡す。あるいは、または追加で、アプリケーションまたはサービスが、APIアプリケーション120と通信し得る。コントローラ200は、計算リソース（複数可）300、ストレージリソース（複数可）400、及びネットワークリソース（複数可）500に結合される。リソース300、400、500は、単一のノード上に存在する場合としない場合がある。リソース300、400、500のうちの1つまたは複数のは、仮想であってよい。リソース300、400、500は、複数のノード上に、または複数のノード上に様々な組み合わせで存在する場合としない場合がある。物理デバイスは、計算リソース300、ストレージリソース400、及びネットワークリソース500を含むがこれらに限定されないリソースタイプのうちの1つまたは複数もしくはそれぞれを含み得る。リソース300、400、500はまた、異なる物理的位置にあるか否かにかかわらず、また、仮想的であるか否かにかかわらず、リソースのプールを含み得る。また、ベアメタル計算リソースは、仮想またはコンテナ計算リソースの使用を可能にするために使用され得る。

【0045】

ノードの既知の定義に加えて、本明細書で使用するノードは、スタンドアロンデバイスまたはネットワーク接続されたデバイス上で機能を実行するネットワーク（複数可）または他の機能ユニットに接続された任意のシステム、デバイス、またはリソースであり得る。ノードには、例えば、サーバ、物理ホストもしくは仮想ホスト上のサービス／アプリケーション／複数のサービス、仮想サーバ、及び／またはマルチテナントサーバ上のもしくはコンテナ内で動作する複数または単一のサービスも含まれ得るが、これらに限定されない。

【0046】

コントローラ200は、1つまたは複数の物理または仮想コントローラサーバを含んでよく、これらもまた冗長であってよい、及び／または並列に動作し得る。コントローラは、計算ホストとして機能している物理または仮想ホスト上で動作し得る。一例として、コントローラは、例えば、機密性の高いリソースにアクセスできるように、他の目的にも役立つホスト上で動作するコントローラで構成され得る。コントローラは、APIアプリケーション120から要求を受信し、要求を解析し、他のリソースへのタスク分配を適正化して、他のリソースに指示し、リソースからの情報を監視及び受信し、システムの状態及び変更の履歴を保持し、ITシステム内の他のコントローラと通信し得る。また、コントローラはAPIアプリケーション120を含み得る。

【0047】

本明細書で定義する計算リソースは、現実または仮想の、単一の計算ノード、あるいは1つまたは複数の計算ノードを含むリソースプールを含み得る。計算リソースまたは計算ノードは、1つまたは複数のサービスをホストするか、1つまたは複数のアプリケーションを動作させ得る1つまたは複数の物理または仮想マシンまたはコンテナホストを含み得る。計算リソースは、GPU、ASIC、コプロセッサ、CPU、FPGA、及び他の専門的な計算方法を含むがこれらに限定されない、計算、ストレージ、キャッシング、ネットワーク、専門的な計算を含むがこれらに限定されない複数の目的のために設計されたハードウェア上にあり得る。そのようなデバイスは、PCIエクスプレススイッチまたは同様のデバイスを使用して追加されてよく、そのような方法で動的に追加され得る。計算リソースまたは計算ノードは、サービスもしくはアプリケーションを動作させるかまたは仮想計算リソースであり得る複数の異なる仮想マシンを含む1つまたは複数のハイパーバイザまたはコンテナホストを含んでよく、または動作させ得る。計算リソースは、計算機能を提供することに重点が置かれ得るが、データストレージ機能及び／またはネットワ

ーキング機能も含み得る。

【0048】

本明細書で定義するストレージリソースは、ストレージノードまたはストレージリソースのプールを含み得る。ストレージリソースは任意のデータ記憶媒体、例えば、高速、低速、ハイブリッド、キャッシュ、及び／またはRAMを含み得る。ストレージリソースは、1つまたは複数のタイプのネットワーク、マシン、デバイス、ノード、またはそれらの任意の組み合わせを含んでよく、これらは、他のストレージリソースに直接接続されてもされなくてもよい。例示的な実施形態の態様によれば、ストレージリソースは、ベアメタルもしくは仮想またはそれらの組み合わせであり得る。ストレージリソースは、ストレージ機能を提供することに重点が置かれ得るが、計算機能及び／またはネットワーク機能も含み得る。

10

【0049】

ネットワークングリソース（複数可）500は、単一のネットワークングリソース、複数のネットワークングリソース、またはネットワークングリソースのプールを含み得る。ネットワークングリソース（複数可）は、物理デバイスもしくは仮想デバイス（複数可）、ツール（複数可）、スイッチ、ルータ、もしくはシステムリソース間のその他の相互接続、またはネットワークングを管理するためのアプリケーションを含み得る。そのようなシステムリソースは、物理的または仮想的なものであってよく、計算リソース、ストレージリソース、または他のネットワークングリソースを含み得る。ネットワークングリソースは、外部ネットワークとアプリケーションネットワークとの間の接続を提供してよく、ドメインネームシステム（DNSまたはdns）、動的ホスト構成プロトコル（DHCP）、サブネット管理、レイヤ3ルーティング、ネットワークアドレス変換（NAT）、及び他のサービスを含むがこれらに限定されないコアネットワークサービスをホストし得る。これらのサービスの一部は、物理または仮想マシン上の計算リソース、ストレージリソース、またはネットワークングリソースに配備され得る。ネットワークングリソースは、Infiniband、イーサネット、リモートダイレクトメモリアクセス（DMA）オーバーコンバージドイーサネット（RoCE）、ファイバチャネル、及び／またはOmniPathを含むがこれらに限定されない1つまたは複数のファブリックまたはプロトコルを利用してよく、複数のファブリック間の相互接続を含み得る。ネットワークングリソースは、ソフトウェア定義ネットワーク（SDN）対応であってもなくてもよい。コントローラ200は、SDN、仮想ローカルエリアネットワーク（VLAN）などを使用してネットワークングリソース300を直接変更することによって、ITシステムのトポロジを構成することが可能であり得る。ネットワークングリソースは、ネットワーク機能を提供することに重点が置かれ得るが、計算機能及び／またはストレージ機能も備え得る。

20

30

【0050】

本明細書で使用するアプリケーションネットワークとは、アプリケーション、リソース、サービス、及び／または他のネットワークを接続または結合するための、あるいはユーザ及び／またはクライアントをアプリケーション、リソース、及び／またはサービスにつなげるためのネットワークングリソースまたはそれらの任意の組み合わせを意味する。アプリケーションネットワークは、サーバが他のアプリケーションサーバ（物理または仮想）と通信し、クライアントと通信するために使用されるネットワークを含み得る。アプリケーションネットワークは、システム100の外部のマシンまたはネットワークと通信し得る。例えば、アプリケーションネットワークは、Webフロントエンドをデータベースに接続し得る。ユーザは、インターネットを介して、またはコントローラで管理されている場合もされていない場合もある他のネットワークを介してWebアプリケーションに接続し得る。

40

【0051】

例示的な実施形態によれば、計算リソース300、ストレージリソース400、及びネットワークングリソース500はそれぞれ、コントローラ200によって自動的に追加、

50

削除、セットアップ、割り当て、再割り当て、構成、再構成、及び／または配備され得る。例示的な実施形態によれば、追加のリソースがリソースプールに追加され得る。

【0052】

ユーザ105がそれを介してシステムにアクセスし、システムと対話し得るWeb UIまたは他のユーザインタフェースなどのユーザインタフェース110を図示しているが、あるいは、または追加で、アプリケーションは、APIアプリケーション（複数可）120を介して、または別の方法で、コントローラ200と通信または対話し得る。例えば、ユーザ105またはアプリケーションは、ITシステムの構築、ITシステム内の個々のスタックの構築、サービスまたはアプリケーションの作成、サービスまたはアプリケーションの移行、サービスまたはアプリケーションの変更、サービスまたはアプリケーションの削除、異なるネットワーク上の他のスタックへのスタックのクローン、リソースまたはシステムコンポーネントの作成、追加、削除、セットアップまたは構成、再構成、を含むがこれらに限定されない要求を送信し得る。

10

【0053】

図1のシステム100は、物理的もしくは仮想的またはそれらの任意の組み合わせであり得る様々な要素、コンポーネントまたはリソースとの接続または他の通信インタフェースを有するサーバを含み得る。変形例によれば、図1に示すシステム100は、接続を有するベアメタルサーバを含み得る。

【0054】

本明細書でより詳細に説明するように、コントローラ200は、リソースまたはコンポーネントに電源投入して、リソースのブートアップを自動的にセットアップ、構成、及び／または制御することによって、リソースを追加し、リソースを割り当て、リソースを管理し、利用可能なリソースを更新するように構成され得る。電源投入プロセスは、ブートされるデバイスの順序が一貫しており、ユーザによるデバイスへの電源投入に依存しないように、コントローラに電源投入することから始まり得る。このプロセスは、電源投入されたリソースの検出も含み得る。

20

【0055】

図2A～図10を参照すると、コントローラ200、コントローラロジック205、グローバルシステムルールデータベース210、ITシステム状態220、及びテンプレート230が示されている。

30

【0056】

システム100は、グローバルシステムルール210を含む。グローバルシステムルール210はとりわけ、計算、ストレージ、及びネットワーキングを含み得るリソースをセットアップ、構成、ブート、割り当て、及び管理するルールを宣言し得る。グローバルシステムルール210は、システム100が正しい状態または所望の状態になるための最小要件を含む。これらの要件は、完了が期待されるITタスクと、所望のシステムを予想通りに構築するために必要な期待されるハードウェアの更新可能なリストとを含み得る。期待されるハードウェアの更新可能なリストにより、コントローラは、（例えば、ルールの開始前またはテンプレートの使用前に）必要なリソースが利用可能であることを確認することが可能になる。グローバルルールは、様々なタスクに必要な動作と、動作及びタスクの順序付けに関連する対応する命令とのリストを含み得る。例えば、ルールは、コンポーネントに電源投入する順序、リソース、アプリケーション、及びサービスをブートする順序、依存関係、様々なタスクをいつ開始すべきか、例えば、アプリケーションのロード、構成、開始、リロード、またはハードウェアの更新をいつ開始すべきかを指定し得る。ルール210はまた、例えば、アプリケーション及びサービスに必要なリソース割り当てのリスト、使用され得るテンプレートのリスト、ロードすべきアプリケーション及び構成方法のリスト、ロードすべきサービス及び構成方法のリスト、アプリケーションネットワーク及びどのアプリケーションがどのネットワークに適合するかのリスト、様々なアプリケーションに固有の構成変数及びユーザ固有のアプリケーション変数のリスト、コントローラがシステム状態をチェックして、状態が期待通りであり、各命令の結果が期待通りであ

40

50

ることを確認できるようになる期待される状態、及び／またはルールの変更の追跡と、異なる状況で異なるルールをテストするかまたはそれらに戻る機能とを可能にし得るルールの変更のリスト（例えば、スナップショット）を含むバージョンリスト、のうちの1つまたは複数を含み得る。コントローラ200は、物理リソース上のITシステム100にグローバルシステムルール210を適用するように構成され得る。コントローラ200は、仮想リソース上のITシステム100にグローバルシステムルール210を適用するように構成され得る。コントローラ200は、物理リソースと仮想リソースとの組み合わせ上のITシステム100にグローバルシステムルール210を適用するように構成され得る。
【0057】

図2Mに、グローバルシステムルールの形を取り得るシステムルール210の例示的なセットを示す。図2Mに示すシステムルール210の例示的なセットは、コントローラ200にロードされるか、またはシステム状態にクエリすることによって導出され得る（210.1参照）。図2Mの例では、システムルール210は、構成ルーチン210.2の形をとることができる命令のセットを含み、また、ITシステムまたは環境を作成及び／または再作成するためのデータ210.3も含む。システムルール210内の構成ルールは、要求されたテンプレートリスト210.7を介してテンプレート230を見つける方法を認識し得る（テンプレート230は、ファイルシステム、ディスク、ストレージリソース内に存在してよい、またはシステムルール内に置かれてよい）。コントローラロジック205はまた、テンプレート230を処理する前にテンプレート230を探し、テンプレート230が存在することを確認した後にシステムルール210を有効にし得る。システムルール210は、システムルールのサブセット210.15を含んでよく、これらのサブセット210.15は、構成ルーチン210.2の一部として実行され得る。

【0058】

また、サブシステムルール210.15は、例えば、統合ITアプリケーションのシステムを構築するためのツールとして使用することができる（その場合、システムルール実行ルーチン210.16によって処理されて、210.15の追加を反映するようにシステム状態及び現在の構成ルールが更新される）。また、サブシステムルール210.15は、他の場所に配置され、ユーザインタラクションによってシステム状態220内にロードされ得る。例えば、サブシステムルール210.15をプレイブックとして有することもでき、それらを利用可能にして動作させることができる（その後グローバルシステムルール210が更新されるので、システムをクローンしたい場合にプレイブックをリプレイすることができる）。

【0059】

構成ルーチン210.2は、システムを構築するために使用される命令のセットであってよい。構成ルーチン210.2は、実施者が望む場合、サブシステムルール210.15またはシステム状態ポインタ210.8も含み得る。構成ルーチン210.2を実行する場合、コントローラロジック205は一連のテンプレートを特定の順序（210.9）で処理することによって、任意選択で並列配備を可能にするが、適切な依存関係の処理（210.12）は維持することができる。構成ルーチン210.2は任意選択で、210.9に従ってテンプレートを処理することによって構成され得るアプリケーションの構成パラメータ210.5を設定し得るAPIコール210.10を呼び出し得る。また、要求されたサービス210.11は、システムがAPIコール（複数可）210.10を行う場合に、立ち上がって稼働している必要があるサービスである。

【0060】

ルーチン210.2は、データのコピー、計算リソースへのデータベースの転送、計算リソースとストレージリソースとのペアリング、及び／または揮発性データ210.6の場所によるシステム状態220の更新を含み得るが、これらに限定されない、揮発性データ210.6に関するデータロード（210.13）のための手順、プログラム、または方法を含み得る。揮発性データへのポインタ（210.4参照）をデータ210.3と共に保持して、他の場所に記憶され得る揮発性データを見つけることができる。データロー

ドルーチン 210.13 は、非標準データストアに配置されている（例えば、データベースに含まれている）場合、構成パラメータ 210.5 のロードにも使用され得る。

【0061】

システムルール 210 はまた、どのコンポーネントがどのリソースに割り当てられるかを指示することができ、コントローラロジック 205 が適切なリソース及び／またはハードウェアが利用可能であるかどうかを判定することを可能にする、リソースリスト 210.18 を含むことができる。システムルール 210 はまた、代替的な配備のための（例えば、ソフトウェアエンジニアが実演のテストを行うことを望むが、データセンタ全体を割り当ててを望まない場合がある、開発環境のための）代替的なハードウェア及び／またはリソースリスト 210.19 も含み得る。システムルールはまた、システムをどのようにバックアップし、冗長化のためにスタンバイをどのように使用するかについての命令を提供するデータバックアップ及び／またはスタンバイルーチン 210.17 を含み得る。バックアップルールを実施するデータバックアップシステム及び／またはバックアップルーチンの例には、図 21A～21J を参照して本明細書に説明するものが含まれるが、これらに限らない。

【0062】

あらゆるアクションが取られた後、システム状態 220 は更新されてよく、クエリ（書き込みを含み得る）がシステム状態クエリ 210.14 として保存されてよい。

【0063】

図 2N は、コントローラロジック 205 が図 2M のシステムルール 210（または、サブシステムルール 210.15）を処理する例示的なプロセスフローを示す。ステップ 210.20 で、コントローラロジック 205 は、適切なリソースが利用可能であることをチェックして確認する（図 2M の 210.18 を参照）。そうでなければ、ステップ 210.21 で、代替的な構成がチェックされてもよい。第 3 のオプションは、ユーザが、図 2M のリスト 210.7 において参照されるテンプレート 230 によってサポートし得る代替的な構成の選択を促進されることを含み得る。

【0064】

ステップ 210.22 で、コントローラロジックは次に、計算リソース（または、適切なリソースのいずれか）が揮発性データにアクセスすることを確認してよい。これは、ストレージリソースに接続すること、またはストレージリソースをシステム状態 220 に追加することを伴ってよい。ステップ 210.23 で、次に、構成ルーチンが処理され、各ルーチンが処理されると、システム状態 220 が更新される（ステップ 210.24）。システム状態 220 はまた、処理する前に特定のステップが終了しているかどうかをチェックするようクエリされてよい（ステップ 210.25）。

【0065】

図 210.23 に示す構成ルーチン処理ステップは、210.26 の手順のいずれか（または、それらの組み合わせ）を含み得る。それはまた、他の手順も含んでよい。例えば、210.26 における処理は、テンプレート処理（210.27）、構成データをロードすること（210.28）、静的データをロードすること（210.29）、動的揮発性データをロードすること（210.30）、及び／または、サービス、アプリケーション、サブシステム及び／または環境を結合すること（210.31）を含んでよい。210.26 内のそのような手順は、いくつかのシステムコンポーネントが独立しており、他のコンポーネントが依存しているので、ループで繰り返されてよい、または並列に実行されてよい。コントローラロジック、サービス依存関係、及び／またはシステムルールは、どのサービスが相互に依存することができるかを指示してよく、システムルールから IT システムをさらに構築するようサービスを結合してもよい。

【0066】

グローバルシステムルール 210 は、ストレージ拡張ルールも含んでよい。ストレージ拡張ルールは、例えば、システム内の既存のストレージリソースにストレージリソースを自動で追加するルールのセットを提供する。さらに、計算リソース（複数可）上で実行す

10

20

30

40

50

るアプリケーションが、ストレージ拡張を要求するときを把握する（または、コントローラ 200 が、計算リソースまたはアプリケーションのストレージを拡大すべきときを把握することができる）トリガポイントを設けてもよい。コントローラ 200 は、新たなストレージリソースを割り当て及び管理してよく、特定の実行中のリソースに対してストレージリソースを既存のストレージリソースとマージまたは統合してもよい。そのような特定の実行中のリソースは、システム内の計算リソース、システム内で計算リソースを実行しているアプリケーション、仮想マシン、コンテナ、または物理もしくは仮想計算ホスト、あるいはそれらの組み合わせであってよいが、これらに限定されない。実行中のリソースは、例えば、ストレージ空間クエリを通じて、ストレージ空間を使い果たそうとしていることをコントローラ 200 に知らせてよい。インバンド管理接続 270、SAN 接続 280、またはコントローラ 200 へのいずれかのネットワーキングもしくは結合が、そのようなクエリにおいて使用されてよい。アウトオブバンド管理接続 260 も使用されてよい。実行していないリソースに対しても、それらのストレージ拡張ルール（または、それらのストレージ拡張ルールのサブセット）が使用されてよい。

【0067】

ストレージ拡張ルールは、システム内で新たなストレージリソースをどのように見つけ、接続し、セットアップするかを指示する。コントローラは、新たなストレージリソースをシステム状態 220 に登録し、ストレージリソースがどこに存在するか、及びそれにどのように接続するかを実行中のリソースに通知する。実行中のソースは、そのような登録情報を使用してストレージリソースに接続する。コントローラ 200 は、新たなストレージリソースを既存のストレージリソースにマージしてよい、または、新たなストレージリソースをボリュームグループに追加してよい。

【0068】

図 2 B は、例示的なストレージ拡張ルールのセットの動作の例示的なフローを示す。ステップ 210.41 で、実行中のリソースは、トリガポイントまたはその他に基づいて、ストレージが少ないと判定する。ステップ 210.42 で、実行中のリソースは、インバンド管理接続 270、SAN 接続 280、またはオペレーティングシステムに対して可視的な別のタイプの接続によってコントローラ 200 に接続する。この接続を通して、実行中のリソースは、ストレージが少ないことをコントローラ 200 に通知することができる。ステップ 210.43 で、コントローラは、実行中のリソースに対してストレージ容量を増大するようストレージリソースを構成する。ステップ 210.44 で、コントローラは、新たに構成されたストレージリソースがどこに位置するかに関する情報を実行中のリソースに提供する。ステップ 210.45 で、実行中のリソースは、新たに構成されたストレージリソースに接続する。ステップ 210.46 で、コントローラは、新たなストレージリソースの位置のシステム状態 220 へのマッピングを追加する。次に、コントローラは、新たなストレージリソースを、実行中のリソースに割り当てられたボリュームグループに追加することができる（ステップ 210.47）、またはコントローラは、新たなストレージリソースの実行中のリソースへの割り当てをシステム状態 220 に追加することができる（ステップ 210.48）。

【0069】

図 2 C は、図 2 B におけるステップ 210.41 及び 210.42 を実行する代替例を示す。ステップ 210.50 で、コントローラは、実行中のリソースに対するストレージ状態更新のためのモニタまたはコンソールを閲覧するよう、アウトオブバンド管理接続 260 を通してキーコマンドを送信する。例えば、モニタは、ipmi コンソールであってよく、ipmi コンソールを通してアウトオブバンド接続 260 を介して画面を見直すことができる。一例として、アウトオブバンド接続 260 は、キーボード／マウスとして USB につなげることができ、VGA モニタポートにつなげることができる。ステップ 210.51 で、実行中のリソースは、画面上に情報を表示する。ステップ 210.52 で、コントローラは次に、アウトオブバンド管理接続 260 及び画面スクレイプまたは同様の操作を介してモニタまたはコンソール上に提示された情報を読み出し、この読み出された

情報は、トリガポイントに基づいて低ストレージ状態を示し得る。プロセスフローは次に、図 2 B のステップ 2 1 0 . 4 3 を続けてよい。

【 0 0 7 0 】

図 2 D は、図 2 B におけるステップ 2 1 0 . 4 1 及び 2 1 0 . 4 2 を実行する別の代替例を示す。ステップ 2 1 0 . 5 5 で、実行中のリソースは、コントローラによって読み出すために情報をモニタまたはコンソール上に自動的に表示する。ステップ 2 1 0 . 5 6 で、コントローラは、実行中のリソースをチェックするためにモニタまたはコンソールを自動的に、定期的に、または絶えず読み出す。この読み出しに応答して、コントローラは、実行中のリソースが、ストレージが少ないと確認する（ステップ 2 1 0 . 5 7 ）。プロセスフローは次に、図 2 B のステップ 2 1 0 . 4 3 を続けてよい。

10

【 0 0 7 1 】

コントローラ 2 0 0 はまた、ベアメタル及び／またはサービステンプレートを含み得るテンプレート 2 3 0 のライブラリを含む。それらのテンプレートは、電子メール、ファイルストレージ、ボイスオーバー IP、ソフトウェアアカウンティング、ソフトウェア X M P P、w i k i、バージョン制御、アカウント認証管理、及びユーザインタフェースによって構成可能であってよい第三者アプリケーションを含み得るが、これらに限定されない。テンプレート 2 3 0 は、リソース、アプリケーション、またはサービスとの関連付けを有することができ、且つ、そのようなリソース、アプリケーション、またはサービスがシステムにどのように統合されるかを定義するレシピとして機能することができる。

【 0 0 7 2 】

20

このように、テンプレートは、リソース、またはリソースに対してロードされたアプリケーションもしくはサービスを作成、構成、及び／または配備するために使用される確立された情報のセットを含み得る。そのような情報は、カーネル、i n i t r d ファイル、ファイルシステム、もしくはファイルシステムイメージ、ファイル、構成ファイル、構成ファイルテンプレート、異なるハードウェア及び／もしくは計算バックエンドのための適切なセットアップを判定するために使用される情報、及び／またはアプリケーションの作成、ブート、もしくは実行を可能及び／または容易にするアプリケーション及びオペレーティングシステムイメージを動かすためのリソースを構成するための他の利用可能なオプションを含んでよいが、これらに限定されない。

【 0 0 7 3 】

30

テンプレートは、複数の物理サーバタイプまたはコンポーネント、複数のハードウェアタイプ上で実行する複数のハイパーバイザ、複数のハードウェアタイプ上でホストすることができるコンテナホストを含むが、これらに限定されない、複数のサポートされたハードウェアタイプ／及びまたは計算バックエンド上にアプリケーションを配備するために使用し得る情報を含み得る。

【 0 0 7 4 】

テンプレートは、計算リソース上で実行するアプリケーションまたはサービスのブートイメージを導出し得る。テンプレート及びテンプレートから導出されたイメージを使用して、アプリケーションを作成し、アプリケーションもしくはサービスを配備し、及び／または様々なシステム機能にリソースを準備してよく、これは、アプリケーションの作成を可能及び／または促進する。テンプレートは、デフォルト設定またはコントローラから与えられた設定のいずれかからの構成オプションにより上書きし得るファイル、ファイルシステム、及び／またはオペレーティングシステムイメージにおける可変パラメータを有してよい。テンプレートは、アプリケーションまたは他のリソースを構成するために使用される構成スクリプトを有してよく、テンプレートは、構成変数、構成ルール、及び／またはデフォルトルールもしくは変数を利用してよく、それらのスクリプト、変数、及び／またはルールは、特定のハードウェアまたは他のリソース固有のパラメータ、例えば、ハイパーバイザ（仮想のとき）、利用可能なメモリについての特定のルール、スクリプト、または変数を含み得る。テンプレートは、バイナリリソースの形式のファイル、バイナリリソース、またはハードウェアもしくは他のリソース固有のパラメータをもたらすコンパイ

40

50

ル可能ソースコード、バイナリリソースの特定のセット、あるいは特定のハードウェアまたは他のリソース特有パラメータ、例えば、ハイパーバイザ（仮想のとき）、利用可能なメモリについてのコンパイル命令を有するソースコードを有してよい。テンプレートは、リソース上で実行しているものとは独立した情報のセットを含み得る。

【0075】

テンプレートは、ベースイメージを含み得る。ベースイメージは、ベースオペレーティングシステムファイルシステムを含み得る。ベースオペレーティングシステムは、読み出し専用であってよい。ベースイメージはまた、実行中のものとは独立したオペレーティングシステムの基本ツールを含み得る。ベースイメージは、ベースディレクトリ及びオペレーティングシステムツールを含み得る。テンプレートは、カーネルを含み得る。カーネルまたは複数のカーネルは、`initrd`カーネル、または異なるハードウェアタイプ及びリソースタイプに対して構成された複数のカーネルを含み得る。イメージは、テンプレートから導出され、1つまたは複数リソースにロードされ、または配備されてよい。ロードされたイメージは、対応するテンプレートのカーネルまたは`initrd`のカーネルなどのブートファイルも含み得る。

【0076】

イメージは、テンプレートに基づいてリソースにロードし得るテンプレートファイルシステム情報を含んでよい。テンプレートファイルシステムは、アプリケーションまたはサービスを構成し得る。テンプレートファイルシステムは、例えば、ファイルシステムが記憶されるストレージ空間を節約し、または読み出し専用ファイルの使用を容易にするように、全てのリソースまたは同様のリソースに共通な共有ファイルシステムを含み得る。テンプレートファイルシステムまたはイメージは、配備されるサービスに共通のファイルのセットを含み得る。テンプレートファイルシステムは、コントローラ上で予めロードされていてもよく、またはダウンロードされてもよい。テンプレートファイルシステムは、更新されてよい。テンプレートファイルシステムは、再構築することを必要としないことがあるので、相対的に素早い配備を可能にし得る。他のリソースまたはアプリケーションとファイルシステムを共有すると、ファイルが不必要に複製されないので、ストレージの削減を可能にし得る。これはまた、テンプレートファイルシステムとは異なるファイルだけを回復すればよいので、障害からのより容易な復旧を可能にし得る。

【0077】

テンプレートブートファイルは、カーネル及び／または`initrd`もしくはブートプロセスを支援するために使用される同様のファイルシステムを含み得る。ブートファイルは、オペレーティングシステムをブートしてよく、テンプレートファイルシステムをセットアップしてよい。`initrd`は、ブートすることができるようテンプレートをどのようにセットアップするかに関する命令を有する小規模一時ファイルシステムを含み得る。

【0078】

テンプレートはさらに、BIOS設定を含み得る。テンプレートBIOS設定は、物理ホスト上でアプリケーションを実行するための任意選択の設定を設定するために使用されてよい。使用される場合、図1～12に関して本明細書で説明されるように、アウトオブバンド管理260を使用して、リソースまたはアプリケーションをブートしてよい。物理ホストは、アウトオブバンド管理ネットワーク260またはCDROMを使用して、リソースまたはアプリケーションをブートしてよい。コントローラ200は、そのようなテンプレートにおいて定義されたアプリケーション固有のBIOS設定を設定してよい。コントローラ200は、特定のリソースに固有のAPIを通して、直接のBIOS変更を行うためにアウトオブバンド管理システムを使用してよい。設定は、コンソール及び画像認識を通して検証されてよい。したがって、コントローラ200は、コンソール機能を使用してよく、仮想キーボード及びマウスによりBIOS変更を行ってよい。コントローラはまた、UEFIシェルを使用してよく、コンソールに直接タイプしてよく、成功した結果を検証し、コマンドを正確にタイプし、設定変更の成功を保証するために画像認識を使用してよい。特定のBIOSバージョンへのBIOS変更または更新に利用可能なブート可能

10

20

30

40

50

なオペレーティングシステムが存在する場合、コントローラ 200 は、BIOS を更新し、信頼できる方式で構成変更を可能にするアプリケーションをオペレーティングシステムが実行するディスクイメージまたは ISO ブートをリモートにロードしてよい。

【0079】

テンプレートはさらに、テンプレート固有のサポートされるリソースのリストまたは特定のアプリケーションもしくはサービスを実行するために必要とされるリソースのリストを含み得る。

【0080】

テンプレートイメージ、またはイメージもしくはテンプレートの一部は、コントローラ 200 に記憶されてよい、あるいはコントローラ 200 は、ストレージリソース 410 にそれを移動させてよい、またはコピーしてよい。

【0081】

図 2 E は、例示的なテンプレート 230 を示す。テンプレートは、アプリケーションまたはサービスを作成するために必要な全ての情報を含む。テンプレート 230 はまた、類似または同一の機能性を提供する異なるハードウェアタイプについての情報、代替的なデータ、ファイル、バイナリを含み得る。例えば、異なるアーキテクチャに対してコンパイルされたバイナリ 234 により、`/usr/bin` 及び `/bin` についてのファイルシステム `blob 232` が存在し得る。テンプレート 230 はまた、デーモン 233 またはスクリプト 231 を含み得る。デーモン 233 は、ホストが電源投入され、準備完了状態になるブート時に実行し得るバイナリまたはスクリプトであり、いくつかのケースでは、デーモン 233 は、コントローラによってアクセス可能であってもよく、コントローラがホストの設定を変更することを可能にし得る（及び、コントローラは続いて、アクティブシステムルールを更新してよい）API を動かしてよい。デーモンは、上記及び以下で説明するアウトオブバンド管理 260 またはインバンド管理 270 を通して電源を切られてよく、再起動されてよい。それらのデーモンはまた、新たなサービス（例えば、`nginx` または `apache` を制御する API と通信する一般的なウェブサーバ API）に対して依存するサービスを提供するよう、一般的な API を動かしてよい。スクリプト 231 は、イメージをブートしている間もしくは後、またはデーモンを開始した後、もしくはサービスを有効にした後に実行し得るインストールスクリプトであってよい。

【0082】

テンプレート 230 はまた、カーネル 235 及びプリブートファイルシステム 236 を含み得る。テンプレート 230 はまた、異なるハードウェア及び異なる構成についての複数のカーネル 235 及び 1 つまたは複数の事前ブートファイルシステム（Linux の `initrd` もしくは `initramfs`、または `bsd` の読み出し専用 RAM ディスクなど）を含み得る。`initrd` はまた、以下で説明するように、任意選択で SAN 接続 280 を通してストレージリソースに接続することができる `initramfs 236` にブートすることによって、オーバレイとして提示されるファイルシステム `blob 232` をマウントし、リモートストレージ上でルートファイルシステムをマウントするために使用されてよい。

【0083】

ファイルシステム `blob 232` は、別個の `blob` に分割し得るファイルシステムイメージである。`blob` は、構成オプション、ハードウェアタイプ、及びセットアップにおける他の差異に基づいて相互に変更可能であってよい。テンプレート 230 からブートされるホストは、1 つまたは複数のファイルシステム `blob` から作成された複数の `blob` またはイメージを含むユニオンファイルシステム（`overlayfs` など）からブートされてよい。

【0084】

テンプレート 230 はまた、揮発性データ 238 及び／または構成パラメータ 239 などの追加の情報 237 を含んでよい、または追加の情報 237 とリンクされてよい。例えば、揮発性データ 238 は、テンプレート 230 に含まれてよい、または、外部に含まれ

10

20

30

40

50

てよい。揮発性データ 238 は、ファイルシステム blob 232、またはデータベース、フラットファイル、ディレクトリに記憶されたファイル、ファイルの tarball、ギット、もしくは他のバージョン制御リポジトリを含む他のデータストア形式のものであってよいが、これらに限定されない。さらに、構成パラメータ 239 は、テンプレート 230 の外部または内部に含まれてよく、任意選択で、システムルールに含まれ、テンプレート 230 に適用される。

【0085】

システム 100 はさらに、リソースを含むが、これに限定されない、システム 100 の状態を追跡し、維持し、変更し、及び更新する IT システム状態 220 を含む。システム状態 220 は、利用可能なリソースを追跡してよく、それは、ルールの実装とテンプレートとに利用可能なリソースがあるかどうかと、どのリソースが利用可能であることをコントローラロジックに通知する。システム状態は、使用されたリソースを追跡してよく、それは、アップグレードのため、または効率性改善もしくは優先度などの他の理由により、切り替えの必要性があるかどうかに関わらず、コントローラロジック 205 が効率性を検査し、効率性を利用することを可能にする。システム状態は、どのアプリケーションが実行しているかを追跡してよい。コントローラロジック 205 は、システム状態、及び改定する必要があるかどうかに従って、期待される実行するアプリケーションと実行中の実際のアプリケーションと比較してよい。システム状態 220 はまた、アプリケーションがどこで実行しているかを追跡してよい。コントローラロジック 205 は、効率性、変更管理、更新、トラブルシューティング、または監査証跡を評価する目的でこの情報を使用してよい。システム状態は、ネットワーキング情報、例えば、どのネットワークがオンもしくは現在実行中か、または構成値及び履歴を追跡してよい。システム状態 220 は、変更の履歴も追跡してもよい。システム状態 220 はまた、どのテンプレートが使用されるかを規定したグローバルシステムルールに基づいて、どのテンプレートがどの配備において使用されているかを追跡してよい。履歴は、監査、警告、変更管理、構築レポート、ハードウェア及びアプリケーション及び構成と相関付けられた追跡バージョン、または構成変数のために使用されてよい。システム状態 220 は、監査、コンプライアンステスト、またはトラブルシューティングの目的のために構成の履歴を維持してよい。

【0086】

コントローラは、システム状態、テンプレート、及びグローバルシステムルールに含まれた全ての情報を管理するためのロジック 205 を有する。コントローラロジック 205、グローバルシステムルールデータベース 210、IT システム状態 220、及びテンプレート 230 は、コントローラ 200 によって管理され、コントローラ 200 に存在する場合もしない場合もある。コントローラロジックまたはアプリケーション 205、グローバルシステムルールデータベース 210、IT システム状態 220、及びテンプレート 230 は、物理的であっても仮想であってもよく、分散サービス、分散データベース、及び／またはファイルであってもよく、またはそうでなくてもよい。API アプリケーション 120 は、コントローラロジック／コントローラアプリケーション 205 と共に含まれてよい。

【0087】

コントローラ 200 は、スタンドアロンマシンマシンを実行してよい、及び／または 1 つまたは複数のコントローラを含み得る。コントローラ 200 は、コントローラサービスまたはアプリケーションを含んでよく、別のマシン内で実行してよい。コントローラマシンは、スタックの全体またはスタックのグループのブートを順序付け及び／または一貫させることを保証するよう、コントローラサービスを最初に起動してよい。

【0088】

コントローラ 200 は、計算リソース、ストレージリソース、及びネットワーキングリソースを有する 1 つまたは複数のスタックを制御してよい。各スタックは、グローバルシステムルール 210 内のルールの異なるサブセットによって制御されてもよく、または制御されなくてもよい。例えば、事前作成、作成、開発、検査スタック、並列、バックアッ

10

20

30

40

50

プ、及び／またはシステム内で異なる機能を有する他のスタックが存在してよい。

【0089】

コントローラロジック205は、所望のITシステム状態を達成するよう、グローバルシステムルールを読み出し、解釈するように構成されてよい。コントローラロジック205は、アプリケーションまたはサービスなどのシステムコンポーネントを構築し、所望のITシステム状態を達成するよう、リソースを割り当て、追加し、または削除するために、グローバルルールに従ったテンプレートを使用するように構成されてよい。コントローラロジック205は、グローバルシステムルールを読み出し、正しい状態に到達するためのタスクリストを開発し、利用可能な操作に基づいてルールを満たす命令を発してよい。コントローラロジック205は、操作を実行するためのロジック、例えば、システムを起動し、リソースを追加、削除、再構成し、それを行うために何が利用可能であるかを識別するためのロジックを含み得る。コントローラロジックは、起動時に、及び定期的な間隔で、システム状態をチェックして、ハードウェアが利用可能であるかどうかを確認してよく、利用可能な場合、タスクを実行してよい。必要なハードウェアが利用可能でない場合、コントローラロジック205は、グローバルシステムルール210、テンプレート220、及びシステム状態230からの利用可能なハードウェアを使用して、代替的なオプションを提示し、それに従って、グローバルルール及び／またはシステム状態220を修正してよい。

10

【0090】

コントローラロジック205は、どの変数が必要とされるか、継続するのにユーザが何を入力することを必要とするか、または機能するためにシステムにおいてユーザが何を必要とするかを認識し得る。コントローラロジックは、グローバルシステムルールからのテンプレートのリストを使用してよく、システム状態において必要とされるテンプレートと比較して、必要とされるテンプレートが利用可能であることを確認してよい。コントローラロジック205は、システム状態データベースから、テンプレート固有のサポートされるリソースのリスト上のリソースが利用可能であるかどうかを識別してよい。コントローラロジックは、リソースを割り当ててよく、状態を更新してよく、グローバルルールを実装するようタスクの次のセットに進んでよい。コントローラロジック205は、グローバルルールにおいて指定されたような割り当てられたリソース上でアプリケーションを開始／実行してよい。ルールは、テンプレートからアプリケーションをどのように構築するかを指定し得る。コントローラロジック205は、テンプレート（複数可）を獲得してよく、変数からアプリケーションを構成してよい。テンプレートは、どのカーネル、ブートファイル、ファイルシステム、及びサポートされたハードウェアリソースが必要とされるかをコントローラロジック205に通知してよい。次に、コントローラロジック205は、アプリケーション配備に関する情報をシステム状態データベースに追加してよい。各命令の後、コントローラロジック205は、予測される動作が正確に完了したかどうかを検証するよう、グローバルルールの予測される状態に対してシステム状態データベースをチェックしてよい。

20

30

【0091】

コントローラロジック205は、バージョンルールに従ったバージョンを使用してよい。システム状態220は、異なる配備においてどのルールバージョンが使用されているかを相関付けるデータベースを有してよい。

40

【0092】

コントローラロジック205は、ルール最適化に対する効率的なロジック及び効率的な順序を含み得る。コントローラロジック205は、リソースを最適化するように構成されてよい。実行中の、または実行が予測されるアプリケーションに関連するシステム状態の情報、ルール、及びテンが、リソースに対する効率性または優先度を実装するために、コントローラロジックによって使用されてよい。コントローラロジック205は、アップグレード、再利用、または別の目的のためにリソースを切り替える効率性または必要性を判定するために、システム状態220において「使用済みリソース」における情報を使用し

50

てよい。

【0093】

コントローラは、システム状態220に従って実行中のアプリケーションをチェックしてよく、グローバルルールの実行が予測されたアプリケーションと比較してよい。アプリケーションが実行中でない場合、アプリケーションを開始してよい。アプリケーションが実行すべきでない場合、それを停止してよく、適切な場合にリソースを再割り当てしてよい。コントローラロジック205は、リソース（計算、ストレージ、ネットワーキング）仕様のデータベースを含み得る。コントローラロジックは、使用することができるシステムに対して利用可能なリソースタイプを認識するためのロジックを含み得る。これは、アウトオブバンド管理ネットワーク260を使用して実行されてよい。コントローラロジック205は、アウトオブバンド管理260を使用して新たなハードウェアを認識するように構成されてよい。コントローラロジック205はまた、監査、レポートの構築、及び変更管理の目的のため、変更の履歴、使用されるルール、及びバージョンに関する情報をシステム状態220から取り出してよい。

【0094】

図2Fは、テンプレート230を処理し、この例示的な目的のためにホストと称され得るリソースをブートし、電源投入し、及び／または有効にするイメージを導出することに関するコントローラロジック205の例示的なプロセスフローを示す。このプロセスはまた、ストレージリソースを構成することと、ストレージホスト及び計算ホスト及び／またはリソースを結合することを含み得る。コントローラロジック205は、システム100において利用可能なハードウェアリソースを把握し、システムルール210は、どのハードウェアリソースが利用できるかを示してよい。コントローラロジック205は、ステップ205.1で、テンプレート230を解析し、テンプレート230は、コントローラロジックに、図2Eに示されたテンプレート230の外部にあるファイルを収集させるよう実行され得る命令ファイルを含んでよい。命令ファイルは、jsonフォーマットであってよい。ステップ205.2で、コントローラロジックは、必要なファイルバケットのリストを収集する。また、ステップ205.3で、コントローラロジック205は、ハードウェアによって、及び任意選択で、ハイパーバイザ（または、コンテナホストシステム、マルチテナンシタイプ）によって参照される、必要なハードウェア固有ファイルをバケットに収集する。ハイパーバイザ（または、コンテナホストシステム、またはマルチテナンシタイプ）の参照は、ハードウェアが仮想マシン上で実行することになる場合に必要になり得る。

【0095】

ハードウェア固有ファイルが存在する場合、コントローラロジックは、ステップ205.4においてハードウェア固有ファイルを収集する。いくつかのケースでは、ファイルシステムイメージは、カーネルモジュール（または、最終的にディレクトリに置かれるカーネルモジュール）を含むディレクトリと共にカーネル及びinitramfsを含み得る。コントローラロジック205は次に、ステップ205.5において互換性を有する適切なベースイメージを選択する。ベースイメージは、アプリケーションまたはテンプレート230から導出されたイメージに固有でない場合もあるオペレーティングシステムファイルを含む。この文脈における互換性は、動作するアプリケーションにテンプレートを変えるために必要なファイルをベースイメージが含むことを意味する。ベースイメージは、空間を節約するための機構としてテンプレートの外部で管理されてよい（また、多くの場合、ベースイメージは、いくつかのアプリケーションまたはサービスに対して同一であってよい）。さらに、ステップ205.6で、コントローラロジック205は、実行ファイル、ソースコード、及びハードウェア固有構成ファイルを有するバケット（複数可）を選択する。テンプレート230は、構成ファイル、構成ファイルテンプレート（コントローラ200が構成テンプレートを構成ファイルに変えることができ、任意選択で、APIエンドポイントを通して構成ファイルを変更し得るように、テンプレート230において既知とされてよいシステムルール210における変数により満たされるプレースホルダまたは

10

20

30

40

50

変数を含む構成ファイルである）、バイナリ、及びソースコード（イメージがブートされるときにコンパイルされてよい）を含むが、これらに限定されない他のファイルを参照してよい。ステップ205.7で、ステップ205.4、205.5、及び205.6において選択された要素に対応するハードウェア固有命令は、ブートされるイメージの一部としてロードされてよい。コントローラロジック205は、選択されたコンポーネントからイメージを導出する。例えば、物理ホスト対仮想マシンについての異なるプリインストールスクリプト、またはPowerpc対x86についての差異が存在してよい。

【0096】

ステップ205.8で、コントローラロジック205は、overlayfsをマウントし、対象ファイルを単一のファイルシステムblobに再パッケージ化する。複数のファイルシステムblobが使用されるとき、複数のblobによりイメージが作成されてよく、tarballを解凍し、及び／またはギットをフェッチする。ステップ205.8が実行されない場合、ファイルシステムblobは、別個のままであってよく、イメージが、ファイルシステムblobのセットとして作成され、複数のより小規模のファイルシステム（overlayfsなど）と一緒にマウントすることが可能なファイルシステムを用いてマウントされる。コントローラロジック205は次に、ステップ205.9において互換性を有するカーネル（または、システムルール210において指定されたカーネル）を見つけてよく、ステップ205.10において適用可能なinitrdを見つけてよい。互換性を有するカーネルは、テンプレート及びテンプレートを実装するために使用されるリソースの依存関係を満たすカーネルであってよい。互換性を有するinitrdは、テンプレートを所望の計算リソースにロードするinitrdであってよい。多くの場合、initrdは、完全にブートする前に、ストレージリソースをマウントできるように（ルートファイルシステムがリモートであり得るので）、物理リソースに使用されてよい。カーネル及びinitrdは、ファイルシステムblobにパッケージ化されてよく、ファイルシステムblobは、予備のオペレーティングシステムをブートした後に稼働中のシステム上のカーネルを変更するために、kexecを使用して直接カーネルブートに対して使用されてよく、または物理ホスト上で使用されてよい。

【0097】

コントローラは次に、計算リソース（複数可）が205.11、205.12、及び／または205.13によって示された技術のいずれかを使用して、アプリケーション（複数可）及び／またはイメージ（複数可）を動かすことを可能にするようストレージリソース（複数可）を構成する。205.11により、overlayfsファイルは、ストレージリソースとして提供することができる。205.12により、ファイルシステムが提示される。例えば、ストレージリソースは、組み合わせられたファイルシステム、またはoverlayfsと同様のファイルシステムを使用して計算リソースが同時にマウントし得る複数のファイルシステムblobを提示してよい。205.13により、blobは、ファイルシステムを提示する前にストレージリソースに送信される。

【0098】

図2G及び2Hは、図2Fのステップ205.11及び205.12についての例示的なプロセスフローを示す。さらに、システムは、ストレージ接続プロセスと称してよい、コンピュータリソースをストレージリソースに接続するためのプロセス及びルールを採用することができる。図2G及び2Hによって示されたプロセスに加えてそのようなストレージ接続プロセスの例が、明細書に含まれた付属書Aにおいて提供される。図2Gは、ストレージリソースの接続のための例示的なプロセスフローを示す。いくつかのストレージリソースは読み出し専用であってよく、その他は、書き込み可能であってよい。ストレージリソースは、競争状態を生じさせる同時書き込みが存在しないように、自身の書き込みロックを管理してよい、またはシステム状態220は、どの接続がストレージリソースに書き込み得るかを追跡してよい（例えば、ステップ205.20を参照）、及び／またはリソースへの複数の読み出し－書き込み接続を防いでよい（ステップ205.21）。コントローラロジックまたはリソース自体は、ストレージリソースの位置及び伝送タイプ（

10

20

30

40

50

例えば、インターネット小型コンピュータシステムインタフェース (ISCSI、iSCSI、またはiscsi)、ISCSIEクステンション・フォア・リモートダイレクトメモリアクセス (RDMAまたはrdma) (ISER、iser、またはiser)、Non-Volatile Memory Express over Fabrics (NVMEOFまたはnvmeof)、ファイバチャネル (FCまたはfc)、ファイバチャネルオーバーイーサネット (FCOE、FCoE、またはfcoe)、ネットワークファイルシステム (NFSまたはnfs)、nfsオーバーrdma、Andrewファイルシステム (AFSまたはafs)、コモンインターネットファイルシステム (CIFSまたはcifs)、windows共有) についてコントローラのシステム状態220にクエリしてよい (ステップ205.22)。計算リソースが仮想である場合、ハイパーバイザ (例えば、ハイパーバイザデーモンを介した) は、ストレージリソースへの接続を扱ってよい (ステップ205.23)。これは、仮想マシンがSAN280を認識していないことがあるので、望ましいセキュリティの利点を有し得る。

【0099】

ステップ205.24を参照して、計算リソース及びストレージリソースを接続するプロセスは、システムルール210において指示されてよい。コントローラロジックは次に、リソースが利用可能であり、システム状態220にクエリして、必要な場合書き込み可能であることを確認する (ステップ205.22)。システム状態220は、SQLクエリ (または、他のタイプのデータベースクエリ)、JSON解析などの複数の技術のうちのいずれかを介してクエリすることができる。クエリは、計算リソースがストレージリソースに接続するために必要な情報を返す。コントローラ200、システム状態220、またはシステムルール210は、計算リソースがシステム状態に接続するための認証信用情報を提供してよい (ステップ205.25)。計算リソースは次に、直接またはコントローラを介してのいずれかにより、システム状態220を更新する (ステップ205.26)。

【0100】

図2Hは、物理、仮想、または他のタイプの計算リソース、アプリケーション、サービス、またはホストがストレージリソースを電源投入し、ストレージリソースに接続するための例示的なブートプロセスを示す。ストレージリソースは任意選択で、融合ファイルシステム及び／または拡張可能ボリュームを利用してよい。コントローラまたは他のシステムが物理ホストを有効にする状況では、物理ホストは、システムを構成するためのオペレーティングシステムによりプリロードされてよい。したがって、ステップ205.31で、コントローラは、initramfsによりブートディスクをプリロードしてよい。また、コントローラ200は、アウトオブバンド管理接続260を使用して、予備のオペレーティングシステムをネットワークブートし (ステップ205.30)、次に、任意選択で、予備のオペレーティングシステムによりホストをプリロードしてよい (ステップ205.31)。initramfsは次に、ステップ205.32においてロードし、ストレージリソースは、ステップ205.33で、図2Gに示す方法を使用して接続される。次に、拡張可能ボリュームが存在する場合、一緒に結合されるサブボリュームまたはデバイスは、ステップ205.34で、論理ボリューム管理 (LVM) が使用中である場合にボリュームグループとして任意選択で組み立てられる。または、それらは、ステップ205.34で、ディスクを組み合わせた他の方法を使用して結合されてよい。

【0101】

融合ファイルシステムが使用されている場合、ステップ205.36で、ファイルが組み合わされてよく、次に、ブートプロセスが続いてよい (ステップ205.46)。いくつかの既知の問題点を修正するためにlinuxにおいてoverlayfsが使用される場合、以下のサブプロセスが実行されてよい。揮発性であり得る各マウントされたファイルシステムblobにおいて／dataディレクトリが作成されてよい (ステップ205.37)。次に、ステップ205.38において、new_rootディレクトリが作成されてよく、ステップ205.39で、overlayfsがディレクトリにマウント

される。次に、`initramfs`は、`/new_root`上で`exec_root`を実行する（ステップ205.40）。

【0102】

ホストが仮想マシン（VM）である場合、直接カーネルブートなどの追加のツールが利用可能であってよい。この状況では、ハイパーバイザは、VMをブートする前にストレージリソースに接続してよい（ステップ205.41）、または、ブートしている間にこれを行ってもよい。VMは次に、`initramfs`をロードすることと共にブートされた直接カーネルであってよい（ステップ205.42）。`initramfs`は次に、ステップ205.43においてロードし、ハイパーバイザは、この時点で、リモートであり得るストレージリソースに接続してよい（ステップ205.44）。これが達成されるために、ハイパーバイザホストは、インタフェースにパスイン（pass in）することが必要な場合がある（例えば、`inifiniband`がiSERターゲットに接続するために必要な場合、それは、`pci-passhtr`を使用した仮想機能に基づいたSR-IOVにパスイン（pass in）してよく、またはいくつかの状況では、準仮想化されたネットワークインタフェースを使用してよい）。それらの接続は、`initramfs`によって使用可能である。仮想マシンは次に、まだ接続されていない場合、ステップ205.45においてストレージリソースに接続してよい。仮想マシンはまた、ハイパーバイザを通して（任意選択で、準仮想化されたストレージを通して）そのストレージリソースを受信してよい。プロセスは、任意選択で、融合ファイルシステム及びLVMスタイルディスクをマウントしている仮想マシンに対して同様であってよい。

【0103】

図20は、205.13にあるように、ファイルシステムblobまたはファイルの他のグループからストレージリソースを構成する例示的なプロセスフローを示す。blobは、ステップ205.75において収集され、（ストレージリソースホストがファイルシステムblob232を保持するデバイスとは異なる場合）205.73においてストレージリソースホストに直接コピーされてよい。ストレージリソースが適切な位置にあると、システム状態は次に、ストレージリソースの位置及び利用可能な伝送（例えば、iSER、nvmeof、iSCSI、FCoE、ファイバチャネル、nfs、nfs over rdma）により205.74において更新される。それらのblobのいくつかは、読み出し専用であってよく、そのケースでは、システム状態が同一のままであり、新たな計算リソースまたはホストは、その読み出し専用ストレージリソースに接続してよい（例えば、ベースイメージに接続するとき）。いくつかのケースでは、205.70に示すように、任意の融合ファイルシステムのオーバーヘッドを回避するために、ファイルを単一のファイルシステムイメージに置くことが望ましいことがある。これは、blobを融合ファイルシステムとしてマウントし（ステップ205.71）、次に、それらを新たなファイルシステムにコピーし、またはそれらを単一のファイルシステムとして再パッケージ化し（ステップ205.72）、次に、任意選択で、新たなファイルシステムイメージがストレージリソースとして提示されるための適切な場所に新たなファイルシステムイメージをコピーすることによって達成されてよい。いくつかの融合ファイルシステムは、ステップ205.71において最初にマウントすることなく、マージを行うことを可能にし、融合ファイルシステムを単一のステップでマージすることを可能にし得る。

【0104】

図2Iは、図2Eに示すような別の例示的なテンプレート230を示す。この例では、コントローラは、中間構成ツールを用いて図2Iに示すようなテンプレート230を使用するように構成されてよい。例示的な実施形態によると、中間構成ツールは、新たなアプリケーションまたはサービスを依存されるアプリケーションまたはサービスと結合するために使用される共通APIを含み得る。したがって、テンプレート230は追加で、テンプレートのサービスのセットアップに必要となり得る依存関係リスト244を含み得る。テンプレート230はまた、依存関係の共通APIへの呼び出しを含み得る接続ルール245を含んでよい。テンプレート230はまた、1つまたは複数の共通API243と、

共通API及びバージョンのリスト242とを含み得る。共通API243は、依存されるアプリケーションまたはサービスをコントローラが構成することを可能にする、アプリケーションまたはコントローラから呼び出し可能（または呼び出し不可）であってよい、メソッド、関数、スクリプト、または命令を有してよく、それによって、依存されるアプリケーションまたはサービスは、テンプレート230によって構築される新たなアプリケーションに結合されてよい。コントローラは、共通API243と通信してよい、及び／または新たなサービスまたはアプリケーションと被依存サービスまたはアプリケーションとの結合を構成するようにAPI呼び出しを行ってよい。あるいは、命令は、アプリケーションまたはサービスが、依存されるアプリケーションまたはサービス上の共通API243と直接通信し、及び／または呼び出しを送信することを可能にし得る。テンプレート230の接続ルール245は、新たなサービスまたはアプリケーションを被依存サービスまたはアプリケーションと接続することに関するAPI呼び出しを含み得るルール及び／または命令のセットである。

10

【0105】

システム状態220はさらに、実行中のサービスのリスト246を含み得る。実行中のサービスのリスト246は、テンプレート230からの依存関係244を満たすことを求めるよう、コントローラロジック205によってクエリされてよい。コントローラは、特定のサービス／アプリケーションまたはサービス／アプリケーションのタイプに対して利用可能な様々な共通APIのリスト247も含んでよく、また、共通APIを含むテンプレートも含み得る。リストは、コントローラロジック205、システムルール210、システム状態220、またはコントローラがアクセスすることができるテンプレートストレージに存在してよい。コントローラはまた、全ての既存のまたはロードされたテンプレートからコンパイルされた共通APIのインデックス248を維持する。

20

【0106】

図2Jは、図2Fによって示されたような、コントローラロジック205がテンプレート230を処理することに関するが、ステップ255では、コントローラがサービス依存関係を管理する例示的なプロセスフローを示す。図2Kは、図2Jのステップ255の例示的なプロセスフローを示す。ステップ255.1において、コントローラは、テンプレートから依存関係リスト244を収集する。コントローラはまた、テンプレートから共通APIのリスト243を収集する。（A）ステップ255.2において、コントローラは、テンプレートからの共通APIのリスト243を共通APIのインデックス248と比較することによって、及び依存関係を満たすと考えられるアプリケーションまたはサービスのタイプに基づいて、可能な依存されるアプリケーションまたはサービスのリストを絞り込む。ステップ255.3において、コントローラは、システムルール210が依存関係を満たす方式を指定するかどうかを判定する。

30

【0107】

ステップ255.3においてyesである場合、コントローラは、実行中のテンプレートのリストにクエリすることによって、被依存サービスまたはアプリケーションが実行しているかどうかを判定する（ステップ255.4）。ステップ255.4においてnoである場合、コントローラロジックが被依存サービス／アプリケーションのテンプレートを処理することを含み得るサービスアプリケーションが実行される（及び／または、構成され、次に実行される）（ステップ255.5）。被依存サービスまたはアプリケーションが実行中であるとステップ255.4において判定された場合、プロセスフローは、ステップ255.6に進む。ステップ255.6において、コントローラは、テンプレートを使用して、構築される新たなサービスまたはアプリケーションを被依存サービスまたはアプリケーションに結合する。新たなサービスまたはアプリケーションと依存されるアプリケーション／サービスとを結合する際、コントローラは、それが処理しているテンプレートを検討し、接続ルール245を実行する。コントローラは、接続ルール245に基づいて、依存関係244をどのように満たし、及び／またはアプリケーション／サービスをどのように結合するかについてのコマンドを共通API243に送信する。共通API24

40

50

3は、サービスのAPI関数を呼び出すこと、構成を変更すること、スクリプトを実行すること、他のプログラムを呼び出すことを含み得るが、これらに限定されない、新たなサービスまたはアプリケーションと依存されるアプリケーションまたはサービスを接続する、コントローラからの命令を変換する。ステップ255.6に続いて、プロセスフローは、図2Jのステップ205.2に進む。

【0108】

ステップ255.3が、システムルール210が依存関係を満たす方式を指定しないと判定した場合、コントローラは、ステップ255.7において、適切な依存されるアプリケーションまたはサービスが実行中かどうかを確認するよう、システム状態220にクエリする。ステップ255.8において、コントローラは、適切な依存されるアプリケーションまたはサービスが実行中かどうかについてのクエリに基づいて、その判定を行う。ステップ255.8においてnoである場合、コントローラは、アクションの管理者またはユーザに通知する（ステップ255.9）。ステップ255.8においてyesである場合、プロセスフローは、上記で説明したように動作することができるステップ255.6に進む。ユーザは任意選択で、新たなアプリケーションが実行中の依存されるアプリケーションに接続するべきかどうかについてクエリされてよく、その場合、コントローラは、ステップ255.6において、以下のように新たなアプリケーションまたはサービスを依存されるアプリケーションまたはサービスに結合してよく、コントローラは、それが処理しているテンプレート230を検討し、接続ルール245を実行する。コントローラは次に、依存関係244をどのように満たすかに関して、接続ルール245に基づいて、コマンドを共通API243に送信する。共通API243は、新たなサービスまたはアプリケーションと依存されるアプリケーションまたはサービスを接続する、コントローラからの命令を変換する。

【0109】

外部ユーザインタフェースもしくはウェブUI、またはアプリケーションによってユーザは、コントローラアプリケーションまたはロジック205にも組み込まれてもよいAPIアプリケーション120を通してコントローラ200と通信する。

【0110】

コントローラ200は、複数のネットワーク、相互接続、またはそれを通じてコントローラが計算リソース、ストレージリソース、及びネットワークリソースに動作するよう指示することができる他の接続のうちの1つまたは複数によってスタックまたはリソースと通信する。そのような接続は、アウトオブバンド管理接続260、インバンド管理接続270、SAN接続280、及び任意選択のネットワークインバンド管理接続290を含み得る。

【0111】

アウトオブバンド管理は、コントローラ200を通してシステム100のコンポーネントを検出、構成、及び管理するためにコントローラ200によって使用されてよい。アウトオブバンド管理接続260は、プラグインされ、利用可能であるが、オンにされていないリソースをコントローラ200が検出することを可能にし得る。リソースは、プラグインされるとき、ITシステム状態220に追加されてよい。アウトオブバンド管理は、ブートイメージをロードし、システム100に属するリソースを構成及び監視するように構成されてよい。アウトオブバンド管理はまた、オペレーティングシステムの診断のために一時イメージをブートしてよい。アウトオブバンド管理は、BIOS設定を変更するために使用されてよく、また、実行中のオペレーティングシステム上でコマンドを実行するためにコンソールツールを使用してよい。設定はまた、コンソール、キーボード、及びVGA、DVI、もしくはHDMIポートなどのハードウェアリソース上の物理もしくは仮想モニタポートからのビデオ信号の画像認識を使用して、及び／またはアウトオブバンド管理によって提供されるAPI、例えば、Redfishを使用して、コントローラによって変更されてよい。

【0112】

本明細書で使用されるようなアウトオブバンド管理は、オペレーティングシステム及びメインマザーボードから独立したリソースまたはノードに接続することが可能な管理システムを含み得るが、これらに限定されない。アウトオブバンド管理接続 260 は、ネットワーク、または複数のタイプの直接もしくは間接接続もしくは相互接続を含み得る。アウトオブバンド管理接続のタイプの例は、IPMI、Redfish、SSH、telnet、他の管理ツール、キーボードビデオ及びマウス（KVM）、もしくは KVM over IP、シリアルコンソール、または USB を含むが、これらに限定されない。アウトオブバンド管理は、ノードまたはリソースの電源をオンまたはオフすることができ、温度及び他のシステムデータを監視することができ、オペレーティングシステムの制御の及ばない可能性のある BIOS 及び他の低レベルの変更を行うことができ、コンソールに接続して、コマンドを送信することができ、キーボード、マウス、モニタを含むが、これらに限定されない、入力を制御することができる、ネットワークを通して使用し得るツールである。アウトオブバンド管理は、物理リソース内のアウトオブバンド管理回路に結合されてよい。アウトオブバンド管理は、インストールメディアをブートするために使用し得るディスクとしてディスクイメージを接続してよい。

【0113】

管理ネットワークまたはインバンド管理接続 270 は、コントローラが、計算リソース、ストレージリソース、ネットワークリソース、または他のリソースに関する情報を収集し、リソースが実行しているオペレーティングシステムに直接通信することを可能にし得る。ストレージリソース、計算リソース、またはネットワークリソースは、接続 260 及び／または 270 とインタフェースする管理インタフェースを含んでよく、それによって、それらは、コントローラ 200 と通信することができ、リソースに関して何が実行しており及び利用可能であるかをコントローラに通知することができ、コントローラからコマンドを受信することができる。本明細書で使用されるようなインバンド管理ネットワークは、リソースと、リソースのオペレーティングシステムに直接通信することが可能な管理ネットワークを含む。インバンド管理接続の例は、SSH、telnet、他の管理ツール、シリアルコンソール、または USB を含む得るが、これらに限定されない。

【0114】

アウトオブバンド管理がインバンド管理ネットワークから物理的または仮想的に分離されたネットワークとして本明細書で説明されるが、それらは、本明細書でさらに詳細に説明されるように、効率化のために互いに組み合わせられてよく、または互いに協力して動作してよい。また、したがって、アウトオブバンド及びインバンド管理またはそれらの態様は、コントローラの同一のポートを通して通信してもよく、または組み合わせられた相互接続と結合されてもよい。任意選択で、接続 260、270、280、290 のうちの 1 つまたは複数は、そのようなネットワークのその他から分離されてもよく、または組み合わせられてもよく、同一のファブリックを含んでもよく、または含まなくてもよい。

【0115】

さらに、計算リソース、ストレージリソース、及びコントローラは、ストレージネットワーク（SAN）280 に、コントローラ 200 が各リソースをブートするためにストレージネットワークを使用することができる方式において結合されてもよく、または結合されなくてもよい。コントローラ 200 は、ブートイメージまたは他のテンプレートを別個のストレージまたは他のリソースに送信してよく、その結果他のリソースは、ストレージまたは他のリソースからブートすることができる。コントローラは、そのような状況においてどこからブートするかを指示してよい。コントローラは、リソースの電源をオンしてよく、どこからブートするか、及びどのように自身を構成するかをリソースに指示してよい。コントローラ 200 は、どのようにブートするか、どのイメージを使用するか、及びイメージが別のリソース上にある場合にそのイメージがどこに位置するかをリソースに指示する。BIOS のリソースは、事前に構成されてよい。コントローラは、追加または代わりに、アウトオブバンド管理を通して BIOS を、BIOS がストレージエリアネットワークからブートするように、構成してよい。コントローラ 200 はまた、ISO から

オペレーティングシステムをブートし、リソースがデータをローカルディスクにコピーすることを可能にするように構成されてよい。次に、ローカルディスクは続いて、ブートするために使用されてよい。コントローラは、リソースがブートすることができるように、他のコントローラを含む他のリソースを構成してよい。いくつかのリソースは、計算機能、ストレージ機能、またはネットワーキング機能を提供するアプリケーションを含み得る。さらに、コントローラが、ストレージリソースをブートアップし、次に、ストレージリソースに後続のリソースまたはサービスのブートイメージを供給する役目を果たさせることが可能である。ストレージはまた、別の目的のために使用されている異なるネットワークを通して管理されてよい。

【0116】

任意選択で、リソースのうちの1つまたは複数は、ネットワークインバンド管理接続290に結合されてよい。接続290は、インバンド管理接続270に関して説明したような1つまたは複数のタイプのインバンド管理を含み得る。接続290は、インバンド管理ネットワークを通して、ネットワークを利用し、またはネットワークを管理するよう、コントローラをアプリケーションネットワークに接続してよい。

【0117】

図2Lは、リソース、またはリソース上にロードされたアプリケーションもしくはサービスをブートするよう、テンプレート230からリソースに直接または間接的に（別のリソースまたはデータベースを通して）ロードすることができるイメージ250を示す。イメージ250は、リソースタイプ及びハードウェアに対するブートファイル240を含み得る。ブートファイル240は、配備するリソース、アプリケーション、またはサービスに対応するカーネル241を含み得る。ブートファイル240はまた、ブートプロセスを支援するために使用される `initrd` または同様のファイルシステムを含み得る。ブートシステム240は、異なるハードウェアタイプ及びリソースタイプに対して構成された複数のカーネルまたは `initrd` を含み得る。さらに、イメージ250は、ファイルシステム251を含み得る。ファイルシステム251は、ベースイメージ252及び対応するファイルシステム、サービスイメージ253及び対応するファイルシステム、並びに揮発性イメージ254及び対応するファイルシステムを含み得る。ファイルシステム及びロードされたデータは、リソースタイプと、実行中のアプリケーションまたはサービスとに応じて変化し得る。ベースイメージ252は、ベースオペレーティングシステムファイルシステムを含み得る。ベースオペレーティングシステムは、読み出し専用であってよい。ベースイメージ252はまた、実行されているものとは独立したオペレーティングシステムの基本ツールを含み得る。ベースイメージ252は、ベースディレクトリ及びオペレーティングシステムツールを含み得る。サービスファイルシステム253は、リソース、アプリケーション、またはサービスについての構成ファイル及び仕様を含み得る。揮発性ファイルシステム254は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、これは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、`overlayFS` などの技術を使用して1つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0118】

上述したように、コントローラ200は、計算リソース、ストレージリソース、及び／またはネットワーキングリソースなどのリソースをシステム100に追加することができる。図11Aは、ベアメタルノードなどの物理リソースをシステム100に追加する例示的な方法を示す。リソース、すなわち、計算リソース、ストレージリソース、またはネットワーキングリソースは、ネットワーク接続によってコントローラにプラグインされる（1110）。ネットワーク接続は、アウトオブバンド管理接続を含み得る。コントローラは、リソースがアウトオブバンド管理接続を通してプラグインされたことを認識

10

20

30

40

50

する（１１１１）。コントローラは、リソースのタイプ、能力、及び／または属性を含み得るが、これらに限定されない、リソースに関連する情報を認識する（１１１２）。コントローラは、リソース及び／またはリソースに関連する情報をそのシステム状態に追加する（１１１３）。テンプレートから導出されたイメージは、リソース、ストレージリソースなどの別のリソース、またはコントローラを含み得るが、これらに限定されない、システムの物理コンポーネントにロードされる（１１１４）。イメージは、構成ファイルを含み得る１つまたは複数のファイルシステムを含む。そのような構成は、ＢＩＯＳ及びブートパラメータを含み得る。コントローラは、イメージのファイルシステムを使用してブートするよう物理リソースに指示する（１１１５）。追加のリソース、または異なるタイプの複数のベアメタルもしくは物理リソースは、このようにして、テンプレートのイメージまたは少なくともその一部を使用して追加されてよい。

10

【０１１９】

図１１Ｂは、例示的な実施形態のグローバルシステムルール及びテンプレートを使用して、リソースを自動で割り当てる例示的な方法を示す。要求を満たすためにリソース割り当てを必要とする要求がシステムに対して行われる（１１２０）。コントローラは、そのシステム状態データベースに基づいて、そのリソースプールを認識している（１１２１）。コントローラは、テンプレートを使用して、必要なリソースを決定する（１１２２）。コントローラは、リソースを割り当て、情報をシステム状態に記憶する（１１２３）。コントローラは、テンプレートを使用してリソースを配備する（１１２４）。

20

【０１２０】

図１２を参照すると、本明細書で説明されるシステム１００を使用して、アプリケーションまたはサービスを自動で配備する例示的な方法が示されている。ユーザまたはアプリケーションは、サービスを要求する（１２１０）。要求は、ＡＰＩアプリケーションに変換される（１２２０）。ＡＰＩアプリケーションは、要求をコントローラにルーティングする（１２３０）。コントローラは、要求を解釈する（１２４０）。コントローラは、システムの状態及びそのリソースを考慮する（１２５０）。コントローラは、サービス配備のためにそのルール及びテンプレートを使用する（１２６０）。コントローラは、リソースに要求を送信し（１２７０）、テンプレートから導出されたイメージを配備し（１２８０）、ＩＴシステム状態を更新する。

30

【０１２１】

リソースを追加すること、リソースを割り当てること、及びアプリケーションまたはサービスを配備することなどの動作の追加の及びさらなる詳細な例が、以下でさらに詳細に説明される。

【０１２２】

システムへの計算リソースの追加

図３Ａを参照すると、システム１００への計算リソース３１０の追加が示されている。計算リソース３１０が追加されるとき、計算リソース３１０は、コントローラ２００に結合され、電源をオフにされてもよい。計算リソース３１０がイメージによりプリロードされる場合、ネットワーク接続のいずれかが使用されて、リソースと通信し、リソースをブートし、システム状態に情報を追加し得る代替的なステップが続いてよい。計算リソース及びコントローラが同一のノード上にある場合、計算リソースを実行するサービスはオフである。

40

【０１２３】

図３Ａに示すように、計算リソース３１０は、ネットワーク、すなわち、アウトオブバンド管理接続２６０、インバンド管理接続２７０、及び任意選択で、ＳＡＮ２８０によってコントローラに結合される。計算リソース３１０はまた、サービス、アプリケーションユーザ、及び／またはクライアントが相互に通信することができる１つまたは複数のアプリケーションネットワーク３９０に結合される。アウトオブバンド管理接続２６０は、計算リソース３１０がプラグインされるときにオンされる、計算リソース３１０の独立したアウトオブバンド管理デバイス３１５または回路に結合されてよい。デバイス４１５は、

50

デバイスの電源オン／オフ、コンソールへのアタッチ及びコマンドの打ち込み、温度及び他のコンピュータ健全性関連要素の監視、ならびに、BIOS設定及びオペレーティングシステムの範囲外の他の特徴の設定を含むがこれらに限定されない特徴を可能にし得る。コントローラ200は、アウトオブバンド管理ネットワーク260を通して計算リソース310を確認することができる。コントローラ200はまた、インバンド管理またはアウトオブバンド管理を使用して、計算リソースのタイプを識別してよく、その構成を識別してよい。コントローラロジック205は、追加されたハードウェアを探して、アウトオブバンド管理260またはインバンド管理270を調べるように構成される。計算リソース310が検出される場合、コントローラロジック205は、リソースが自動で、またはユーザと対話することによって構成されるかどうかを判定するために、グローバルシステムルール220を使用してよい。リソースが自動で追加される場合、セットアップは、コントローラ200内のグローバルシステムルール210に従う。リソースがユーザによって追加される場合、コントローラ200内のグローバルシステムルール210は、リソースの追加及びユーザが計算リソースにより何を行うことを望んでいるかを確認するためにユーザに問い合わせてよい。コントローラ200は、新しいリソースが承認されたことの確認を、APIアプリケーション（複数可）にクエリしてよく、あるいは、その他の場合には、ユーザまたはスタックを制御する任意のプログラムに要求してもよい。承認プロセスはまた、新しいリソースの正当性を確認するために、暗号を使用して、自動的かつ確実に完了させてよい。コントローラロジック205は、計算リソース310がプラグインされるスイッチまたはネットワークを含むITシステム状態220に計算リソース310を追加する。

10

20

【0124】

計算リソースが物理的である場合、コントローラ200は、アウトオブバンド管理ネットワーク260を通して計算リソースの電源をオンしてよく、計算リソース310は、例えば、SAN280によって、グローバルシステムルール210及びコントローラロジック205を使用して、テンプレート230からロードされたイメージ350からブートしてよい。イメージは、他のネットワーク接続を通して、または別のリソースによって間接的にロードされてよい。ブートされると、インバンド管理接続270を通して受信された計算リソース310に関連する情報も、収集され、ITシステム状態220に追加されてよい。計算リソース310は次に、ストレージリソースプールに追加されてよく、コントローラ200によって管理され、ITシステム状態220において追跡されるリソースになる。

30

【0125】

計算リソースが仮想である場合、コントローラ200は、インバンド管理ネットワーク270を通して、またはアウトオブバンド管理260を通して計算リソースの電源をオンしてよい。計算リソース310は、例えば、SAN280によって、グローバルシステムルール210及びコントローラロジック205を使用して、テンプレート230からロードされたイメージ350からブートしてよい。イメージは、他のネットワーク接続を通して、または別のリソースによって間接的にロードされてよい。ブートされると、インバンド管理接続270を通して受信された計算リソース310に関連する情報も、収集され、ITシステム状態220に追加されてよい。計算リソース310は次に、ストレージリソースプールに追加されてよく、コントローラ200によって管理され、ITシステム状態220において追跡されるリソースになる。

40

【0126】

コントローラ200は、リソースをオフにして電力を節約する、またはリソースをオンにしてアプリケーション性能を改善する、またはITシステムユーザが有し得る他の理由のためなど、ITシステムユーザによって決められた理由によりグローバルシステムルールに従ってリソースを自動でオン及びオフし、ITシステム状態を更新することが可能であってよい。

【0127】

50

図3Bは、計算リソースをブートし、及び／またはアプリケーションをロードするために、テンプレート230から計算リソース310に直接または間接的に（別のリソースまたはデータベースを通して）ロードされるイメージ350を示す。イメージ350は、リソースタイプ及びハードウェアについてのブートファイル340を含み得る。ブートファイル340は、配備されることになるリソース、アプリケーション、またはサービスに対応するカーネル341を含み得る。ブートファイル340はまた、`initrd`、またはブートプロセスを支援するために使用される同様のファイルシステムを含み得る。ブートシステム340は、異なるハードウェアタイプ及びリソースタイプに対して構成された複数のカーネルまたは`initrd`を含み得る。さらに、イメージ350は、ファイルシステム351を含み得る。ファイルシステム351は、ベースイメージ352及び対応するファイルシステムと共に、サービスイメージ353及び対応するファイルシステムと、揮発性イメージ354及び対応するファイルシステムとを含み得る。ファイルシステム及びロードされたデータは、リソースタイプ及び実行中のアプリケーションまたはサービスに応じて変化してよい。ベースイメージ352は、ベースオペレーティングシステムのファイルシステムを含み得る。ベースオペレーティングシステムは、読み出し専用であってよい。ベースイメージ352はまた、実行されているものとは独立したオペレーティングシステムの基本ツールを含み得る。ベースイメージ352は、ベースディレクトリ及びオペレーティングシステムツールを含み得る。サービスファイルシステム353は、リソース、アプリケーション、またはサービスについての構成ファイル及び仕様を含み得る。揮発性ファイルシステム354は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、それは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、`overlayFS`などの技術を使用して1つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0128】

図3Cは、計算リソース310などのリソースをシステム100に追加する例示的なプロセスフローを示す。この例では、対象のリソースは、計算リソース310として説明されるが、図3Cのプロセスフローについての対象のリソースは、ストレージリソース410及び／またはネットワーキングリソース510であってもよいことが理解されるべきである。図3Cの例では、追加されるリソース310は、コントローラ200と同一のノード上にはない。ステップ300.1において、リソース310は、電源オフ状態にあるコントローラ200に結合される。図3Cの例では、リソース310を接続するためにアウトオブバンド管理接続260が使用される。しかしながら、実施者によって要求される場合、他のネットワーク接続が使用されてもよいことが理解されるべきである。ステップ300.2及び300.3において、コントローラロジック205は、システムのアウトオブバンド管理接続を調べ、追加されているリソース310のタイプ及びその構成を認識及び識別するためにアウトオブバンド管理接続260を使用する。例えば、コントローラロジックは、タイプ及び構成情報を取得するための参照として、BIOSまたはリソースについての他の情報（シリアル番号情報など）を確認することができる。

【0129】

ステップ300.4において、コントローラは、特定のリソース310が自動で追加されるべきであるかどうかを判定するために、グローバルシステムルールを使用する。自動で追加されるべきでない場合、コントローラは、その使用が承認されるまで待機する（ステップ300.5）。例えば、ユーザは、特定のリソース310を使用することを望まないこと、またはステップ300.4において使用されることになるまで、それが自動で保留されてもよいことをクエリに回答してよい。ステップ300.4が、リソース310が自動で追加されるべきであると判定する場合、コントローラは、自動セットアップのためにそのルールを使用し（ステップ300.6）、ステップ300.7に進む。

【0130】

ステップ300.7において、コントローラは、システム状態220にリソースを追加するために、リソースと関連付けられたテンプレート230を選択及び使用する。いくつかのケースでは、テンプレート230は、特定のリソースに固有であってよい。しかしながら、いくつかのテンプレート230は、複数のリソースタイプを網羅してもよい。例えば、いくつかのテンプレート230は、ハードウェアに依存しないものであってよい。ステップ300.8において、コントローラは、グローバルシステムルール210に従って、そのアウトオブバンド管理接続260を通してリソース310の電源をオンする。ステップ300.9において、グローバルシステムルール210を使用して、コントローラは、選択されたテンプレート（複数可）からリソースについてのブートイメージを見つけてロードする。リソース310は次に、対象のテンプレート230から導出されたイメージからブートされる（ステップ300.10）。リソース310に関する追加の情報は次に、リソース310がブートされた後に、インバンド管理接続270を通してリソース310から受信されてよい（ステップ300.11）。そのような情報は、例えば、ファームウェアバージョン、ネットワークカード、リソースを接続し得る任意の他のデバイスを含み得る。ステップ300.12において、新たな情報がシステム状態220に追加されてよい。リソース310は次に、リソースプールに追加されると考えられてよく、割り当てに対して準備される（ステップ300.13）。

10

【0131】

図3Cに関して、リソース及びコントローラが同じノード上にある場合、リソースを実行するサービスがそのノードの外にあってよいことは理解されるべきである。そのような場合、コントローラは、例えば、unixソケット、ループバックアダプタ、またはリソースと通信する他のプロセス間通信技術などの、リソースとのプロセス間通信技術を使用してよい。システムルールから、コントローラは、既知のテンプレートを使用してアプリケーションを実行するために、仮想ホスト、またはハイパーバイザもしくはコンテナホストをコントローラからインストールしてよい。次に、リソースアプリケーション情報を、システム状態220に追加することができ、リソースは、割り当ての準備ができていることになる。

20

【0132】

システムへのストレージリソースの追加

30

図4Aは、システム100へのストレージリソース410の追加を示す。例示的な実施形態において、図3Cの例示的なプロセスフローは、システム100にストレージリソース410を追加するために従うことができ、ここで、追加されるストレージリソース410は、コントローラ200と同じノード上にない。また、ストレージリソース410にイメージがプリロードされる場合、ネットワーク接続のいずれかを使用して、ストレージリソース410と通信し、ストレージリソース410をブートし、システム状態220に情報を追加し得る代替のステップに従ってよいことに注目すべきである。

【0133】

ストレージリソース410が追加されるとき、ストレージリソース410はコントローラ200に結合されて、電源オフであってよい。ストレージリソース410は、ネットワーク、つまり、アウトオブバンド管理ネットワーク260、インバンド管理接続270、SAN280、及び任意選択で、接続290を介してコントローラに結合される。ストレージリソース410はまた、サービス、アプリケーションユーザ、及び／またはクライアントが互いに通信することができる1つまたは複数のアプリケーションネットワーク390に結合されてもよく、または、結合されなくてもよい。アプリケーションまたはクライアントは、アプリケーションを介して、リソースのストレージに直接アクセスまたは間接アクセスを有してよく、それによって、アプリケーションまたはクライアントは、SANを通してアクセスされない。アプリケーションネットワークは、それに組み込まれたストレージを有してもよく、または、アクセスされて、ITシステム状態においてストレージリソースと識別されてもよい。アウトオブバンド管理接続260は、ストレージリソース

40

50

410がプラグインされたときにオンになるストレージリソース410の独立したアウトオブバンド管理デバイス415または回路に結合されてよい。デバイス415は、デバイスの電源オン／オフ、コンソールへのアタッチ及びコマンドの打ち込み、温度及び他のコンピュータ健全性関連要素の監視、ならびに、BIOS設定及びオペレーティングシステムの範囲外の他の特徴の設定を含むがこれらに限定されない特徴を可能にし得る。コントローラ200は、アウトオブバンド管理ネットワーク260を通してストレージリソース410を参照してよい。コントローラ200はまた、ストレージリソースのタイプを識別してよく、インバンドまたはアウトオブバンド管理を使用してその構成を識別してよい。コントローラロジック205は、追加されたハードウェアを探して、アウトオブバンド管理260またはインバンド管理270を調べるように構成される。ストレージリソース410が検出された場合、コントローラロジック205は、グローバルシステムルール220を使用して、リソース410が自動的に、またはユーザとの対話によって構成されるべきかを判定してよい。リソース410が自動的に追加される場合、セットアップは、コントローラ200内のグローバルシステムルール210に従う。リソース410がユーザによって追加される場合、コントローラ200内のグローバルシステムルール210は、リソースの追加とユーザがストレージリソースで行いたいことを確認するようにユーザに求めてよい。コントローラ200は、新しいリソースが承認されたことの確認を、APIアプリケーション（複数可）にクエリしてよく、あるいは、その他の場合には、ユーザまたはスタックを制御する任意のプログラムに要求してもよい。承認プロセスはまた、新しいリソースの正当性を確認するために、暗号を使用して、自動的かつ確実に完了させてよい。コントローラロジック205は、ストレージリソース410がプラグインされるスイッチまたはネットワークを含むITシステム状態220にストレージリソース410を追加する。

【0134】

コントローラ200は、アウトオブバンド管理ネットワーク260を通して、ストレージリソース410の電源をオンにしてもよく、ストレージリソース410は、グローバルシステムルール210及びコントローラロジック205を使用して、例えば、SAN280を介して、テンプレート230からロードされたイメージ450からブートする。イメージはまた、他のネットワーク接続を通して、または、別のリソースを介して間接的に、ロードされてよい。ブートされると、ストレージリソース410に関するインバンド管理接続270を通して受信された情報も収集され、ITシステム状態220に追加されてよい。ここで、ストレージリソース410は、ストレージリソースプールに追加され、ストレージリソース410は、コントローラ200によって管理されて、ITシステム状態220において追跡されるリソースになる。

【0135】

ストレージリソースは、ITシステムが独立してまたは同時に、使用またはアクセスし得るストレージリソースプールまたは複数のストレージリソースプールを備えてよい。ストレージリソースが追加されたとき、ストレージリソースは、ストレージプール、複数のストレージプール、ストレージプールの一部、及び／またはストレージプールの複数の部分をITシステム状態に提供し得る。コントローラ及び／またはストレージリソースは、プールの様々なストレージリソース、またはプール内のこのようなリソースのグループを管理してよい。ストレージプールは、複数のストレージリソース上で実行される複数のストレージプールを含み得る。例えば、フラッタディスクまたはアレイをキャッシュするフラッシュストレージディスクもしくはアレイ、または、専用ストレージノード上のプールに結合された専用計算ノード上のストレージプールは帯域幅及び待ち時間を同時に最適化する。

【0136】

図4Bは、ストレージリソースをブートする及び／またはアプリケーションをロードするために、テンプレート230から（別のリソースまたはデータベースから）ストレージリソース410に直接的または間接的にロードされるイメージ450を示す。イメージ4

10

20

30

40

50

50は、リソースタイプ及びハードウェアのためのブートファイル440を備えてよい。ブートファイル440は、配備されるリソース、アプリケーション、またはサービスに対応するカーネル441を備えてよい。ブートファイル440は、ブートプロセスを支援するために使用されるinitrdまたは類似のファイルシステムも備えてよい。ブートシステム440は、異なるハードウェアタイプ及びリソースタイプのために構成される複数のカーネルまたはinitrdを備えてよい。さらに、イメージ450は、ファイルシステム451を備えてよい。ファイルシステム451は、ベースイメージ452及び対応するファイルシステムと共に、サービスイメージ453及び対応するファイルシステムと、揮発性イメージ454及び対応するファイルシステムとを備えてよい。ロードされるファイルシステム及びデータは、リソースタイプ及びアプリケーション、または実行するサービスに応じて変わり得る。ベースイメージ452は、ベースオペレーティングシステムファイルシステムを備えてよい。ベースオペレーティングシステムは、読み取り専用であってよい。ベースイメージ452はまた、実行されているものから独立しているオペレーティングシステムの基本ツールを備えてよい。ベースイメージ452は、ベースディレクトリ及びオペレーティングシステムツールを含み得る。サービスファイルシステム453は、リソース、アプリケーション、またはサービスのための構成ファイル及び仕様を含み得る。揮発性ファイルシステム454は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、それは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、overlayFSなどの技術を使用して1つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0137】

図5Aは、JBODまたは他のタイプの直接接続ストレージによるノードの形態をとってよい、別のストレージリソース、すなわち、直接接続ストレージ510が、システムに対する追加のストレージリソースとしてストレージリソース410に結合された例を示す。JBODは通常、ストレージリソースを提供する、ノードに接続された外部ディスクアレイであり、図5Aでは、JBODが直接接続ストレージ510の例示的な形態として使用されているが、他のタイプの直接接続ストレージが510として採用される可能性があることは理解されるべきである。

【0138】

コントローラ200は、例えば、図5Aに関して記載されるように、ストレージリソース410及びJBOD510をそのシステムに追加してよい。JBOD510は、アウトオブバンド管理接続260を介してコントローラ200に結合される。ストレージリソース410は、ネットワーク、つまり、アウトオブバンド管理接続260、インバンド管理接続270、SAN280、及び任意選択で、接続290に結合される。ストレージノード410は、SASまたは他のディスクドライブファブリック520を通してJBOD510のストレージと通信する。JBOD510は、アウトオブバンド管理接続260を通してコントローラと通信するアウトオブバンド管理デバイス515も備えてよい。アウトオブバンド管理260を通して、コントローラ200は、JBOD510及びストレージリソース410を検出してよい。コントローラ200は、例えば、様々なアウトオブバンド管理回路に関して本明細書に記載されるような、オペレーティングシステムによって制御されない他のパラメータも検出してよい。コントローラ200グローバルシステムルール210は、まだ追加されていないJBOD及びストレージノードのブートまたは起動のための構成起動ルールを提供する。ストレージリソースをオンにする順序は、グローバルルール220を使用して、コントローラロジック205によって制御されてよい。グローバルシステムルール220の1つのセットに従って、コントローラは最初に、JBOD510の電源をオンにしてよく、次に、コントローラ200は、図4に関して記載されたものと類似の方法で、ロードされたイメージ450を使用して、ストレージリソースの電源

をオンにしてよい。グローバルシステムルール別のセットでは、コントローラ 200 は最初に、ストレージリソース 410、次に JBOD 510 をオンにしてよい。他のグローバルシステムルールでは、様々なデバイス間の電源オンのタイミングまたは遅れが指定されてよい。コントローラ ロジック 205、グローバルシステムルール 210、及び／または テンプレート 230 によって、様々なリソースの準備状態または動作状態の検出が判定されてよく、及び／または、コントローラ 200 によるデバイス割り当て管理で使用されてよい。IT システム状態 220 は、ストレージリソース 410 との通信によって更新されてよい。ストレージ ノード 410 は、ディスク ファブリック 520 を通して JBOD にアクセスすることによって、JBOD 510 のストレージ パラメータ及び構成を認識している。ストレージリソース 410 は、次に、利用可能なストレージの量及び他の属性に関する情報で IT システム状態 220 を更新するコントローラ 200 に情報を提供する。ストレージリソース 410 がブートされて、ストレージリソース 410 がシステム 100 のストレージリソース 400 のプールの一部として認識されると、コントローラは IT システム状態 220 を更新する。ストレージ ノードは、コントローラ 200 によって設定された構成を使用して、JBOD ストレージリソースを制御するためのロジックを取り扱う。例えば、コントローラは、RAID 10 または他の構成からプールを作成するために、ストレージ ノードに JBOD を構成するように指示してよい。

【0139】

図 5 B は、ストレージリソース 410 及びストレージリソース 410 のための直接接続ストレージ 510 をシステム 100 に追加するための例示的なプロセスフローを示す。ステップ 500.1 では、直接接続ストレージ 510 が、アウトオブバンド管理接続 260 を介して、電源オフ状態のコントローラ 200 に結合される。ステップ 500.2 では、ストレージリソース 410 が、アウトオブバンド管理接続 260 及びインバンド管理接続 270 を介して、電源オフ状態のコントローラ 200 に結合され、同時に、ストレージリソース 410 が、例えば、ディスクドライブ ファブリックなどの SAS 520 を介して、直接接続ストレージ 510 に結合される。

【0140】

次に、コントローラ ロジック 205 が、ストレージリソース 410 及び直接接続ストレージ 510 を検出するために、アウトオブバンド管理接続 260 を調べてよい（ステップ 500.3）。任意のネットワーク接続を使用することができるが、この例では、アウトオブバンド管理が、追加されているリソースのタイプ（この場合、ストレージリソース 410 及び直接接続ストレージ 510）ならびにそれらの構成を認識及び識別するコントローラ ロジックのために使用されてよい（ステップ 500.4）。

【0141】

ステップ 500.5 では、コントローラ 200 が、リソース 410 及び 510 をシステム状態 220 に追加するために、各タイプのストレージデバイスに対する特定のタイプのストレージのためのテンプレート 230 を選択して使用する。ステップ 500.6 では、コントローラは、電源をオンにするブート順を指定することができるグローバルシステムルール 210 に従って、そのような順序で、アウトオブバンド管理接続 260 を通して、直接ストレージ及びストレージ ノードの電源をオンにする（500.6）。グローバルシステムルール 210 を使用して、コントローラは、そのストレージリソース 410 のために選択されたテンプレート 230 からストレージリソース 410 のためのブートイメージを見つけてロードし、次に、ストレージリソースがイメージからブートされる（ステップ 500.7）。ストレージリソース 410 が、ディスク ファブリック 520 を通して直接接続ストレージ 510 にアクセスすることによって、直接接続ストレージ 510 のストレージ パラメータ及び構成を認識している。次に、ストレージリソース 410 及び／または直接接続ストレージ 510 に関する追加情報が、ストレージリソースへのインバンド管理接続 270 を通して、コントローラに提供されてよい（ステップ 500.8）。ステップ 500.9 では、コントローラが、ステップ 500.8 で得られた情報で、システム状態 220 を更新する。ステップ 500.10 では、コントローラが、直接接続ストレージ 5

10を取り扱うストレージリソース410のための構成、及び直接接続ストレージを構成する方法を設定する。次に、ステップ500.11では、直接接続ストレージ510とともにストレージリソース410を備える新しいリソースは、リソースプールに追加されてよく、システム内に割り当てる準備ができる。

【0142】

例示的な実施形態の別の態様によると、コントローラは、計算またはサービスに関与しなくてもよいスタックにおいて他のデバイスを認識するために、アウトオブバンド管理を使用してよい。例えば、そのようなデバイスは、冷却塔／空調装置、照明、温度、音、警報、電力システム、またはシステムと関連する任意の他のデバイスを含み得るが、それらに限定されない。

【0143】

システムへのネットワーキングリソースの追加

図6Aは、システム100へのネットワーキングリソース610の追加を示す。ある例示的な実施形態において、図3Cの例示的なプロセスフローは、システム100にネットワーキングリソース610を追加するために従うことができ、ここで、追加されるネットワーキングリソース610は、コントローラ200と同じノード上にない。また、ネットワーキングリソース610にイメージがプリロードされている場合、ネットワーク接続のいずれかを使用して、ネットワークリソース610と通信し、ネットワークリソース610をブートし、システム状態220に情報を追加する代替のステップに従ってよいことに注目すべきである。

【0144】

ネットワーキングリソース610が追加されるとき、ネットワーキングリソース610はコントローラ200に結合されて、電源オフであってよい。ネットワーキングリソース610は、接続、つまり、アウトオブバンド管理接続260及び／またはインバンド管理接続270を介してコントローラ200に結合されてよい。ネットワーキングリソース610は任意選択で、SAN280及び／または接続290に接続される。ネットワーキングリソース610はまた、サービス、アプリケーションユーザ、及び／またはクライアントが互いに通信することができる1つまたは複数のアプリケーションネットワーク390に結合されてもよく、または、結合されなくてもよい。アウトオブバンド管理接続260は、ネットワーキングリソース610がプラグインされたときにオンになるネットワーキングリソース610の独立したアウトオブバンド管理デバイス615または回路に結合されてよい。デバイス615は、デバイスの電源オン／オフ、コンソールへのアタッチ及びコマンドの打ち込み、温度及び他のコンピュータ健全性関連要素の監視、ならびに、BIOS設定及びオペレーティングシステムの範囲外の他の特徴の設定を含むがこれらに限定されない特徴を可能にし得る。コントローラ200は、アウトオブバンド管理接続260を通してネットワーキングリソース610を参照してよい。コントローラ200は、ネットワーキングリソース及び／またはネットワークファブリックのタイプを識別してもよく、インバンドまたはアウトオブバンド管理を使用して構成を識別してもよい。コントローラロジック205は、追加されたハードウェアを探して、アウトオブバンド管理260またはインバンド管理270を調べるように構成される。ネットワーキングリソース610が検出された場合、コントローラロジック205は、ネットワーキンググローバルシステムルール220を使用して、リソース610が自動的に、またはユーザとの対話によって構成されるべきかを判定してよい。リソース610が自動的に追加される場合、セットアップは、コントローラ200内のグローバルシステムルール210に従う。ユーザによって追加される場合、コントローラ200内のグローバルシステムルール210は、リソースの追加とユーザがリソースで行いたいこととを確認するようにユーザに求めてよい。コントローラ200は、新しいリソースが承認されたことの確認を、APIアプリケーション（複数可）にクエリしてよく、または、その他の場合には、ユーザまたはスタックを制御する任意のプログラムに要求してよい。承認プロセスはまた、新しいリソースの正当性を確認するために、暗号を使用して、自動的かつ確実に完了させてよい。次に、コントロ

10

20

30

40

50

ーラロジック 205 は、ネットワーキングリソース 610 を IT システム状態 220 に追加してよい。自身をコントローラに識別させることができないスイッチのために、ユーザは、システム状態にそれらを手動で追加してよい。

【0145】

ネットワーキングリソースが物理的なものである場合、コントローラ 200 は、アウトオブバンド管理接続 260 を通して、ネットワーキングリソース 610 の電源をオンにしてもよく、ネットワーキングリソース 610 は、グローバルシステムルール 210 及びコントローラロジック 205 を使用して、例えば、SAN 280 を介して、テンプレート 230 からロードされたイメージ 605 からブートしてよい。イメージはまた、他のネットワーク接続を通して、または、他のリソースを介して間接的に、ロードされてもよい。ブートされると、ネットワーキングリソース 610 に関するインバンド管理接続 270 を通して受信された情報も、収集され、IT システム状態 220 に追加されてよい。ネットワーキングリソース 610 は、ストレージリソースプールに追加され、ネットワーキングリソース 610 は、コントローラ 200 によって管理され、IT システム状態 220 において追跡されるリソースになる。任意選択で、いくつかのネットワーキングリソーススイッチは、アウトオブバンド管理 260 に接続されたコンソールポートを通して制御されてよく、電源オン時に構成されてよく、または、ブートローダを通して、例えば、ONIE を通してインストールされたスイッチオペレーティングシステムを有してもよい。

【0146】

ネットワーキングリソースが仮想である場合、コントローラ 200 は、インバンド管理ネットワーク 270 を通して、またはアウトオブバンド管理 260 を通して、ネットワーキングリソースの電源をオンにしてよい。ネットワーキングリソース 610 は、グローバルシステムルール 210 及びコントローラロジック 205 を使用して、SAN 280 を介して、テンプレート 230 からロードされたイメージ 650 からブートしてよい。ブートされると、ネットワーキングリソース 610 に関するインバンド管理接続 270 を通して受信された情報は、収集され、IT システム状態 220 に追加されてよい。次に、ネットワーキングリソース 610 は、ストレージリソースプールに追加され、ネットワーキングリソース 610 は、コントローラ 200 によって管理され、IT システム状態 220 において追跡されるリソースになる。

【0147】

コントローラ 200 は、異なる物理または仮想リソース、すなわち、本明細書で定義されるような接続、ストレージ、または計算に接続するために、ポートの割り当て、再割り当て、または移動を、物理的または仮想的なネットワーキングリソースに指示してよい。これは、SDN、InfiniBand パーティショニング、VLAN、vXLAN を含むがこれらに限定されない技術を使用して行われてよい。コントローラ 200 は、仮想スイッチまたは仮想スイッチをホストするリソースとのネットワークまたは相互接続通信に仮想インタフェースを移動させるまたは割り当てるように、仮想スイッチに指示してよい。いくつかの物理または仮想スイッチは、コントローラに結合された API によって制御されてよい。

【0148】

そのような変更が可能であるとき、コントローラ 200 はまた、計算リソース、ストレージリソース、またはネットワーキングリソースにファブリックタイプを変更するように指示してよい。ポートは、異なるファブリックに切り替えるように、例えば、ハイブリッド InfiniBand / イーサネットインタフェースのファブリックを切り換えるように構成されてよい。

【0149】

コントローラ 200 は、複数のアプリケーションネットワークを切り替えるスイッチまたは他のネットワーキングリソースを含み得るネットワーキングリソースに命令を与えてよい。スイッチまたはネットワークデバイスは、異なるファブリックを備えてよく、または、例えば、好ましくは、SDN 機能及び複数のファブリックによって、InfiniB

10

20

30

40

50

andスイッチ、ROCEスイッチ、及び／または他のスイッチにプラグインされてよい。
【0150】

図6Bは、ネットワーキングリソースをブートする及び／またはアプリケーションをロードするために、テンプレート230から（例えば、別のリソースまたはデータベースを介して）ネットワーキングリソース610に直接的または間接的にロードされるイメージ650を示す。イメージ650は、リソースタイプ及びハードウェアのためのブートファイル640を備えてよい。ブートファイル640は、配備されるリソース、アプリケーション、またはサービスに対応するカーネル641を備えてよい。ブートファイル640は、ブートプロセスを支援するために使用されるinitrdまたは類似のファイルシステムも備えてよい。ブートシステム640は、異なるハードウェアタイプ及びリソースタイプのために構成される複数のカーネルまたはinitrdを備えてよい。さらに、イメージ650は、ファイルシステム651を備えてよい。ファイルシステム651は、ベースイメージ652及び対応するファイルシステムと共に、サービスイメージ653及び対応するファイルシステムと、揮発性イメージ654及び対応するファイルシステムとを備えてよい。ロードされるファイルシステム及びデータは、リソースタイプ及びアプリケーション、または実行するサービスに応じて変わり得る。ベースイメージ652は、ベースオペレーティングシステムファイルシステムを備えてよい。ベースオペレーティングシステムは、読み取り専用であってよい。ベースイメージ652はまた、実行中のものから独立しているオペレーティングシステムの基本ツールを備えてよい。ベースイメージ652は、ベースディレクトリ及びオペレーティングシステムツールを含み得る。サービスファイルシステム653は、リソース、アプリケーション、またはサービスのための構成ファイル及び仕様を含み得る。揮発性ファイルシステム654は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、これは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、overlayFSなどの技術を使用して1つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0151】

リソース上へのアプリケーションまたはサービスの配備

図7Aは、コントローラ200と、第1の計算ノード311、第2の計算ノード312、及び第3の計算ノード313を備える物理及び仮想計算リソースと、ストレージリソース410と、ネットワークリソース610とを備えるシステム100を示す。リソースは、図1～6Bに関して本明細書に記載されるような方法で、セットアップされて、ITシステム状態220に追加されるとして示されている。

【0152】

複数の計算ノードがこの図に示されているが、ある例示的な実施形態による、単一の計算ノードが使用されてもよい。計算ノードは、物理または仮想計算リソースをホストしてよく、物理または仮想計算ノード上でアプリケーションを実行してよい。同様に、単一のネットワークプロバイダノード及びストレージノードが示されているが、これらのタイプの複数のリソースノードは、例示的な実施形態のシステムで使用されてもよく、または、使用されなくてもよいことが企図されている。

【0153】

サービスまたはアプリケーションは、例示的な実施形態によるシステムのいずれかに配備されてよい。計算ノード上にサービスを配備する例は、図7Aに関して記載され得るが、システム100の異なる配置で同様に使用されてもよい。例えば、図7Aのコントローラ200は、グローバルシステムルール210に従って、計算リソース310を計算ノード311、312、313の形態で自動的に構成してよい。次に、それらは、ITシステム状態220に追加されてよい。したがって、コントローラ200は、（電源オフであってよい、または、電源オフでなくてもよい）計算リソース311、312、313、及び

、場合によっては、計算リソースまたはノード上で実行する任意の物理または仮想アプリケーションを認識してよい。コントローラ200は、グローバルシステムルール210及びテンプレート230に従って、ストレージリソース（複数可）410及びネットワーキングリソース（複数可）610を自動的に構成してよく、それらをITシステム状態220に追加してよい。コントローラ200は、電源オフ状態で開始してもよい、または、開始しなくてもよいストレージリソース410及びネットワーキングリソース610を認識してよい。

【0154】

図7Bは、ITシステム100にリソースを追加するための例示的プロセスを示す。ステップ700.1で、新しい物理リソースがシステムに結合される。ステップ700.2で、コントローラが新しいリソースを認識する。リソースは、リモートストレージに接続されもよい（ステップ700.4）。ステップ700.3で、コントローラが、新しいリソースをブートする方法を構成する。リソースに行われるすべての接続は、システム状態220にログすることができる（ステップ700.5）。上で説明した図3Cは、図7Bに示されるものなどのプロセスフローの例示的な実施形態に関するさらなる詳細を提供する。

【0155】

図7C及び7Dは、複数の計算リソース、複数のサーバ、複数の仮想マシン、及び／または複数のサイトへのアプリケーションの配備のための例示的なプロセスフローを示す。本例のためのプロセスは、ITシステム100が、冗長で互いに関係するアプリケーション及び／またはサービスを結合するコンポーネントを必要とするという事実において、標準的なテンプレート配備と異なる。コントローラロジックは、ステップ700.11でメタテンプレートを処理してよく、ここで、メタテンプレートは、複数のテンプレート230、ファイルシステムblob232、及び、マルチホームサービスを構成するために必要とされる（他のテンプレート230の形態であってよい）他のコンポーネントを含み得る。

【0156】

ステップ700.12で、コントローラロジック205が、利用可能なリソースのシステム状態220をチェックする。しかしながら、十分なリソースがない場合、コントローラロジックは、配備され得る冗長なサービスの数を減らしてよい（冗長なサービスの数が識別される700.16を参照）。ステップ700.13で、コントローラロジック205が、サービスを一緒に接続するために必要とされるネットワーキングリソース及び相互接続を構成する。サービスまたはアプリケーションが複数のサイトにわたって配備される場合、メタテンプレートは、サイト全体でのデータ同期及び相互運用性を可能にするテンプレートから任意選択で構成されるサービスを含み得る（または、コントローラロジック205が、それらを構成してもよい）（700.15参照）。

【0157】

ステップ700.16で、コントローラロジック205が、システムルール、メタテンプレートデータ、及びリソース利用可能性から、冗長なサービスの数を決定してよい（冗長なサービスが複数のホスト上にある場合）。700.17で、他の冗長なサービスとの結合、及びマスタとの結合がある。複数の冗長なホストがある場合、コントローラロジック205またはテンプレート内のロジック（バイナリ234、デーモン232、またはオペレーティングシステムにおける設定を指示する構成ファイルを含み得るファイルシステムblob）が、ネットワークアドレス及びホスト名の競合を防ぎ得る。任意選択で、コントローラロジックは、ネットワークアドレス（700.18参照）を提供し、DNS（700.19）及びシステム状態220（700.18）に各冗長なサービスを登録する。システム状態220は、冗長なサービスを追跡し、ホスト名（例えば、ソフトウェア定義アクセス（SDA）名）、dns名、ネットワークアドレスなどの競合するパラメータを有する冗長なサービスがすでにシステム状態220にあることに気付いた場合、コントローラロジック205は、重複した登録を許可しない。

10

20

30

40

50

【0158】

図7Dが示す構成ルーチンは、メタテンプレートのテンプレート（複数可）を処理する。構成ルーチンは、すべての冗長サービスを処理し、マルチホストまたはクラスタ化サービスを複数のホストに配備し、ホストを結合するサービスを配備する。システムルールからITシステムを配備することができる任意のプロセスが、構成ルーチンを実行することができる。マルチホストサービスの場合、例示的なルーチンは、700.32のようにサービステンプレートを処理し、700.33のようにストレージリソースをプロビジョニングし、700.35のようにホストの電源をオンにし、700.36のようにホスト／計算リソースをストレージリソースと結合（且つ、システム状態220に登録）してよく、（次に、冗長サービスの数だけ繰り返す（700.38））。毎回、システム状態220に登録し（700.20参照）、コントローラロジックを使用して、個々のサービスを追跡して、競合を防ぐ情報を記録する（700.31参照）。

10

【0159】

サービステンプレートのいくつかは、マルチホストサービスを結合し得るサービス及びツールを含んでよい。これらのサービスのいくつかは、依存関係として扱われてよく（700.39）、次に、700.40の結合ルーチンが、サービスの結合、及びシステム状態220への結合の登録に使用されてよい。さらに、サービステンプレートのうちの1つは、マスタテンプレートであってよく、その場合、700.39の依存するサービステンプレートは、スレーブまたは二次サービスであり、700.40の結合ルーチンは、それらを接続する。ルーチンは、メタテンプレートで定義することができる。例えば、冗長なdns構成のために、700.40の結合ルーチンは、マスタdnsへのスレーブdnsの接続、及び、dnssecとのゾーン転送の構成を含み得る。いくつかのサービスは、性能を改善するために、物理ストレージ（700.34参照）を使用してよく、それは、図5Bで開示された予備OSがロードされてよい。結合サービスのためのツールは、テンプレート自体に含まれてよく、サービス間の構成は、マルチノードアプリケーション／サービスにおいてコントローラ及び／または他のホストがアクセスできるAPIによって行われてよい。

20

【0160】

コントローラ200は、ユーザまたはコントローラが、アプリケーションのために使用する適切な計算バックエンドを決定することを可能にし得る。コントローラ200は、ユーザまたはコントローラが、リソースの使用状況を判定することによって、適切な物理または仮想計算リソースにアプリケーションを最適に配置することを可能にし得る。ハイパーバイザまたは他の計算バックエンドが計算ノードに配備されるとき、それらは、インバンド管理接続270を通して、コントローラリソース使用統計を折り返し報告してよい。コントローラが、それ自体のロジック及びグローバルシステムルール、または、ユーザ入力のいずれかから、仮想計算リソース上のアプリケーションを作成すると決めたとき、コントローラは、最適のホスト上のハイパーバイザを自動的に選択し、そのホスト上の仮想計算リソースの電源をオンにしてよい。

30

【0161】

例えば、コントローラ200は、テンプレート（複数可）230を使用して、1つまたは複数の計算リソースにアプリケーションまたはサービスを配備する。そのようなアプリケーションまたはサービスは、例えば、アプリケーションまたはサービスを実行する仮想マシンであってよい。ある例において、図7Aは、複数の計算ノード上への複数の仮想マシン（VM）の配備を示し、図示のコントローラ200は、複数の計算リソース310が計算ノード311、312、313の形態でその計算リソースプールにあることを認識することができる。計算ノードは、例えば、ハイパーバイザが配備されてもよく、または、代わりに、仮想マシンの使用が速度のために好ましくないことがある場合、ベアメタル上にあってもよい。この例において、計算リソース310は、ハイパーバイザアプリケーションがロードされて、計算ノード311上に構成及び配備されたVM（1）321及びVM（2）322を有する。例えば、計算ノード311が、追加のVMのためのリソースを

40

50

有しない場合、または、特定のサービスのために、他のリソースが好ましい場合、コントローラ 200 は、スタック状態 220 に基づいて、利用可能なリソースが計算ノード 311 上に無いこと、または、異なるリソースで新しい VM をセットアップすることが好ましいことを認識し得る。ハイパーバイザが計算リソース 312 上にロードされ、例えば、他の目的のために使用されるベアメタル計算ノードであり得るリソース 313 上にロードされないことも認識され得る。よって、インストールされているサービスまたはアプリケーションテンプレートの要件、及びシステム状態 220 の状態に従って、本例のコントローラは、次に必要とされるリソース VM (3) 323 の配備のために計算ノード 313 を選択してよい。

【0162】

システムの計算リソースは、ストレージノードのためにストレージリソース上でストレージを共有するように構成されてよい。

【0163】

ユーザは、ユーザインタフェース 110 またはアプリケーションを通して、サービスがシステム 100 のためにセットアップされることを要求してよい。サービスは、電子メールサービス、ウェブサービス、ユーザ管理サービス、ネットワークプロバイダ、LDAP、Dev ツール、VOIP、認証ツール、会計を含み得るが、これらに限定されない。

【0164】

API アプリケーション 120 は、ユーザまたはアプリケーションの要求を変換し、メッセージをコントローラ 200 に送信する。コントローラ 200 のサービステンプレートまたはイメージ 230 は、どのリソースがそのサービスに必要なかを識別するために使用される。次に、使用されるリソースが、IT システム状態 220 に従って利用可能性に基づいて識別される。コントローラ 200 は、必要とされる計算サービスのための計算ノード 311、312、または 313 のうちの 1 つまたは複数に対する要求、必要とされるストレージリソースのためのストレージリソース 410 に対する要求、及び、必要とされるネットワークリソースのためのネットワークリソース 610 に対する要求を行う。次に、IT システム状態 220 は更新され、割り当てられるリソースを識別する。次に、サービスは、サービスまたはアプリケーションのためのテンプレート 230 に従って、グローバルシステムルール 210 を使用して割り当てられたリソースにインストールされる。

【0165】

例示的な実施形態によると、複数の計算ノードが、同じサービスまたは異なるサービスのためかどうかにかかわらず使用されてよく、一方、例えば、ストレージサービス及び／またはネットワークプロバイダプールは、計算ノード間で共有されてよい。

【0166】

図 8A を参照すると、コントローラ 200、計算リソース 300、ストレージリソース 400、及びネットワークリソース 600 が、単一のノードなどの、同じまたは共有物理ハードウェア上にあるシステム 100 が示されている。図 1～10 に示され、説明される様々な特徴は、単一のノードに組み込まれてよい。ノードの電源がオンにされると、コントローライメージはノード上にロードされる。計算リソース 300、ストレージリソース 400、及びネットワークリソース 600 は、テンプレート 230 によって、グローバルシステムルール 210 を使用して構成される。コントローラ 200 は、計算リソースとして計算バックエンド 318、319 をロードするように構成されてよく、計算バックエンドは、ノードまたは異なるノード（複数可）上に追加されてもよく、または、追加されなくてもよい。このようなバックエンド 318、319 は、仮想計算リソース、ネットワークリソース、及びストレージリソースを作成するために、仮想化、コンテナ、及びマルチテナントプロセスを含み得るが、これらに限定されない。

【0167】

アプリケーションまたはサービス 725、例えば、ウェブサービス、電子メールサービス、コアネットワークサービス（DHCP、DNS など）、共同ツールは、コントローラ 200 と共有されたノード／デバイス上の仮想リソース上にインストールされてよい。こ

10

20

30

40

50

これらのアプリケーションまたはサービスは、コントローラ 200 から独立している物理リソースまたは仮想リソースの方へ移動されてよい。アプリケーションは、単一のノード上の仮想マシンで実行されてよい。

【0168】

図 8 B は、単一のノードシステムから、（例えば、図 8 A に示されるようなノード 318 及び／または 319 による）複数のノードシステムに拡張するための例示的なプロセスフローを示す。そこで、図 8 A 及び 8 B を参照すると、単一のサーバ上で実行しているコントローラ 200 を有する IT システムを考えることができ、IT システムをマルチノード IT システムとして拡張することが望ましい。よって、拡張前、IT システムは単一ノード状態にある。図 8 A に示すように、コントローラ 200 は、ストレージリソース、計算リソース、ハイパーバイザ、及び／またはコンテナホストを含み得るが、それらに限定されない様々な IT システム管理アプリケーション及び／またはリソースを動かすために、マルチテナント単一ノードシステム上で実行される。

【0169】

ステップ 800. 2 で、新しい物理リソースをアウトオブバンド管理接続 260、インバンド管理接続 270、SAN 280、及び／またはネットワーク 290 を通して接続することによって、新しい物理リソースが、単一のノードシステムに結合される。本例については、この新しい物理リソースを、ハードウェアまたはホストと呼ぶこともできる。コントローラ 200 は、管理ネットワーク上で新しいリソースを検出してよく、次に、デバイスにクエリしてよい。あるいは、新しいデバイスは、自身をコントローラ 200 に知らせるメッセージをブロードキャストしてよい。例えば、新しいデバイスは、MAC アドレス、アウトオブバンド管理、及び／または、予備 OS のブート及びインバンド管理の使用及びそれによるハードウェアタイプの識別によって識別することができる。いずれにしても、ステップ 800. 3 で、新しいデバイスは、そのノードタイプ及びその現在利用可能なハードウェアリソース及びソフトウェアリソースに関する情報をコントローラに提供する。次に、コントローラ 200 は、新しいデバイス及びその能力を認識する。

【0170】

ステップ 800. 4 で、コントローラ 200 を実行するシステムに課されたタスクが、新しいホストに割り当てられてよい。例えば、ホストに、オペレーティングシステム（ストレージホストオペレーティングシステムまたはハイパーバイザなど）がプリロードされる場合、コントローラ 200 は、新しいハードウェアリソース及び／または能力を割り当てる。次に、コントローラは、イメージを提供して、新しいハードウェアをプロビジョニングしてよい、または、新しいハードウェアはコントローラにイメージを要求してよく、上記でかつ以下で開示される方法を使用して自身を構成してよい。新しいホストがストレージリソースまたは仮想計算リソースをホストすることができる場合、新しいリソースは、コントローラ 200 に利用可能とすることができる。次に、コントローラ 200 は、既存のアプリケーションを新しいリソースに移動及び／または割り当ててよい、または、新たに作成されたアプリケーションまたはその後作成されるアプリケーションのために新しいリソースを使用してよい。

【0171】

ステップ 800. 5 で、IT システムは、コントローラ上で実行しているその現在のアプリケーションを保持してよい、または、それらを新しいハードウェアに移行してよい。仮想計算リソースを移行する場合、VM 移行技術（例えば、qemu + kvm の移行ツール）が使用されてよく、新しいシステムルールと共にシステム状態を更新する。以下で説明する変更管理技術を使用して、これらの変更を確実かつ安全に行うことができる。より多くのアプリケーションがシステムに追加され得るので、コントローラは、システムのリソースを割り当てる方法を決定するための様々な技術の任意のものを使用してもよく、それらの技術は、ラウンドロビン技術、重み付きラウンドロビン技術、最少利用技術、重み付き最少利用技術、利用に基づく支援訓練による予測技術、計画技術、所望能力技術、及び、最大サイズ技術を含むがこれらに限定されない。

【0172】

図8Cは、ストレージリソースの新しい物理ストレージリソースへの移行のための例示的なプロセスフローを示す。次に、ストレージリソースは、ミラーリングされてよく、移行されもよく、またはその組合せであってよい（例えば、ストレージはミラーリングされてよく、次に、元のストレージリソースが切断される）。ステップ820で、ストレージリソースは、その新しいストレージリソースがコントローラに連絡する、または、コントローラに新しいストレージリソースを発見させることによって、システムに結合される。これは、アウトオブバンド管理接続260、インバンド管理接続270、SANネットワーク280、または、アプリケーションネットワークが使用してよいフラットネットワーク、または、これらの組合せを使用して行うことができる。インバンド管理によって、オペレーティングシステムはプリブートされてよく、新しいリソースはコントローラに接続してよい。

10

【0173】

ステップ822で、新しいストレージターゲットが、新しいストレージリソース上に作成され、これは、ステップ824でデータベースにログすることができる。ある例において、ストレージターゲットは、ファイルをコピーすることによって作成されてよい。別の例において、ストレージターゲットは、ブロックデバイスを作成して、（ファイルシステムblob（複数可）の形態であってもよい）そのデータをコピーすることによって作成されてよい。別の例において、ブロックデバイス間に2つ以上のストレージリソースをミラーリング（例えば、RAIDを作成）し、任意選択で、iscsi、iser、nvmeof、nfs、nfs over rdma、fc、fc oe、srpなどを含むがこれらに限定されないリモートストレージトランスポート（複数可）を通して接続することによって、ストレージターゲットは作成されてよい。ステップ824で、ストレージリソースが他のリソースまたはホストと同じデバイス上にある場合、データベースエントリは、リモートまたはローカルで新しいストレージリソースに接続するために、計算リソース（または、他のタイプのリソース及び／またはホスト）のための情報を含み得る。

20

【0174】

ステップ826で、ストレージリソースは同期される。例えば、ストレージはミラーリングすることができる。別の例として、ストレージはオフラインで同期することができる。raid1（または、他のタイプのraid、通常、raid1またはraid0、しかし、必要に応じて、raid110であってもよい（ミラーリングされたraid10））などの技術（mdadm、zfs、btrfs、ハードウェアraid）が、ステップ826で採用されてよい。

30

【0175】

次に、古いストレージリソースからのデータは、ステップ828でのデータベースログインの後、任意選択で接続される（それが後で行われるときは、そのようなデータを記録しなければならない場合、データベースはデータをコピーする状態に関連する情報を含み得る）。ストレージターゲットが前のホストから離れて移行中（例えば、以前に示されたように、図8A及び8Bの通り、単一のノードシステムから、マルチノード及び／または分散ITシステムへ移動している）場合、新しいストレージリソースは、ステップ830で、コントローラ、システム状態、計算リソース、またはそれらの組合せによって、一次ストレージリソースと指定されてよい。これは、古いストレージリソースを削除するステップとして行われてもよい。場合によっては、次に、リソースに接続された物理または仮想ホストを更新する必要がある、場合によっては、ステップ832で、移行中、電源をオフにしてよい（次に、電源をオンに戻してよい）（物理または仮想ホストの電源をオンにするための本明細書で開示された技術を使用することができる）。

40

【0176】

図8Dは、マルチテナントシステムの単一ノード上の仮想マシン、コンテナ、及び／またはプロセスを、計算及びストレージのために別々のハードウェアを有し得るマルチノードシステムに移行するための例示的なプロセスフローを示す。ステップ850で、コント

50

ローラ200は、新しいノード上にあってよい新しいストレージリソースを作成する（例えば、図8Aのノード318及び319参照）。次に、ステップ852で、古いアプリケーションホストは電源をオフにされてよい。次に、ステップ854で、データは、コピーまたは同期される。ステップ854でのコピー／同期前に、ステップ852で電源を落とすことによって、移行が単一ノードからのVMの移行を含む場合に、移行はより安全となる。電源オフは、VMから物理マシンへの移動にとっても有益であろう。ステップ854は、データ事前同期ステップ862を介して電源を落とす前に実現されてもよく、それによって、関連するダウンタイムを最小限に抑えることができる。さらに、ホストは、ステップ852のように電源を落とさなくてもよく、その場合、新しいホストの準備ができる（または、新しいストレージリソースの準備ができる）まで、古いホストはオンラインのままである。電源オフステップ852を回避するための技術は、以下でさらに詳細に説明する。ステップ854で、ストレージリソースがホットスタンバイを使用してミラーリングまたは同期されない限り、データは任意選択で同期することができる。

【0177】

ここで、新しいストレージリソースは、稼働中であり、ステップ856でデータベースにログインされてよく、それによって、コントローラ200は、ステップ858で、新しいホストを新しいストレージリソースに接続することができる。複数の仮想ホストによって単一のノードから移行されるとき、このプロセスを複数のホストに対して繰り返すことが必要な場合がある（ステップ860）。それらが追跡される場合、ブートの順番は、アプリケーションの依存関係を使用して、コントローラロジックによって決定されてよい。

【0178】

図8Eは、システムにおける単一ノードから複数ノードへの拡張のための別の例示的なプロセスフローを示す。ステップ870で、新しいリソースが単一のノードシステムに結合される。コントローラは、システムのためのシステムルール及び／または拡張ルールのセットを有してよい（または、コントローラは、サービス実行、サービスのテンプレート、及びサービスの互いの依存関係に基づいて拡張ルールを導出してよい）。ステップ872で、コントローラは、拡張を容易にするために使用されるそのようなルールをチェックする。

【0179】

新しい物理リソースがストレージリソースを含む場合、ストレージリソースは、ステップ874で、より単純なITシステムの単一ノードまたは他の形態から移動されてよい（または、ストレージリソースがミラーリングされてよい）。ストレージリソースが移動される場合、ストレージリソースが移動された後、計算リソースまたは実行中のリソースは、ステップ876でリロードまたはリブートされてよい。別の例において、計算リソースは、ステップ876で、ミラーリングされたストレージリソースに接続されてよく、実行しているままでよい、一方、単一ノードシステム上の古いストレージリソースまたは以前のシステムのハードウェアリソースは、切断または無効にされてよい。例えば、実行中のサービスは、2つのミラーリングされたブロックデバイスに結合されてよい。つまり、一方は単一ノードサーバ上（例えば、mdadm raid1を使用）、及び他方はストレージリソース上に結合されてよい。データが同期されると、単一ノードサーバ上のドライブは切断されてよい。以前のハードウェアは、ITシステムの一部を含んだままでもよく、混合モード（ステップ878）におけるコントローラとして、同じノード上でそれを実行してもよい。元のノードがコントローラを動かすのみになるまで、システムはこの移行プロセスを繰り返し続けてよく、そこで、システムは分散される（ステップ880）。さらに、図8Eのプロセスフローのステップのそれぞれにおいて、コントローラは、システム状態220を更新することができ、データベースにシステムに対する変更をログすることができる（ステップ882）。

【0180】

図9Aを参照すると、アプリケーション910は、リソース900上にインストールされる。リソース900は、本明細書に記載されるような図1～10に関する計算リソース

10

20

30

40

50

310、ストレージリソース410、またはネットワーキングリソース610であってよい。リソース900は物理リソースであってよい。物理リソースは、物理マシンまたは物理ITシステムコンポーネントを含み得る。リソース900は、例えば、物理計算リソース、物理ストレージリソース、または物理ネットワーキングリソースであってよい。リソース900は、本明細書の図2A～10に関して記載されるような計算リソース、ネットワーキングリソース、またはストレージリソースのうちの他のリソースによって、システム100のコントローラ200に結合されてよい。

【0181】

リソース900は、初めは電源が落とされていてよい。リソース900は、ネットワーク、つまり、アウトオブバンド管理接続260、インバンド管理接続270、SAN280、及び／またはネットワーク290を介してコントローラに結合されてよい。リソース900はまた、サービス、アプリケーションユーザ、及び／またはクライアントが互いに通信することができる1つまたは複数のアプリケーションネットワーク390に結合されてよい。アウトオブバンド管理接続260は、リソース900がプラグインされたときにオンになるリソース900の独立したアウトオブバンド管理デバイス915または回路に結合されてよい。デバイスは、デバイスの電源オン／オフ、コンソールへのアタッチ及びコマンドの打ち込み、温度及び他のコンピュータ健全性関連要素の監視、ならびに、BIOS設定195及びオペレーティングシステムの範囲外の他の特徴の設定を含むがこれらに限定されない特徴を可能にしてよい。

【0182】

コントローラ200は、アウトオブバンド管理ネットワーク260を通してリソース900を検出し得る。コントローラ200はまた、リソースのタイプを識別してよく、インバンド管理またはアウトオブバンド管理を使用してその構成を識別してよい。コントローラロジック205は、追加のハードウェアを探して、アウトオブバンド管理260またはインバンド管理270を調べるように構成されてよい。リソース900が検出された場合、コントローラロジック205は、リソース900が自動的に、またはユーザとの対話によって構成されるべきかどうか判定するために、グローバルシステムルール220を使用してよい。リソース900が自動的に追加される場合、セットアップは、コントローラ200内のグローバルシステムルール210に従う。リソース900がユーザによって追加される場合、コントローラ200内のグローバルシステムルール210は、リソースの追加とユーザが計算リソースで行いたいことを確認するようにユーザに求めてよい。コントローラ200は、新しいリソースが承認されたことの確認を、APIアプリケーション（複数可）にクエリしてよく、あるいは、その他の場合には、ユーザまたはスタックを制御する任意のプログラムに要求してもよい。承認プロセスはまた、新しいリソースの正当性を確認するために、暗号を使用して、自動的にかつ確実に完了させてよい。次に、リソース900は、リソース900が接続されるスイッチまたはネットワークを含むITシステム状態220に追加される。

【0183】

コントローラ200は、アウトオブバンド管理ネットワーク260を通して、リソースの電源をオンにしてよい。コントローラ200は、アウトオブバンド管理接続260を使用して、物理リソースの電源をオンにし、BIOS195を構成してよい。コントローラ200は、自動的に、コンソール190を使用してよく、所望のBIOSオプションを選択してもよく、それは、コントローラ200が、画像認識でコンソールイメージを読み取り、アウトオブバンド管理を通してコンソール190を制御することによって実現されてよい。ブートアップ状態は、リソース900のコンソールを通じた画像認識、仮想キーボードによるアウトオブバンド管理、リソースで動作しているサービスにクエリすること、または、アプリケーション910のサービスにクエリすることによって判定されてよい。いくつかのアプリケーションは、コントローラ200が、監視する、または、場合によっては、インバンド管理270を使用してアプリケーション910の設定を変更することを可能にするプロセスを有してよい。

【0184】

物理リソース900上の（または、本明細書の図1～10に関して記載されるようなリソース300、310、311、312、313、400、410、411、412、600、610の）アプリケーション910は、BIOSブートオプション、または、PXEブートもしくはFlexブートを可能にすることなどのリモートブートを構成する他の方法を使用して、SAN280または別のネットワークを介してブートしてよい。さらに、または、あるいは、コントローラ200は、アウトオブバンド管理260及び／またはインバンド管理接続270を使用して、物理リソース900にイメージ950のアプリケーションイメージをブートするように指示してよい。コントローラは、リソース上のブートオプションを構成してよい、または、PXEブートまたはFlexブートなどの既存の使用可能リモートブート方法を使用してよい。コントローラ200は、任意選択でまたは代替的にアウトオブバンド管理260を使用して、ISOイメージからブートしてローカルディスクを構成し、次に、リソースにローカルディスク（複数可）920からのブートを指示してよい。ローカルディスク（複数可）は、ブートファイルをロードされてよい。これは、アウトオブバンド管理260、画像認識、及び仮想キーボードを使用することによって達成されてよい。リソースは、ブートファイル及び／またはブートローダもインストールされてよい。リソース900及びアプリケーションは、グローバルシステムルール210及びコントローラロジック205を使用して、例えば、SAN280を介して、テンプレート230からロードされたイメージ950をからブートしてよい。グローバルシステムルール220は、ブートの順番を指定してよい。例えば、グローバルシステムルール220は、最初にリソース900をブートし、次にアプリケーション910をブートすることを要求してよい。イメージ950を使用してリソース900がブートされると、リソース900に関するインバンド管理接続270を通して受信された情報が収集され、ITシステム状態220に追加されてよい。リソース900は、ストレージリソースプールに追加され、リソース900は、コントローラ200によって管理され、ITシステム状態220において追跡されるリソースになる。アプリケーション910は、イメージ950またはリソース900上にロードされたアプリケーションイメージ956を使用して、グローバルシステムルール220で指定された順番でブートされてよい。

【0185】

コントローラ200は、アウトオブバンド管理接続260または別の接続によって、アプリケーション910をアプリケーションネットワーク390に接続するネットワークングリソース610を構成してよい。物理リソース900は、ISER（ISCSI over RDMA）、NVME OF FCOE、FC、もしくはISCSIを含むがこれらに限定されないブロックストレージリソースなどのリモートストレージ、または、SWIFT、GLUSTER、もしくはCEPHFSなどの別のストレージバックエンドに接続されてよい。サービスまたはアプリケーションが立ち上げられ、実行中であるとき、ITシステム状態220は、アウトオブバンド管理接続260及び／またはインバンド管理接続270を使用して更新されてよい。コントローラ200は、アウトオブバンド管理接続260またはインバンド管理接続270を使用して、物理リソース900の電源状態、すなわち、オンかオフかを判定してよい。コントローラ200は、アウトオブバンド管理接続260またはインバンド管理接続270を使用して、サービスまたはアプリケーションが実行中またはブートアップ状態であるかどうかを判定してよい。コントローラは、それが受信する情報及びグローバルシステムルール210に基づいて、他のアクションを行ってよい。

【0186】

図9Bは、アプリケーション910をブートするために、テンプレート230から計算ノードに（例えば、別のリソースまたはデータベースを介して）直接的または間接的にロードされるイメージ950を示す。イメージ950は、アプリケーション910のためのカスタムカーネル941を備えてよい。

【0187】

イメージ 950 は、リソースタイプ及びハードウェアのためのブートファイル 940 を備えてよい。ブートファイル 940 は、配備されるリソース、アプリケーション、またはサービスに対応するカーネル 941 を備えてよい。ブートファイル 940 は、ブートプロセスを支援するために使用される `initrd` または類似のファイルシステムも備えてよい。ブートシステム 940 は、異なるハードウェアタイプ及びリソースタイプのために構成される複数のカーネルまたは `initrd` を備えてよい。さらに、イメージ 450 は、ファイルシステム 951 を備えてよい。ファイルシステム 951 は、ベースイメージ 952 及び対応するファイルシステムと共に、サービスイメージ 953 及び対応するファイルシステムと、揮発性イメージ 954 及び対応するファイルシステムとを備えてよい。ロードされるファイルシステム及びデータは、リソースタイプ及びアプリケーション、または実行するサービスに応じて変わり得る。ベースイメージ 952 は、ベースオペレーティングシステムファイルシステムを備えてよい。ベースオペレーティングシステムは、読み取り専用であってよい。ベースイメージ 952 はまた、実行されているものから独立しているオペレーティングシステムの基本ツールを備えてよい。ベースイメージ 952 は、ベースディレクトリ及びオペレーティングシステムツールを含み得る。サービスファイルシステム 953 は、リソース、アプリケーション、またはサービスのための構成ファイル及び仕様を含み得る。揮発性ファイルシステム 954 は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、それは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、`overlayFS` などの技術を使用して 1 つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0188】

図 9C は、テンプレート 230 の 1 つのタイプとすることができる NT パッケージからアプリケーションをインストールする例を示す。ステップ 900.1 で、コントローラは、パッケージ `blob` がインストールされる必要があることを判定する。ステップ 900.2 で、コントローラは、`blob` タイプ（ブロック、ファイル、ファイルシステム）に対して、デフォルトデータストア上にストレージリソースを作成する。ステップ 900.3 で、コントローラは、ストレージリソースタイプに対して利用可能なストレージトランスポートを介してストレージリソースに接続する。ステップ 900.4 で、コントローラは、接続されたストレージリソースにパッケージ `blob` をコピーする。次に、コントローラは、ストレージリソースから切断し（ステップ 900.5）、ストレージリソースを読み取り専用に設定する（ステップ 900.6）。次に、パッケージ `blob` は正常にインストールされる（ステップ 900.7）。

【0189】

別の例において、本明細書に添付の付属書類 B は、システムが計算リソースを `overlayfs` に接続する方法に関する例示的な詳細を記載する。このような技術を使用して、図 9A に従ってリソース上にアプリケーションをインストールすること、または、図 2F のステップ 205.11 に従ってストレージリソースから計算リソースをブートすることを容易にすることができる。

【0190】

図 9D は、リソース 900 上に配備されるアプリケーション 910 を示す。リソース 900 は、仮想計算リソースを備え得る、例えば、ハイパーバイザ 920、1 つまたは複数の仮想マシン 921、922、及び／またはコンテナを備え得る計算ノードを備えてよい。リソース 900 は、リソース 900 上にロードされたイメージ 950 を使用して、図 1～図 10 に関して本明細書に記載されるのと類似の方法で構成されてよい。この例において、リソース 920 は、仮想マシン 921、922 を管理するハイパーバイザとして示される。コントローラ 200 は、インバンド管理 270 を使用して、リソースを作成するハイパーバイザ 920 をホストするリソース 900 と通信してよく、リソースを構成して適

切なハードウェアリソースを割り当ててよい。適切なハードウェアリソースには、CPU、RAM、GPU、（別のホストにリモート接続するためにRDMAを使用してよい）リモートGPU、ネットワーク接続、ネットワークファブリック接続、及び／または、パーティショニング及び／またはセグメント化されたネットワークへの仮想及び物理接続を含むがこれらに限定されない。コントローラ200は、仮想コンソール190（例えば、SPICEまたはVNCを含むがこれらに限定されない）及び画像認識を使用して、リソース900及びハイパーバイザ920を制御してよい。追加的にまたは代替的に、コントローラ200は、アウトオブバンド管理260またはインバンド管理接続270を使用して、ハイパーバイザ920に、グローバルシステムルール210を使用してテンプレート230からアプリケーションイメージ950をブートするように指示してよい。イメージ950は、コントローラ200上に記憶されてよい、または、コントローラ200は、それらをストレージリソース410に移動またはコピーしてよい。VM921、922のためのブートイメージは、例えば、イメージ950、またはブロックデバイスもしくはリモートホストのファイルとしてローカルに記憶されて、例えば、qcow2またはrawなどのイメージタイプを使用するNFS over RDMA／NFSなどのファイル共有によって共有されてよい、または、ISCSI、ISER、NVME OF、FC、FCOEを使用するリモートブロックデバイスを使用してもよい。イメージ950の一部は、ストレージリソース410または計算ノード310上に記憶されてよい。コントローラ200は、グローバルルール及び／またはテンプレートを使用して、アウトオブバンド管理接続260または別の接続によって、アプリケーションを適切にサポートするネットワーキングリソース610を構成してよい。リソース900の上のアプリケーション910は、BIOSブートオプションを使用してSAN280もしくは別のネットワークによってロードされたイメージ950を使用すること、または、リソース900上のハイパーバイザ920が、ISER（ISCSI over RDMA）、NVME OF FCOE、FC、もしくはISCSIを含むがこれらに限定されないブロックストレージリソース、もしくは、SWIFT、GLUSTER、もしくはCEPHFSなどの別のストレージバックエンドに接続することを可能にすることを介してブートしてよい。ストレージリソースは、ストレージリソース上のテンプレートターゲットからコピーされてよい。ITシステム状態220は、情報を求めてハイパーバイザ920にクエリすることによって更新されてよい。インバンド管理接続270は、ハイパーバイザ920と通信してよく、リソースの電源状態、すなわち、オンかオフかを判定するために、または、ブートアップ状態を判定するために、使用されてよい。ハイパーバイザ920は、仮想化されたアプリケーション910への仮想インバンド接続923を使用してもよく、アウトオブバンド管理に類似した機能のためにハイパーバイザ920を使用してもよい。この情報は、電源が入れているまたはブートされているかどうかによって、サービスまたはアプリケーションが立ち上がって実行中であるかどうかを示してよい。

【0191】

ブートアップ状態は、リソース900のコンソール190を通した画像認識、仮想キーボードによるアウトオブバンド管理260、リソースで動作しているサービスにクエリすること、または、アプリケーション910自体のサービスにクエリすることによって判定されてもよい。いくつかのアプリケーションは、コントローラ200が、監視すること、または、場合によっては、インバンド管理270を使用してアプリケーション910の設定を変更することを可能にするプロセスを有してよい。いくつかのアプリケーションは仮想リソース上にあってよく、コントローラ200は、インバンド管理270（または、アウトオブバンド管理260）を使用して、ハイパーバイザ920と通信することによって監視してよい。アプリケーション910は、入力の監視及び／または追加のためのこのようなプロセス（または、リソースを保存するために切り換えられてもよいこのようなプロセス）を持たない場合がある。そのような場合、コントローラ200は、アウトオブバンド管理接続260を使用してよく、変更及び／または管理プロセス上での切り換えを行うために、画像処理及び／またはシステムにログオンする仮想キーボードを使用してよい。

仮想計算リソースと同様に、仮想マシンコンソール 190 が使用されてよい。

【0192】

図 9 E は、仮想計算リソースホストを IT システム 100 に追加するための例示的なプロセスフローを示す。ステップ 900.11 で、仮想計算リソースとして使用可能であるホストが、システムに追加される。コントローラは、図 15 B のプロセスフローに従ってベアメタルサーバを構成してよい（ステップ 900.12）、または、オペレーティングシステムがプリロードされてよく、及び／または、ホストは事前に構成されてよい（ステップ 900.13）。次に、リソースは、仮想計算リソースプールとしてシステム状態 220 に追加され（ステップ 900.14）、リソースは、コントローラ 200 から API によってアクセスできるようになる（ステップ 900.15）。API は通常、インバンド管理接続 270 を通してアクセスされる。しかしながら、インバンド管理接続 270 は、仮想キーボードで選択的に有効及び／または無効にされてよい。そして、コントローラは、アウトオブバンド接続 260 を通して通信するために、アウトオブバンド管理接続 260 と仮想キーボード及びモニタとを使用してよい（ステップ 900.16）。ここで、ステップ 900.17 で、コントローラは、仮想計算リソースとして新しいリソースを利用できるようになる。

【0193】

例示的なマルチコントローラシステム

図 10 を参照すると、複数の物理計算ノード 311、312、313 を備える、本明細書で図 1～10 に関して記載される計算リソース 300、310 と、複数のストレージノード 411、412 及び JBOD 413 の形態で本明細書に記載されるストレージリソース 400、410 と、コンポーネント 205、210、220、230（図 1～9 C）を含み、本明細書に記載されるコントローラ 200 として構成される複数のコントローラ 200a、200b と、複数のファブリック 611、612、613 を含み、本明細書に記載されるネットワーキングリソース 600、610 と、アプリケーションネットワーク 390 とを有するシステム 100 が示されている。

【0194】

図 10 は、例示的な実施形態のシステム 100 のコンポーネントの可能な配置を示しているが、システム 100 のコンポーネントの可能な配置を限定するものではない。

【0195】

ユーザインタフェースまたはアプリケーション 110 は、API アプリケーション 120 と通信し、API アプリケーション 120 は、コントローラ 200a または 200b のいずれかまたは両方と通信する。コントローラ 200a、200b は、アウトオブバンド管理接続 260、インバンド管理接続 270、SAN 280 またはネットワークインバンド管理接続 290 に結合されてよい。本明細書の図 1～図 9 C を参照して説明するように、コントローラ 200a、200b は、計算ノード 311、312、313、JBOD 413 を含むストレージ 411、412、及びネットワーキングリソース 610 に、接続 260、270、280 及び任意選択で 290 を介して結合される。アプリケーションネットワーク 390 は、計算ノード 311、312、313、ストレージリソース 411、412、413 及びネットワーキングリソース 610 に結合される。

【0196】

コントローラ 200a、200b は、並列で動作してよい。コントローラ 200a または 200b のいずれかは、本明細書の図 1 から図 9 C に関して説明したように、最初はマスタコントローラ 200 として動作してよい。コントローラ（複数可）200a、200b は、システム 100 全体を電源オフ状態から構成するようにされてよい。コントローラ 200a、200b のうちの 1 つはさらに、アウトオブバンド接続 260 及びインバンド接続 270 のいずれかを通して他のコントローラを探索することによって、既存の構成からシステム状態 220 をポピュレートすることができる。コントローラ 200a、200b のいずれかが、1 つまたは複数の接続 260、270 を通してリソースまたは他のコントローラからリソース状態及び関連情報にアクセスするか、またはこれらを受信してよい

。コントローラまたは他のリソースは他のコントローラを更新してよい。したがって、追加のコントローラがシステムに追加されると、このコントローラはシステム 100 をシステム状態 220 に戻すように構成されてよい。コントローラのうちの 1 つまたはマスタコントローラに障害が発生した場合、他のコントローラをマスタコントローラとして指定してよい。IT システム状態 220 はさらに、利用可能またはリソース上に記憶された状態情報から再構築可能であってよい。例えば、アプリケーションは、システム状態が記憶または複製される仮想計算リソースを作成するようにアプリケーションが構成されている計算リソース上に配備されてよい。グローバルシステムルール 210、システム状態 220 及びテンプレート 230 はまた、リソースまたはリソースの組み合わせに保存またはコピーされてよい。したがって、全てのコントローラがオフラインになり、新しいコントローラが追加された場合、新しいコントローラがシステム状態 220 を回復できるようにシステムを構成してよい。

10

【0197】

ネットワークングリソース 610 は、複数のネットワークファブリックを備えてよい。例えば、図 10 に示すように、複数のネットワークファブリックには、SDN イーサネットスイッチ 611、ROCE スwitch 612、InfiniBand スwitch 613 もしくは他のスイッチ、またはファブリック 614 のうち 1 つまたは複数が含まれ得る。計算ノード上に仮想マシンを備えるハイパーバイザは、ファブリックのうちの所望の 1 つまたは複数を利用して、物理スイッチまたは仮想スイッチに接続し得る。ネットワークング配置は、例えば、セキュリティまたは他のリソース最適化のために、例えば、セグメント化されたネットワークングを通して物理ネットワークの制限を許可してよい。

20

【0198】

本明細書の図 1 ～図 10 に説明されているコントローラ 200 を通したシステム 100 は、サービスまたはアプリケーションを自動的にセットアップしてよい。ユーザは、ユーザインタフェース 110 またはアプリケーションを通して、システム 100 のためにサービスをセットアップするよう要求してよい。サービスは、電子メールサービス、Web サービス、ユーザ管理サービス、ネットワークプロバイダ、LDAP、開発者ツール、VOIP、認証ツール、会計ソフトウェアを含んでよいが、これらに限定されない。API アプリケーション 120 は、ユーザまたはアプリケーション要求を変換し、メッセージをコントローラ 200 に送信する。コントローラ 200 のサービステンプレートまたはイメージ 230 は、どのリソースがサービスに必要なかを識別するために使用される。必要なリソースは、システム状態 220 に応じた利用可能性に基づいて識別される。コントローラ 200 は、必要な計算サービスのために、計算リソース 310 または計算ノード 311、312 または 313 に要求を出し、必要なストレージリソースのためにストレージリソース 410 に要求を出し、必要なネットワークングリソースのためにネットワークングリソース 610 に要求を出す。次に、システム状態 220 が更新され、割り当てられることになるリソースが識別される。次に、サービステンプレートに従って、グローバルシステムルール 210 を使用して、割り当てられたリソースにサービスがインストールされる。

30

【0199】

システムセキュリティの向上

40

図 13A を参照すると、リソース 1310 を含み、リソース 1310 はベアメタルまたは物理リソースであり得る IT システム 100 が示されている。図 13A は、システム 100 に接続された単一のリソース 1310 のみを示しているが、システム 100 は複数のリソース 1310 を含み得ることを理解されたい。リソース 1310 (複数可) は、ベアメタルクラウドノードであってよい、またはそれを備え得る。ベアメタルクラウドノードには、物理ホストまたは仮想マシンへのリモートアクセスを可能にし、仮想マシンの作成を可能にし、外部ユーザがリソース (複数可) 上でのコードを実行するのを可能にするように外部ネットワーク 1380 に接続されているリソースが含まれ得るが、これに限定されない。リソース 1310 (複数可) は、外部ネットワーク 1380 またはアプリケーションネットワーク 390 に直接的または間接的に接続されてよい。外部ネットワーク 13

50

80は、コントローラ200またはITシステム100のコントローラによって管理されないインターネットまたは他のリソース（複数可）であってよい。外部ネットワーク1380は、インターネット、インターネット接続（複数可）、コントローラによって管理されないリソース（複数可）、他の広域ネットワーク（例えば、Stratcom、ピアツーピアメッシュネットワークまたは公的にアクセス可能もしくはアクセス不能であり得る他の外部ネットワーク）または他のネットワークを含み得るが、これらに限定されない。

【0200】

物理リソース1310は、ITシステム100aに追加されると、コントローラ200に結合され、電源がオフにされてよい。リソース1310は、アウトオブバンド管理（OOBM）接続260、任意選択でインバンド管理（IBM）接続270、及び任意選択でSAN接続280のうちの、1つまたは複数のネットワークを介してコントローラ200aに結合される。本明細書で使用されるSAN280は、構成SANを備えてもよく、備えなくてもよい。構成SANは、物理リソースの電源オンまたは構成に使用されるSANを含み得る。構成SANは、SAN280の一部であってよい、またはSAN280から分離されていてもよい。インバンド管理はまた、本明細書に示すように、SAN280であってもよく、そうでなくてもよい構成SANを備えてよい。リソースが使用されている場合、構成SANはまた、無効になっているか、切断されているか、または、利用できなくてよい。OOBM接続260はシステム100のOSには見えないが、IBM接続270及び／または構成SANはシステム100のOSに見えてよい。図13Aのコントローラ200は、本明細書の図1～図12Bを参照して説明されるコントローラ200と同様に構成されてよい。リソース1310は、内部ストレージを備え得る。いくつかの構成では、コントローラ200は、ストレージをボビュレートしてよく、データ及び／または情報をフェッチするためにSANに接続するようにリソースを一時的に構成してよい。アウトオブバンド管理接続260は、独立したアウトオブバンド管理デバイス315またはリソース1310がプラグインされたときに電源がオンになるリソース1310の回路に結合されてよい。デバイス315は、デバイスの電源オン／オフ、コンソールへの接続及びコマンドの打ち込み、温度及び他のコンピュータの健全性関連要素の監視、ならびにBIOS設定及びオペレーティングシステムの範囲外の他の特徴の設定を含むが、これらに限定されない特徴を可能にし得る。コントローラ200は、アウトオブバンド管理ネットワーク260を通してリソース1310を参照し得る。コントローラはまた、リソースのタイプを識別してよく、インバンド管理またはアウトオブバンド管理を使用してその構成を識別してよい。以下で説明する図13C～図13Eは、システムセキュリティを強化するように、物理リソース1310をITシステム100aに追加し、及び／またはシステム100を起動または管理するための様々なプロセスフローを示す。

【0201】

ネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースを参照して本明細書で使用される「無効」という用語は、そのようなネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースが（手動または自動により）電源オフにされるか、物理的に切断されるか、及び／または仮想もしくは何らかの他の方法で（例えば、フィルタリングにより）ネットワーク、仮想ネットワーク（例えば、VLAN、VXLAN、InfiniBandパーティションを含むが、これらに限定されない）から切断されるアクションを指す。「無効」という用語には、（送信元からデータを受信または読み取る能力を有したままで）リソースが宛先にデータを送信または書き込みできないようにすること、（宛先にデータを送信または書き込む能力を有したままで）リソースが送信元からデータを受信または読み取りできないようにするなどの、操作性の一方向または単向性の制限も含まれる。そのようなネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースは、追加のネットワーク、仮想ネットワーク、またはリソースの結合から切断され、以前に接続されたネットワーク、仮想ネットワーク、またはリソースの結合に接続されたままであってよい。さらに、そのようなネットワ

ーキングリソースまたはデバイスは、あるネットワーク、仮想ネットワーク、またはリソースの結合から別のものに切り替えることができる。

【0202】

ネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースを参照して本明細書で使用される「有効」という用語は、そのようなネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースが（手動または自動により）電源オンにされるか、物理的に接続されるか、及び／または仮想もしくは何らかの他の方法でネットワーク、仮想ネットワーク（例えば、VLAN、VXLAN、InfiniBandパーティションを含むが、これらに限定されない）に接続するアクションを指す。そのようなネットワーク、ネットワーキングリソース、ネットワークデバイス及び／またはネットワーキングインタフェースは、追加のネットワーク、仮想ネットワーク、または、既に別のシステムコンポーネントに接続されている場合、リソースの結合に接続されてよい。さらに、そのようなネットワーキングリソースまたはデバイスは、あるネットワーク、仮想ネットワークまたはリソースの結合から別のものに切り替えることができる。「有効」という用語には、（送信元からのデータを制限する能力を有しながら）リソースが宛先に対してデータを送信、書き込みまたは受信できるようにすること、（宛先からのデータを制限する能力を有しながら）リソースが送信元からデータを送信、受信または読み取りできるようにするなどの、操作性の一方向または単向性の制限も含まれる。

【0203】

コントローラロジック205は、追加されたハードウェアを探して、アウトオブバンド管理接続260またはインバンド管理接続270及び／または構成SAN280を調べるように構成される。リソース1310が検出された場合、コントローラロジック205は、グローバルシステムルール220を使用して、リソースを自動で構成するか、またはユーザと対話することによって構成するかを判定してよい。自動で追加される場合、セットアップはコントローラ200内のグローバルシステムルール210に従う。ユーザによって追加される場合、コントローラ200内のグローバルシステムルール210は、リソースの追加とユーザがリソース1310で行いたいことを確認するようにユーザに求めてよい。コントローラ200は、新しいリソースが承認されたことの確認を、APIアプリケーション（複数可）にクエリしてよく、あるいは、その他の場合には、ユーザまたはスタックを制御する任意のプログラムに要求してもよい。承認プロセスはまた、新しいリソースの正当性を確認するために、暗号を使用して、自動的かつ確実に完了させてよい。次に、コントローラロジック205は、リソース1310がプラグインされているスイッチまたはネットワークを含むITシステム状態220にリソース1310を追加する。

【0204】

リソースが物理的な場合、コントローラ200はアウトオブバンド管理ネットワーク260を通してリソースの電源をオンにしてよく、リソース1310は、グローバルシステムルール210及びコントローラロジック205を使用して、例えばSAN280を介して、テンプレート230からロードされたイメージ350からブートしてよい。イメージは、他のネットワーク接続を通して、または別のリソースを介して間接的にロードされてよい。一旦ブートされると、リソース1310に関する情報も収集され、ITシステム状態220に追加され得る。これは、インバンド管理接続及び／または構成SAN接続またはアウトオブバンド管理接続を通して行われてよい。リソース1310は、グローバルシステムルール210及びコントローラロジック205を使用して、例えば、SAN280を介して、テンプレート230からロードされたイメージ350からブートしてよい。イメージは、他のネットワーク接続を通して、または別のリソースを介して間接的にロードされてよい。一旦ブートされると、計算リソース310に関してインバンド管理接続270を通して受信された情報も収集され、ITシステム状態220に追加され得る。次に、リソース1310はストレージリソースプールに追加されてよく、これは、コントローラ200によって管理され、ITシステム状態220で追跡されるリソースになる。

【0205】

インバンド管理及び／または構成S A Nは、リソース1310をセットアップ、管理、使用またはリソース1310と通信し、任意のコマンドまたはタスクを実行するために、コントローラ200によって使用されてよい。しかしながら、任意選択で、インバンド管理接続270は、いつでも、またはシステム100もしくはコントローラ200のセットアップ、管理、使用または動作中にオフまたは無効になるようにコントローラ200によって構成されてよい。インバンド管理は、いつでも、またはシステム100もしくはコントローラ200のセットアップ、管理、使用または動作中にオンまたは有効になるようにさらに構成されてよい。任意選択で、コントローラ200は、インバンド管理接続270からコントローラ200（複数可）にリソース1310を制御可能または切り替え可能に切断してよい。そのような切断または切断可能性は物理的であってよく、例えば、自動物理スイッチ、またはネットワークへのリソースのインバンド管理接続及び／または構成S A Nの電源をオフにするスイッチを使用する。例えば、切断は、ネットワークスイッチがリソース1310のインバンド管理270及び／または構成S A N280に接続されたポートへの電源を遮断することによって実行されてよい。このような切断または部分的な切断は、ソフトウェア定義ネットワーキングを使用して行ってよい、またはソフトウェア定義ネットワーキングを使用してコントローラに対して物理的にフィルタリングされてよい。このような切断は、インバンド管理またはアウトオブバンド管理のいずれかを通して、コントローラを介して行ってよい。例示的な実施形態によると、リソース1310がI Tシステムに追加される前、その間、またはその後の任意の時点で、コントローラ200からの選択的な制御命令に応答して、リソース1310をインバンド管理接続270から切断してよい。

【0206】

ソフトウェア定義ネットワーキングを使用して、インバンド管理接続270及び／または構成S A N280は、一部の機能を保持してもよく、または保持しなくてもよい。インバンド管理270及び／または構成S A N280は、コントローラ200または他のリソースとの間の通信のために、制限された接続として使用され得る。接続270は、攻撃者がコントローラ200、他のネットワークまたは他のリソースにピボットするのを防ぐために制限され得る。システムは、コントローラ200及びリソース1310などのデバイスがオープンに通信することを防ぎ、リソース1310を危険にさらすのを防ぐように構成されてよい。例えば、インバンド管理270及び／または構成S A N280において、ソフトウェア定義ネットワーキングまたは（電子的な制限など）ハードウェア変更方法を通して、インバンド管理及び／または構成S A Nのデータ送信のみを可能にし、あらゆる受信を禁止してよい。インバンド管理及び／または構成S A Nは、物理的に、またはコントローラからリソースへの書き込みのみを可能にするソフトウェア定義ネットワーキングを使用してのいずれかで、一方向の書き込みコンポーネントになるように、またはコントローラ200からリソース1310への一方向の書き込み接続として、構成され得る。接続の一方向の書き込みの性質はさらに、セキュリティの望ましい状況及びシステムの動作の様々な段階または時間に従って、制御されるか、またはオンもしくはオフにされてよい。システムはさらに、例えば、リソースからコントローラへの書き込みまたは通信を、例えば、ログまたはアラートの通信に制限するように構成されてよい。インターフェースはさらに、ソフトウェア定義ネットワーキング、V L A N、V X L A N及び／またはI n f i n i B a n dパーティショニングを含むがこれらに限定されない技術によって、他のネットワークへの移動またはネットワークへの追加または削除を行うことができる。例えば、インターフェースをセットアップネットワークに接続し、そのネットワークから削除して、ランタイムに使用されるネットワークに移動することができる。コントローラからリソースへの通信は切断されるか、または制限されることがあり、その結果、コントローラがリソース1310から送信されたあらゆるデータに物理的に応答できなくなる場合がある。一例によれば、リソース1310が追加されてブートされると、インバンド管理270は、物理的に、またはソフトウェア定義ネットワーキングを使用してのいずれかで、スイッ

10

20

30

40

50

チを切るか、またはフィルタリングされてよい。インバンド管理は、ログ管理専用の別のリソースにデータを送信することができるように構成されてよい。

【0207】

インバンド管理は、アウトオブバンド管理またはソフトウェア定義ネットワーキングを使用してオンとオフを切り替えることができる。インバンド管理が切断されていると、デーモンを実行する必要がなくなり、キーボード機能を使用してインバンド管理を再度有効にし得る。

【0208】

さらに、任意選択で、リソース1310はインバンド管理接続を有さなくてよく、リソースはアウトオブバンド管理を通して管理してよい。

【0209】

アウトオブバンド管理は、代替的または追加的に、例えば、キーボード、仮想キーボード、ディスクマウントコンソール、仮想ディスクの接続、BIOS設定の変更、ブートパラメータ及びシステムの他の態様の変更、ブート可能なイメージもしくはインストールCDに存在し得る既存スクリプトの実行、またはリソース1310で実行するオペレーティングシステムの公開の有無に関わらずコントローラ200及びリソース1310を通信可能にするアウトオブバンド管理の他の特徴を含むがこれらに限定されない方法でシステムの様々な態様を操作するために使用されてよい。例えば、コントローラ200は、アウトオブバンド管理260を介して、そのようなツールを使用してコマンドを送信してよい。コントローラ200はさらに、リソース1310の制御を支援するために画像認識を使用してよい。したがって、アウトオブバンド管理接続を使用して、システムは、アウトオブバンド管理接続を介してシステムに接続されるリソースの望ましくない操作を防止または回避し得る。アウトオブバンド管理接続はまた、システムの動作中またはシステムの動作中の選択された時間に、一方向通信システムとして構成されてよい。

【0210】

さらに、アウトオブバンド管理接続260は、実施者が望む場合には、インバンド管理接続と同じ方法で、コントローラ200によって選択的に制御されてよい。

【0211】

コントローラ200は、リソースをオフにして電力を節約する、リソースをオンにしてアプリケーションの性能を向上させる、またはITシステムユーザが有する任意の他の理由など、ITシステムユーザが決定した理由で、グローバルシステムルールに従ってリソースを自動的にオン及びオフし、ITシステムの状態を更新することができる。コントローラはさらにシステム操作中の任意の時に、様々なセキュリティ目的で、構成SAN、インバンド及びアウトオブバンド管理接続をオン及びオフにすること、または、このような接続を一方向の書き込み接続として指定することが可能であってよい（例えば、リソース1310が外部ネットワーク1380または内部ネットワーク390に接続されている間、インバンド管理接続270または構成SAN280を無効にする）。一方向のインバンド管理はさらに、例えば、システムの健全性を監視するために使用されてよく、オペレーティングシステムに見えてよいログ及び情報を監視する。

【0212】

リソース1310はさらに、サービス、アプリケーションユーザ及び／またはクライアントが互いに通信することができるアプリケーションネットワークなどの1つまたは複数の内部ネットワーク390に結合されてよい。このようなアプリケーションネットワーク390はさらに、外部ネットワーク1380に接続されるか、接続可能であってよい。図2A～図12Bを含むがこれらに限定されない、本明細書の例示的な実施形態によれば、インバンド管理は、またはリソースもしくはアプリケーションネットワークが外部ネットワークに接続されている場合、またはリソースが外部ネットワークに接続されていないアプリケーションネットワークに接続されている場合に、追加のセキュリティを提供するために、リソースまたはアプリケーションネットワーク390から切断されるか、切断可能であってよい、またはコントローラからの一方向の書き込みを提供してよい。

【0213】

図13AのITシステム100は、図3Bに示されるITシステム100と同様に構成され得る。イメージ350は、計算リソースのブート及び／またはアプリケーションのロードのために、テンプレート230からリソース1310に直接的または間接的に（別のリソースまたはデータベースを通して）ロードされ得る。イメージ350は、リソースタイプ及びハードウェアのためのブートファイル340を備え得る。ブートファイル340は、配備されるリソース、アプリケーションまたはサービスに対応するカーネル341を備えてよい。ブートファイル340はさらに、ブートプロセスを支援するために使用される `initrd` または同様のファイルシステムを備えてよい。ブートシステム340は、異なるハードウェアタイプ及びリソースタイプのために構成された複数のカーネルまたは `initrd` を備え得る。さらに、イメージ350は、ファイルシステム351を備え得る。ファイルシステム351は、ベースイメージ352及び対応するファイルシステムと共に、サービスイメージ353及び対応するファイルシステムと揮発性イメージ354及び対応するファイルシステムとを備え得る。ロードされるファイルシステム及びデータは、リソースタイプ及び実行されるアプリケーションまたはサービスによって異なる可能性がある。ベースイメージ352は、ベースオペレーティングシステムファイルシステムを含み得る。ベースオペレーティングシステムは読み取り専用であってよい。ベースイメージ352はさらに、実行中のものとは独立したオペレーティングシステムの基本的なツールを備え得る。ベースイメージ352は、ベースディレクトリ及びオペレーティングシステムツールを備え得る。サービスファイルシステム353は、リソース、アプリケーションまたはサービスのための構成ファイル及び仕様を含み得る。揮発性ファイルシステム354は、バイナリアプリケーション、特定アドレス、及び他の情報などの、その配備に固有の情報またはデータを含んでよく、それは、パスワード、セッション鍵、及びプライベート鍵を含むがこれらに限定されない変数として構成されてもよく、または、構成されなくてもよい。ファイルシステムは、`overlayFS` などの技術を使用して1つの単一のファイルシステムとしてマウントされて、いくつかの読み取り専用及びいくつかの読み書きファイルシステムが、アプリケーションのために使用される複製データの量を減少させることを可能にしてよい。

【0214】

図13Bは複数のリソース1310を示し、各リソースは、1つまたは複数の仮想マシンをホストするかまたは備える1つまたは複数のハイパーバイザ1311を備える。コントローラ200aは、それぞれがベアメタルリソースを備えるリソース1310に結合される。図13Bに示し、図13Bを参照して説明されるように、リソース1310はそれぞれコントローラ200aに結合される。本明細書の例示的な実施形態によれば、インバンド管理接続270、構成SAN280及び／またはアウトオブバンド管理接続260は、図13Aに関して説明したように構成されてよい。仮想マシンまたはハイパーバイザのうちの1つまたは複数が危険にさられるか、危険にさられた状態になる可能性がある。従来のシステムでは、そうすると、他のハイパーバイザ上の他の仮想マシンが危険にさられた状態になる可能性がある。例えば、これは仮想マシン内で実行されるハイパーバイザの悪用から生じ得る。例えば、ピボットは、危険にさられたハイパーバイザからコントローラ200aに移動し、そこで危険にさられたコントローラ200aからコントローラ200aに結合された他のハイパーバイザに移動する可能性がある。例えば、危険にさられたハイパーバイザとターゲットのハイパーバイザとの間で、両方に接続されたネットワークを使用してピボットが発生する可能性がある。図13Bに示すコントローラ200a及びリソース1310のインバンド管理270、構成SAN280またはアウトオブバンド管理260の構成では、コントローラ200aとリソース1310との間の所与のリンクでインバンド（または構成SAN）及び／またはアウトオブバンド接続を無効にするために、一部または全てを選択的に制御することができ、これは、危険にさられた仮想マシンが1つのハイパーバイザから抜け出して他のリソースにピボットするために使用されることを防止し得る。

10

20

30

40

50

【0215】

上記で図1から図12に関して説明したように、インバンド管理接続270及びアウトオブバンド管理接続260はさらに、図13A及び13Bに関して説明したのと同様に構成されてよい。

【0216】

図13Cは、ベアメタルノードなどの物理リソースをシステム100に追加または管理するための例示的なプロセスフローを示す。本明細書の図13A及び13Bに示されるか、図1～図12に関して示されるリソース1310は、システム100のコントローラへのアウトオブバンド管理接続260及びインバンド管理接続270及び／またはSANを介して接続されてよい。

10

【0217】

リソース接続のインスタンスの後、ステップ1370で、外部ネットワーク及び／またはアプリケーションネットワークが無効にされる。上述のように、この無効化には、様々な技術のいずれかを使用することができる。例えば、システムのセットアップ、リソースの追加、システムのテスト、システムの更新または他のタスクもしくはコマンドの実行の前に、図13A及び13Bに関して説明したように、インバンド管理接続または構成SANを使用して、システム100のコンポーネント（または攻撃に対して脆弱なもののみ）を外部ネットワークまたはアプリケーションネットワークから無効にするか、切断するか、またはフィルタリングする。

【0218】

ステップ1370の後、次にステップ1371で、インバンド管理接続及び／または構成SANが有効にされる。したがって、ステップ1370と1371との組み合わせによって、インバンド管理及び／またはSAN接続が稼働しているまま、外部ネットワーク及び／またはアプリケーションネットワークからリソースを分離する。次に、インバンド管理接続を介してコントローラ200の制御下でリソースでコマンドを実行することができる（ステップ1372を参照）。例えば、図1～図13Bに関して本明細書で説明されているものなどを含むがこれらに限定されないセットアップ及び構成ステップが、インバンド管理及び／または構成SANを使用してステップ1372で実行されてよい。代替的または追加的に、インバンド管理及び／または構成SANを使用して、ステップ1372で、システムの動作、更新または管理（任意の変更管理またはシステム更新を含み得るがこれらに限定されない）、テスト、更新、データ転送、性能及び健全性に関する情報の収集（エラー、CPU使用率、ネットワーク使用率、ファイルシステム情報及びストレージ使用率を含むがこれらに限定されない）及びログの収集、ならびに本明細書の図1から図13Bに説明されているシステム100を管理するために使用され得る他のコマンドを含むがこれらに限定されない他のタスクが実行されてよい。

20

30

【0219】

図13A及び13Bに関して本明細書で説明されるように、リソースの追加、システムのセットアップ及びそのようなタスクまたはコマンドの実行後、リソースとコントローラもしくはシステムの他のコンポーネントとの間のインバンド管理接続270及び／または構成SAN280は、ステップ1373で、1つまたは複数の方向で無効にされてよい。そのような無効化は、上述のように、切断、フィルタリングなどを採用してよい。ステップ1373の後、ステップ1374で、外部ネットワーク及び／またはアプリケーションネットワークへの接続を復旧してよい。例えば、コントローラは、リソース1310がアプリケーションネットワークまたはインターネットに接続できるようにネットワークングリソースに通知することができる。システムをテストまたは更新する場合も同じステップに従ってよい、つまり、外部ネットワーク及び／またはアプリケーションネットワークへのインバンド管理接続を切断するか、またはフィルタリングしてから、（一方向または両方向で）リソースへのインバンド管理接続は、有効にまたは接続されてよい。したがって、リソースが外部ネットワーク及び／またはアプリケーションネットワークに接続されている間、ステップ1373及び1374は同時に動作して、インバンド管理接続及び／ま

40

50

たは構成 S A N を通したコントローラへの接続からリソースを分離する。

【0220】

アウトオブバンド管理は、システムもしくはリソースの管理、システムもしくはリソースのセットアップ、または、システムもしくはリソースの構成、ブート、もしくは追加に使用されてよい。アウトオブバンド管理は、本明細書の実施形態のいずれかで使用される場合、ブート前に設定を変更するために仮想キーボードを使用してマシンにコマンドを送信してよく、さらに、仮想キーボードに入力することによってオペレーティングシステムにコマンドを送信してよい。マシンがログインしていない場合、アウトオブバンド管理は仮想キーボードを使用してユーザ名及びパスワードを入力してよく、画像認識を使用してログオンを検証し、入力したコマンドを検証し、それらが実行されたかどうかをチェックしてよい。物理リソースにグラフィカルコンソールしかない場合は、仮想マウスも使用されてよく、画像認識によりアウトオブバンド管理は変更を行うことができる。

【0221】

図13Dは、ベアメタルノードなどの物理リソースをシステム100に追加するか、これを管理するための別の例示的なプロセスフローである。ステップ1380で、本明細書の図13A及び13Bまたは図1～図12に示されるリソースは、アウトオブバンド管理260を介してシステムまたはリソースに接続され得る。コントローラによって容易になるアウトオブバンド管理を通してディスクイメージ（例えば、ISOイメージ）へのアクセスを提供することにより、ディスクを仮想的に接続してよい（ステップ1381を参照）。次に、リソースまたはシステムをディスクイメージからブートしてよく（ステップ1382）、次にファイルをディスクイメージからブート可能なディスクにコピーする（ステップ1383を参照）。これは、アウトオブバンド管理を使用してこの方法でリソースがセットアップされているシステムのブートにも使用されてよい。これは、複数のリソースがコントローラをさらに備えるか、システムを構成するかに関わらず、（ネットワーキングリソースを含むがこれに限定されない）一緒に結合可能な複数のリソースを構成及び／またはブートするためにも使用されてよい。したがって、仮想ディスクを使用して、仮想ディスクがリソースに接続されているかのように、コントローラがディスクイメージをリソースに接続することを可能にし得る。アウトオブバンド管理はさらに、リソースにファイルを送信するために使用されてよい。データは、ステップ1383で仮想ディスクからローカルディスクにコピーされてよい。ディスクイメージは、リソースがコピーして操作時に使用することができるファイルを含み得る。ファイルは、スケジュールされたプログラムまたはアウトオブバンド管理からの命令のいずれかを通して、コピーまたは使用されてよい。コントローラは、アウトオブバンド管理を通して、仮想キーボードを使用してリソースにログオンし、コマンドを入力して、ファイルを仮想ディスクからコントローラ自体のディスクまたはリソースにアクセス可能な他のストレージにコピーしてよい。ステップ1384で、システムまたはリソースは、BIOS、EFIまたはブート順序設定を設定することによりブートするように構成され、これによりブート可能ディスクからブートすることになる。ブート構成では、efibootmgrなどのオペレーティングシステムのEFIマネージャを使用してよく、これは、アウトオブバンド管理から直接実行されるか、インストーラスクリプトに含めることで実行されてよい（例えば、リソースのブート時に、efibootmgrを使用するスクリプトが自動的に実行される）。さらに、ブートオプションまたは他のBIOSの変更は、ブート順序コマンドを使用するか、BIOS構成（Supermicro Update ManagerでサポートされているXML BIOS構成など）をアップロードするか、のいずれかを使用するSupermicro Boot Managerなどのアウトオブバンド管理ツールを通して設定されてよい。BIOSはさらに、キーボードとコンソールからの画像認識とを使用して、ブート順序を含む適切なBIOS設定を設定するように構成されてよい。インストーラは、ロードされた予め構成されたイメージに対して実行されてよい。構成は、画面を見て画像認識を使用することによってテストされてよい。構成の後、リソースを有効にする（例えば、電源をオンにする、ブートする、アプリケーションネットワークに接続する、またはそれ

10

20

30

40

50

らの組み合わせ) ことができる (ステップ 1385)。

【0222】

図 13E は、ベアメタルノードなどの物理リソースをシステム 100 に追加するか、または、これを管理するための別の例示的なプロセスフローであり、この場合、PXE、Flexboot または類似するネットワークブートを使用している。ステップ 1390 で、本明細書の図 13A 及び図 13B に示されるか、または図 1～図 12 に関して示されるリソース 1310 は、システム 100 のコントローラへの (1) インバンド管理接続 270 及び／または SAN と、(2) アウトオブバンド管理接続 260 とを介して接続され得る。次に、外部ネットワーク及び／またはアプリケーションネットワーク接続は、ステップ 1391 で (ステップ 1370 に関連して上述したものと同様)、無効にされてよい (例えば、SDN を用いて物理的に、または仮想的に、全体的にまたは部分的に、フィルタリングまたは切断される)。例えば、システムのセットアップ、リソースの追加、システムのテスト、システムの更新またはその他のタスクもしくはコマンドの実行の前に、図 13A 及び図 13B に関して説明したように、インバンド管理接続または SAN を使用して、システム 100 のコンポーネント (または攻撃に対して脆弱なもののみ) を外部ネットワークまたはアプリケーションネットワークから無効にするか、切断するか、またはフィルタリングする。

【0223】

ステップ 1392 で、リソースのタイプが判定される。例えば、リソースに関する情報は、アウトオブバンド管理ツールを使用して、またはディスクがリソースに接続されているかのようにディスクイメージ (例えば、ISO イメージ) をリソースに接続して、リソース情報を識別するために使用することができるツールを有するオペレーティングシステムを一時的にブートすることによって、mac アドレスから収集することができる。次に、ステップ 1393 で、リソースは、構成されるか、または PXE もしくは flexboot などのために事前に構成されているとして識別される。次に、ステップ 1394 で、リソースの電源がオンになり、PXE、Flexboot または同様のブートが行われる (または、リソースが一時的にブートされ、再度電源がオンにされた場合)。次に、ステップ 1395 で、リソースはインバンド管理接続または SAN からブートされる。ステップ 1396 では、データは、図 13D のステップ 1383 を参照して説明したのと同様の方法で、リソースによってアクセス可能なディスクにコピーされる。ステップ 1397 で、リソースは、図 13D のステップ 1384 に関して上に説明したのと同様の方法で、ディスク (複数可) からブートするように構成される。リソースが PXE、flexboot などのために事前に構成されていると識別される場合、ファイルは、1393 から 1396 までの任意のステップでコピーされてよい。インバンド管理が有効にされた場合、ステップ 1398 で無効化にされてよく、ステップ 1399 でアプリケーションネットワークまたは外部ネットワークを再接続するか、または有効にしてよい。

【0224】

さらに、OOBM 以外の技術を使用して、リソースをリモートで有効にし (電源をオンにするなど)、リソースがブートされたことを検証することができることを理解されたい。例えば、システムは、ユーザに電源ボタンを押すように促し、システムがブートしたことをコントローラに手動で伝える (または、キーボード／コントローラへのコンソール接続を使用する)。さらに、システムは、システムがブートされ、コントローラがログオンし、リブートするように指示すると、IBM を通してコントローラに ping することができる (例えば、ssh、telnet またはネットワーク上の別の方法を通して)。例えば、コントローラは ssh を使用してリブートコマンドを送信することができる。PXE が使用されていて、OOBM がない場合は、いずれの場合でも、システムは、電源をオンにしようとしも一でリソースに指示するか、または手動で電源をオンにするようにユーザに指示する方法を有するべきである。

【0225】

コントローラ及び／または環境の配備

例示的な実施形態では、コントローラは、元になるコントローラ 200 からシステム内に配備されてよい（このような元になるコントローラ 200 は、「メインコントローラ」と呼ぶことができる）。したがって、メインコントローラは、分離されるか、または分離可能な IT システムまたは環境であり得るシステムまたは環境をセットアップし得る。

【0226】

本明細書に記載される環境は、コンピュータシステム内の、互いに相互運用することができるリソースの集合を指す。コンピュータシステムは、その中に複数の環境を含んでよいが、これは必須ではない。環境のリソース（複数可）は、その環境で実行される 1 つまたは複数のインスタンス、アプリケーションまたはサブアプリケーションを備え得る。さらに、環境は、1 つまたは複数の環境またはサブ環境を含み得る。環境は、コントローラを含む場合と含まない場合があり、環境は、1 つまたは複数のアプリケーションを動作させる場合がある。環境のそのようなリソースは、例えば、環境内のアプリケーションを含む特定の環境を実行するために使用されるネットワークングリソース、計算リソース、ストレージリソース及び／またはアプリケーションネットワークを含み得る。したがって、環境は、1 つまたは複数のアプリケーションの機能を提供し得ることを理解されたい。いくつかの例では、本明細書に記載される環境は、他の環境から物理的または仮想的に分離されるか、または分離可能であってよい。さらに、他の例では、環境は、他の環境へのネットワーク接続を有し、そのような接続は、必要に応じて無効または有効にされてよい。

【0227】

さらに、メインコントローラは、様々な環境において、または別個のシステムとして、1 つまたは複数の追加のコントローラをセットアップ、配備及び／または管理してよい。このような追加のコントローラは、メインコントローラから独立してよい、または、独立状態になってよい。このような追加のコントローラは、たとえメインコントローラから独立していても、または疑似的に独立していても、動作中の様々な時点で、メインコントローラ（または監視アプリケーションを介した別個のモニタまたは環境）から命令を受けるか、またはそれらに情報を送信し得る。環境は、セキュリティのために（例えば、環境を互いに及び／またはメインコントローラから分離可能にすることによって）及び／または様々な管理目的のために構成されてよい。ある環境は外部ネットワークに接続されてよいが、別の関連する環境は外部ネットワークに接続することも、接続しないこともできる。

【0228】

メインコントローラは、環境またはアプリケーションが別個のシステムであるかどうか、及びそれらがコントローラまたはサブコントローラを備えるかどうかに関わらず、環境またはアプリケーションを管理してよい。メインコントローラは、グローバル構成ファイルまたは他のデータの共有ストレージも管理してよい。メインコントローラはさらに、グローバルシステムルール（例えば、システムルール 210）、またはそれらのサブセットを、異なるコントローラに対して、それらの機能に応じて解析してよい。各新しいコントローラ（「サブコントローラ」と呼ぶことができる）は、メインコントローラの構成ルールのサブセットであり得る新しい構成ルールを受信してよい。コントローラに配備されるグローバル構成ルールのサブセットは、セットアップされる IT システムのタイプに依存してよく、またはそれに対応してよい。メインコントローラは、新しいコントローラまたは別個の IT システムをセットアップまたは配備してよく、これらは、その後、例えば、出荷または配送またはその他のために、メインコントローラから永久的に分離される。グローバル構成ルール（またはそのサブセット）は、様々な環境においてアプリケーションまたはサブアプリケーションをセットアップするためのフレームワーク、及びそれらが互いに対話し得る方法を定義してよい。このようなアプリケーションまたは環境は、メインコントローラによって配備されるグローバル構成ルールのサブセットを備えるサブコントローラ上で実行してよい。いくつかの例では、そのようなアプリケーションまたは環境は、メインコントローラによって管理することができる。しかしながら、他の例では、そのようなアプリケーションまたは環境は、メインコントローラによって管理されない。アプリケーションまたは環境を管理するためにメインコントローラから新しいコントローラが

生成される場合、新しいコントローラによる制御を容易にするために、複数のアプリケーションに亘ってアプリケーションの依存関係チェックを行うことができる。

【0229】

したがって、例示的な実施形態では、システムは、別のコントローラを配備するように構成されたメインコントローラ、またはそのような他のコントローラを備えるITシステムを含み得る。このような実装システムは、メインコントローラから完全に切断されるように構成されてよい。一旦独立すると、そのようなシステムは、スタンドアロンシステムとして動作するように構成されてよい、または動作中の様々な離散時間または連続時間において、メインコントローラなどの別のコントローラ（またはアプリケーションを備えた環境）によって制御または監視されてよい。

10

【0230】

図14Aは、メインコントローラ1401が、異なるシステム1400a及び1400b上にそれぞれ配備されたコントローラ1401a及び1401bを有する例示的なシステムを示す（ここで、1400a及び1400bは、サブシステムと呼ばれてよい。ただし、サブシステム1400a及び1400bは、環境としても機能し得ることを理解されたい）。メインコントローラ1401は、上述のコントローラ200と同様の方法で構成することができる。したがって、コントローラロジック205、グローバルシステムルール210、システム状態220及びテンプレート230を含み得る。

【0231】

システム1400a及び1400bはそれぞれ、リソース1420a、1420bにそれぞれ結合されたコントローラ1401a、1401bを備える。メインコントローラ1401は、サブシステム1400aのコントローラ1401a及びサブシステム1400bのコントローラ1401bなどの1つまたは複数の他のコントローラに結合されてよい。メインコントローラ1400のグローバルルール210は、他のコントローラを管理及び制御し得るルールを含んでよい。メインコントローラ1401は、コントローラロジック205、システム状態220及びテンプレート230と共にそのようなグローバルルール210を使用して、本明細書の図1～図13Eを参照して説明したのと同様の方法で、コントローラ1401a、1401bを通してサブシステム1400a、1400bをセットアップ、プロビジョニング及び配備してよい。

20

【0232】

例えば、メインコントローラ1401は、グローバルルール210（またはそのサブセット）がコントローラ1401a、1401b及びそのサブシステム1400a、1400bの動作を指示するように、グローバルルール210（またはそのサブセット）をそれぞれルール1410a、1410bとしてサブシステム1400a、1400bにロードしてよい。各コントローラ1401a、1401bは、グローバルルール210の同じサブセットまたは異なるサブセットであってよいルール1410a、1410bを有し得る。例えば、グローバルルール210のどのサブセットが所与のサブシステムにプロビジョニングされるかは、配備されるサブシステムのタイプに依存し得る。さらに、コントローラ1401は、システムリソース1420a、1420bまたはコントローラ1401a、1401bにロードされることになるデータをロードしてよい、またはデータのロードを指示してよい。

30

【0233】

メインコントローラ1401は、例えば、図13A～図13Eに説明されるリソースの配備及び管理を参照して、本明細書に説明されるような方法で、配備または管理の様々な段階において有効または無効にされ得るインバンド管理接続270（複数可）及び／またはアウトオブバンド管理接続260（複数可）またはSAN接続280を通して他のコントローラ1401a、1401bに接続されてよい。インバンド管理接続270またはアウトオブバンド管理接続260の選択的な有効化及び無効化を使用することによって、サブシステム1400a、1400bは、サブシステム1400a、1400bが、様々な時点で、メインシステム100またはコントローラ1401または互いについての知識を

40

50

全く有さない（または、知識が制限されているか、制御されているか、または禁止されている）方法で配備されてよい。

【0234】

例示的な実施形態では、メインコントローラ1401は、メインコントローラ1401が複数のITシステムを配備及び／または実行し得るように、メインコントローラ1401によって配備及び構成されたローカルコントローラ1401a、1401bを有する集中型のITシステムを動作させてよい。このようなITシステムは、互いに独立していても、独立していなくてもよい。メインコントローラ1401は、それが作成したITシステムから分離されるか、またはエアギャップされた別個のアプリケーションとして監視をセットアップしてよい。監視のための別個のコンソールは、メインコントローラとローカルコントローラ（複数可）との間の接続、及び／または選択的に有効または無効にされ得る環境間の接続を備えてよい。コントローラ1401は、例えば、ビジネス、データストレージを備えた製造システム、データセンタならびに他の様々な機能ノードを含むが、これらに限定されず、各々が停止または危険にさらされる場合に異なるコントローラを有する様々な用途のための分離されたシステムを配備してよい。このような分離は完全であっても永久的であってもよく、または、例えば、一時的、時間またはタスクに依存して、通信方向に依存して、または他のパラメータに依存して、疑似的に分離されてもよい。例えば、メインコントローラ1401は、いくつかの定義済みの状況に限定されていても、限定されていなくてもよい命令をシステムに提供するように構成されてよく、一方、サブシステムは、メインコントローラと通信する能力が限定されているか、または全くない場合がある。したがって、このようなサブシステムは、メインコントローラ1401を危険にさらすことができない可能性がある。メインコントローラ1401及びサブコントローラ1401a、1401bは、例えば、インバンド管理270を無効にすることによって、一方向の書き込みによって及び／または通信をアウトオブバンド管理260に制限することによって、本明細書に説明するように（以下に説明される特定の例で）互いから分離され得る。例えば、侵害が発生した場合、1つまたは複数のコントローラは、1つまたは複数の他のコントローラに対してインバンド管理接続270を無効にして、侵害またはアクセスの拡散を防ぐことができる。システムセクションは、オフにすることも、分離することもできる。

【0235】

サブシステム1400a、1400bはさらに、インバンド管理270またはアウトオブバンド管理260を通して別の環境またはシステムとリソースを共有するか、またはこれらに接続されてよい。

【0236】

図14B及び図14Cは、メインコントローラを備えたコントローラをプロビジョニングするための可能なステップを示す例示的なフローである。

【0237】

図14Bにおいて、ステップ1460で、メインコントローラは、リソース1420aまたは1420bなどのリソースをプロビジョニングまたはセットアップする。ステップ1461で、メインコントローラは、サブコントローラをプロビジョニングまたはセットアップする。メインコントローラは、システム内にリソースをセットアップしてステップ1460及び1461を実行するために、上述の技術を使用することができる。さらに、図14Bは、ステップ1461の前にステップ1460が実行されることを示すが、これは必須ではないことを理解されたい。メインコントローラ1401は、そのシステムルール210を使用して、どのリソースが必要であるかを判断し、システムまたはネットワーク上にリソースを配置することができる。メインコントローラは、ステップ1461で、サブコントローラをセットアップするためにシステムにシステムルール210をロードすることによって（または、それ自体のシステムルールをセットアップして取得する方法についてサブコントローラに命令を提供することによって）、サブコントローラをセットアップまたは配備してよい。これらの命令には、リソースの構成、アプリケーションの構成

10

20

30

40

50

、サブコントローラによって実行されるITシステムを作成するためのグローバルシステムルール、メインコントローラに再接続して新規のまたは変更されたルールを収集する命令、アプリケーションネットワークから切断して新しい本番環境のための余地を作成する命令が含まれ得るが、これらに限定されない。リソースを配備した後、ステップ1463で、メインコントローラは、システムルール210及び／またはシステム状態220への更新を介して、サブコントローラにリソースを割り当ててよい。

【0238】

図14Cは、配備の代替的なプロセスフローを示す。図14Cの例では、メインコントローラは、ステップ1470でサブコントローラを配備する（ステップ1461に関して説明したように進めることができる）。次に、ステップ1475で、サブコントローラは、例えば図3C及び図7Bに示すような技術を使用してリソースを配備する。

10

【0239】

図15Aは、システム100のメインコントローラ1501が環境1502、1503及び1504を生成する例示的なシステムを示す。環境1502はリソース1522を含み、環境1503はリソース1523を含み、環境1504はリソース1524を含む。さらに、環境1502、1503、1504は、共有リソース1525のプールへのアクセスを共有してよい。このような共有リソースは、例えば、共有データセット、API、または互いに通信する必要がある実行中のアプリケーションを含み得るが、これらに限定されない。

【0240】

20

図15Aの例では、各環境1502、1503、1504は、メインコントローラ1501を共有する。メインコントローラ1501のグローバルシステムルール210は、環境を配備及び管理するルールを含み得る。リソース1522、1523及び／または1524は、1つまたは複数のアプリケーションを管理するために、それぞれの環境1501、1502、1503によって必要とされ得る。そのようなアプリケーションのための構成ルールは、そのような各環境の動作方法ならびに他のアプリケーション及び環境との対話の方法を定義するために、メインコントローラ（または、存在する場合は、環境内のローカルコントローラ）によって実施されてよい。メインコントローラ1401は、コントローラロジック205、システム状態220及びテンプレート230と共にグローバルルール210を使用して、本明細書の図1～図14Cを参照して説明したリソース及びシステムの配備と同様の方法で、環境をセットアップ、プロビジョニング及び配備してよい。環境がローカルコントローラを含む場合、メインコントローラ1501は、グローバルルール（またはそのサブセット）がその環境の動作を定義するように、グローバルルール210（またはそのサブセット）をローカルコントローラまたは関連するストレージにロードしてよい。

30

【0241】

コントローラ1501は、システムルール210を有する構成ルールを使用して、環境1502、1503、1504のそれぞれのリソース1522、1523、1524、及び／または共有リソース1525を配備及び構成してよい。コントローラ1501はさらに、環境を監視するか、リソース1522、1523、1524（または共有リソース1525）を構成して、各環境1502、1503、1504の監視を可能にしてよい。このような監視は、有効または無効にし得る別個の監視コンソールへの接続を介してよい、またはメインコントローラを通してよい。メインコントローラ1501は、図13A～図13E及び図14Aのリソースの配備及び管理を参照して本明細書に説明されるような方法で、配備または管理の様々な段階で有効または無効にされ得るインバンド管理接続270（複数可）及び／またはアウトオブバンド管理接続260（複数可）またはSAN接続280を通して、環境1502、1503、1504のうちの1つまたは複数に接続してよい。インバンド管理接続270またはアウトオブバンド管理接続260またはSAN接続280の有効化及び無効化を使用して、環境1502、1503、1504は、それらが様々な時点で、メインシステム100またはコントローラ1501の知識を全く有さな

40

50

いか、知識が制限されているか、制御されているか、または互いの接続に関して知識を全く有さないか、知識が制限されているか、制御されている方法で配備され得る。

【0242】

環境は、外部である外部環境に接続する外部ネットワーク1580と結合されるか、他のリソースと対話する1つまたは複数のリソースを備えてよい。環境は、物理的であっても非物理的であってもよい。この文脈での「非物理的」とは、環境が同じ物理ホスト（複数可）を共有しているが、仮想的に互いに分離されていることを意味する。環境及びシステムは、同一のハードウェア、類似しているが異なるハードウェアまたは同一ではないハードウェアに配備されてよい。いくつかの例では、環境1502、1503、1504は互いの有効なコピーであり得るが、他の例では、環境1502、1503、1504は、互いに異なる機能性を提供し得る。一例として、環境のリソースは、サーバであってよい。

10

【0243】

本明細書に説明される技術に従って、システム及びリソースを別個の環境またはサブシステムに配置することにより、セキュリティ及び／または性能のために、アプリケーションを分離することを可能にし得る。環境を分離することで、危険にさらされたリソースの影響を軽減することもできる。例えば、ある環境には機密データが含まれる場合があり、インターネットへの公開が少なくなるように構成されてよく、一方、別の環境では、インターネット向けアプリケーションをホストしてよい。

【0244】

図15Bは、図15Aに示すコントローラが環境をセットアップする例示的なプロセスフローを示す。このような例では、システムは、新しい環境を作成しセットアップするようにタスクを課されてよい。これは、ユーザの要求によって、または特定のタスクまたは一連のタスクに従事するときに実行されるシステムルールによってトリガされてよい。以下に説明するように、図17A～図18Bは、システムが新しい環境を作成する特定の変更管理タスクまたは一連のタスクの例を示す。しかしながら、コントローラが新しい環境を作成してセットアップし得る状況は多数存在してよい。

20

【0245】

したがって、図15Bを参照すると、新しい環境をセットアップする際に、コントローラは、環境ルールを選択する（ステップ1500.1）。環境ルールに従って、グローバルシステムルール210及びテンプレート230を使用して、コントローラは、環境のためにリソースを見つける（ステップ1500.2）。ルールは、環境に必要なリソースが見つかるまでコントローラが経する好適なリソース選択の階層を有し得る。ステップ1500.3で、コントローラは、例えば、図3Cまたは図7Bに説明される技術を使用して、ステップ1500.2で見つけたリソースを環境に割り当てる。次に、コントローラは、新しい環境と他のシステムコンポーネントとの間の互換性のある効率的な接続を確保するために、新しい環境に関してシステムのネットワーキングリソースを構成する（ステップ1500.4）。システム状態は、各リソースが有効になり、各テンプレートが処理されるたびに、ステップ1500.5で更新される。次に、コントローラは、環境のリソースの統合及び相互運用性をセットアップして有効にし、新しい環境を配備するために任意のアプリケーションの電源をオンにする（ステップ1500.6）。システム状態は、環境が利用可能になると、ステップ1500.7で再び更新される。

30

40

【0246】

図15Cは、図15Aに示すコントローラが複数の環境をセットアップする例示的なプロセスフローを示す。複数の環境をセットアップする場合は、各環境に対して図15Bに説明する技術を使用して、環境を並列にセットアップしてよい。しかしながら、環境は、図15Cに説明されるように、連続した順序または連続してセットアップされ得ることを理解されたい。図15Cを参照すると、ステップ1500.10で、コントローラは、（図15Bのステップ1500.1に関して説明するように実行することができる）第1の新しい環境をセットアップし、配備する。異なるタイプの環境及び異なる環境の相互運用方法には、異なる環境ルールが存在し得る。ステップ1500.11で、コントローラは

50

、次の環境に対する環境ルールを選択する。ステップ1500.12で、コントローラは、システムルール210によって定義することができる優先順位に従ってリソースを見つける。ステップ1500.13で、コントローラは、ステップ1500.12で見つけたリソースを次の環境に割り当てる。環境はリソースを共有してもよく、共有しなくてもよい。ステップ1500.14で、コントローラは、システムルール210を使用して、次の環境に関して及び依存関係を有する環境間で、システムのネットワーキングリソースを構成する。システム状態は、各リソースが有効であり、テンプレートが処理され、ネットワーキングリソースが環境の依存関係を含めて構成されるたびに、ステップ1500.15で更新される。次に、コントローラは、次の環境及び環境間のリソースの統合及び相互運用性をセットアップして有効にし、新しい環境を配備するために任意のアプリケーションの電源をオンにする（ステップ1500.16）。システム状態は、次の環境が利用可能になると、ステップ1500.17で更新される。

【0247】

監視をサポートする一方向通信

図16Aは、第1のコントローラ1601が、1601a、1601b及び／または1601bなどの1つまたは複数のコントローラをセットアップするためのメインコントローラとして動作する例示的な実施形態を示す。メインコントローラ1601は、コントローラ200／1401／1501などのコントローラに関して上述した技術を使用して、動作において互いに依存してもよく、依存しなくてもよい環境1602、1603、1604として、複数のクラウドホスト、システム及び／またはアプリケーションを生成するために使用されてよい。図16Aに示すように、ITシステム、環境、クラウド及び／またはそれらの任意の組み合わせ（複数可）は、環境1602、1603、1604として生成され得る。環境1602は第2のコントローラ1601aを含み、環境1603は第3のコントローラ1601bを含み、環境1604は第4のコントローラ1601cを含む。環境1602、1603、1604はそれぞれ、1つまたは複数のリソース1642、1643、1644をそれぞれ含み得る。リソースは、それらの上で実行中であってよい1つまたは複数のアプリケーション1642、1643、1644を含み得る。これらのアプリケーションは、共有されているかどうかに関わらず、割り当てられたリソースに接続してよい。これらのまたは他のアプリケーションは、インターネット上で、または共有アプリケーションもしくはアプリケーションネットワークを含み得るプール1660内の1つまたは複数の共有リソース上で実行してよい。アプリケーションは、ユーザまたは環境もしくはクラウドのうちの1つまたは複数にサービスを提供してよい。環境1602、1603、1604は、リソースまたはデータベースを共有してよい、及び／または特定の環境に特に割り当てられたプール1660内のリソースを含み得るもしくは使用してよい。メインコントローラ1601及び／または1つまたは複数の環境を含むシステムの様々なコンポーネントはさらに、アプリケーションネットワークまたはインターネットなどの外部ネットワーク1615に接続可能であってよい。

【0248】

任意のリソース、環境またはコントローラと、別のリソース、環境、コントローラまたは外部接続との間に、本明細書の図13A～図13Eに関して説明される方法で、選択的に有効及び／または無効にされるように構成され得る接続が存在してよい。例えば、インバンド管理接続270、アウトオブバンド管理接続270またはSAN接続280を介して、または物理的に切断することによって、任意のリソース、コントローラ、環境または外部接続を無効にするか、またはコントローラ1601、環境1602、環境1603及び／または環境1604、リソース、あるいはアプリケーションから切断してよい。一例として、コントローラ1601と環境1602、1603、1604のいずれかとの間のインバンド管理接続270は、コントローラ1601を保護するために無効にされてよい。別の例として、このようなインバンド管理接続270（複数可）は、環境1602、1603、1604の動作中に選択的に無効または有効にされてよい。本明細書の図13A～図13Eに関して説明したセキュリティ目的に加えて、環境1602、1603、16

10

20

30

40

50

04からメインコントローラ1601を無効にするか、または切断することにより、メインコントローラ1601は、環境1602、1603、1604をクラウドとして生成することを可能にしてよく、環境1602、1603、1604は、その後、メインコントローラ1601または他のクラウドもしくは環境から分離されてよい。この意味で、コントローラ1601は、複数のクラウド、ホストまたはシステムを生成するように構成される。

【0249】

本明細書に説明される無効化または切断要素を使用して、ユーザは、特定の用途のために、メインコントローラ1601を通じた環境への限定的なアクセスを可能にされてよい。例えば、開発者は、開発環境へのアクセスを提供されてよい。別の例として、アプリケーションの管理者は、特定のアプリケーションまたはアプリケーションネットワークに限定されてよい。別の例として、メインコントローラが生成する環境またはコントローラによって、メインコントローラが危険にさらされることなく、データを収集するために、ログは、メインコントローラ1601を通して見えるようにされてよい。

【0250】

メインコントローラ1601が環境1602をセットアップした後、環境1602は、メインコントローラ1601から切断され、こうして、環境1602は、メインコントローラ1601とは独立して動作してよい、及び／またはメインコントローラ1601または、環境1602に関連付けられるか、もしくは環境によって実行される他のアプリケーションによって選択的に監視及び維持されてよい。

【0251】

環境1602などの環境は、購入者またはユーザによる環境1602へのアクセスを可能にするユーザインタフェースまたはコンソール1640に結合されてよい。環境1602は、アプリケーションとしてユーザコンソールをホストしてよい。環境1602は、ユーザによってリモートにアクセスされてよい。各環境1602、1603、1604は、共通のまたは別個のユーザインタフェースまたはコンソールによってアクセスされてよい。

【0252】

図16Bは、環境1602、1603、1604が別の環境1641に書き込むように構成され得る例示的なシステムを示し、ここで、例えば、コンソール（環境1641と直接的または間接的のいずれかで接続してよい任意のコンソールであってよい）を使用してログを見てよい。このようにして、環境1641は、環境1602、1603、1604のうちの1つまたは複数がイベントを書き込むログサーバとして機能することができる。次に、メインコントローラ1601は、後述するように、このような環境1602、1603、1604との直接的な接続を維持することなく、ログサーバ1641にアクセスして、環境1602、1603、1604上のイベントを監視することができる。環境1641はさらに、メインコントローラ1601から選択的に切断されることがあり、他の環境1602、1603、1604からの読み取り専用となるように構成されてよい。

【0253】

メインコントローラ1601は、図16Cに示すように、メインコントローラ1601がその環境1602、1603、1604のいずれかから切断されていても、その環境1602、1603、1604の一部または全てを監視するように構成されてよい。図16Cは、メインコントローラ1601と環境1602、1603、1604との間のインバンド管理接続270が切断されており、これは、環境1602、1603、1604が危険にさらされた場合にメインコントローラ1601を保護するのに役立ち得る。図16Cに示すように、アウトオブバンド接続260は、メインコントローラ1601と環境1602との間のインバンド接続270が切断されていても、メインコントローラ1601と1602などの環境との間で維持されてよい。さらに、環境1641は、選択的に有効または無効され得るメインコントローラ1601への接続を有してよい。メインコントローラ1601は、環境1602、1603、1604から分離されるか、またはエアギャップされた環境1641内の別個のアプリケーションとして監視をセットアップしてよい。メ

インコントローラ1601は、監視のために一方向通信を使用してよい。例えば、ログは、環境1602、1603、1604から環境1641への一方向通信を通して提供されてよい。このような一方向の書き込みを通して、かつ環境1641とメインコントローラ1601の間の接続を介して、メインコントローラ1601は、メインコントローラ1601と環境1602、1603、1604との間にインバンド接続270がない場合であっても、環境1641を介してデータを収集して環境1602、1603、1604を監視することができ、これによって環境1602、1603、1604がメインコントローラ1601を危険にさらすリスクを軽減してよい。アクセスは、フィルタリングされるか、または制御されてよく、及び／またはアクセスは、インターネットから独立してよい。例えば、図16Dに示すように、メインコントローラ1601と環境1602の間のインバンド接続270が接続されている場合、メインコントローラ1601は、環境1602をインターネットなどの外部ネットワーク1615から切断するようにネットワークスイッチ1650を制御することができる。環境1602がインバンド接続270によってメインコントローラ1601と接続されているときに、環境1602を外部ネットワーク1615から切断すると、メインコントローラ1601のセキュリティを向上させることができる。

【0254】

したがって、図16B～図16Dの例示的な実施形態は、環境1602、1603、1604への公開を最小限に抑えながら、メインコントローラがこれらの環境1602、1603、1604を安全に監視できる方法を示すことを理解されたい。したがって、メインコントローラ1601は、環境1602、1603、1604が一方向の書き込み権限を有することができる環境1641のログサーバを介してそれらの環境を監視するための機構を維持しながら、環境1602、1603、1604からコントローラ自体を切断する（または少なくともインバンドリンクからコントローラ自体を切断する）ことができる。したがって、環境1641のログを見直す過程で、環境1602がマルウェアによって危険にさらされる可能性があることをメインコントローラ1601が発見した場合、メインコントローラ1601は、SDNツールを使用して、アウトオブバンド接続260のみが存在するように環境1602を分離してよい（例えば、図16Cを参照）。さらに、コントローラ1601は、環境1602の管理者に問題の可能性に関する通知を送信してよい。コントローラはさらに、危険にさらされた環境と他の環境1603、1604のいずれかとの間の任意の接続（例えば、インバンド管理接続270）を選択的に無効にすることによって、危険にさらされた環境1602を分離することができる。別の例では、メインコントローラ1601は、ログを通して、環境1603内のリソースが熱すぎることを発見してよい。これにより、メインコントローラが介入し、アプリケーションまたはサービスを環境1603から別の環境（既存の環境または新しく生成された環境のいずれであっても）に移行してよい。

【0255】

コントローラ1601はさらに、購入者またはユーザの要求に従って同様の1つまたは複数のシステムをセットアップしてよい。図16Eに示すように、購入アプリケーション1650は、例えば、コンソールまたはその他に備えられてよく、これにより、購入者が、クラウド、ホスト、システム環境またはアプリケーションを購入し、購入者のために、それらをセットアップする要求を行うことができる。購入アプリケーション1650は、環境1602をセットアップするようにコントローラ1601に指示してよい。環境1602は、例えば、リソースを環境1602に割り当てるか、またはアサインすることによって、ITシステムを配備または構築するコントローラ1601aを備えてよい。

【0256】

図16Fは、環境1602、1603、1604がそれぞれクラウドとして動作し、コントローラを備える場合も、備えない場合もある環境で使用され得るユーザインタフェース1632、1633、1634を示す。ユーザインタフェース1632、1633、1634（それぞれ、環境1602、1603、1604に対応する）は、それぞれ、ユー

10

20

30

40

50

ザインタフェースと環境との接続を管理するメインコントローラ1601を通して接続してよい。代替的または追加的に、インタフェース1640a（コンソールの形態を取ってよい）は、環境1602に直接、結合されてよく、インタフェース1640b（コンソールの形態を取ってよい）は、環境1603に直接、結合されてよく、インタフェース1640c（コンソールの形態を取ってよい）は、環境1604に直接、結合されてよい。メインコントローラ1601との接続が分離されているか、切断されているか、または無効にされているかに関わらず、ユーザは、インタフェースのうちの1つまたは複数を使用して、環境またはクラウドを使用してよい。

【0257】

変更管理サポートのためのシステムのクローニング及びバックアップ

環境1602、1603、1604のいくつかは、開発者が使用する典型的なセットアップソフトウェアのクローンであってよい。また、これらの環境は、例えば、異なる位置にある別のデータセンタの環境をクローンすることで位置が原因の遅延を削減するなど、拡張の方法として、現在の作業環境のクローンであってよい。

【0258】

したがって、システム及びリソースを個別の環境またはサブシステムにセットアップするメインコントローラは、ITシステムの一部のクローニングまたはバックアップを可能にし得ることを理解されたい。これは、本明細書で説明するように、テスト及び変更管理で使用されてよい。このような変更には、コード、構成ルール、セキュリティパッチ、テンプレートへの変更、及び／または他の変更が含まれ得るが、これらに限定されない。グローバルルールは、本明細書の様々な例で記載される変更管理で 사용할 ことができるバックアップルールを含むサブセットを含み得る。したがって、バックアップルール（その例は、本明細書の他の箇所に記載される）が変更管理に使用できることは理解されたい。バックアップルールを実施するシステムの例は、図21A～21Jに関してさらに詳細に記載される。

【0259】

例示的な実施形態によれば、本明細書に説明されるITシステムまたはコントローラは、1つまたは複数の環境をクローンするように構成されてよい。新しい環境またはクローン環境は、元の環境と同じリソースを備える場合と、備えない場合がある。例えば、新しい環境またはほぼクローンされた環境では、物理リソース及び／または仮想リソースの全く異なる組み合わせを使用することが望ましいか、あるいは必要になる場合がある。使用の最適化を管理し得る異なる位置または時間帯に環境をクローンすることが望ましい場合がある。仮想環境に環境をクローンすることが望ましい場合がある。環境のクローンの際、コントローラまたはメインコントローラのグローバルシステムルール210及びグローバルテンプレート230は、様々なタイプのハードウェアを構成及び／または実行する方法に関する情報を含み得る。システムルール210内の構成ルールは、特定の利用可能なリソースを考慮して、リソース及びアプリケーションがより最適になるように、リソースの配置及び使用を指示してよい。

【0260】

メインコントローラ構造は、システム及びリソースを別個の環境またはサブシステムにセットアップする能力を提供し、クローン環境のための構造を提供し、開発環境を作成するための構造を提供し、及び／またはアプリケーション及び／またはリソースの標準化されたセットを配備するための構造を提供する。このようなアプリケーションまたはリソースは、例えば、アプリケーションの開発及び／または実行、またはITシステム及び他の災害復旧アプリケーションの一部のバックアップまたはバックアップからの復元（例えば、LAMP（apache、mysql、php）スタック、ウェブフロントエンド及びreact／reduxを実行するサーバを含むシステム、ならびにnode.jsを実行するリソースならびにmongoデータベース及び他の標準化された「スタック」）に使用可能なものを含み得るが、これらに限定されない。場合により、メインコントローラは、別の環境のクローンである環境を配備してよく、元の環境を作成するために使用され

10

20

30

40

50

た構成ルールのサブセットから構成ルールを導出してよい。

【0261】

例示的な実施形態によれば、システムまたはシステムのサブセットの変更管理は、1つまたは複数の環境、及びそのような環境の構成ルールまたは構成ルールのサブセットをクローンすることによって達成されてよい。例えば、コード、構成ルール、セキュリティパッチ、テンプレートへの変更、ハードウェアの変更、コンポーネント及び依存アプリケーションの追加／削除、ならびにその他の変更を行うために、変更が必要になる場合がある。

【0262】

例示的な実施形態によれば、システムに対するそのような変更は、変更の直接的な手動入力のエラーを回避するために自動化されてよい。変更は、稼働中のシステムに変更を自動的に実施する前に、開発環境でユーザによってテストされてよい。例示的な実施形態によれば、本番環境と同じ構成ルールを使用して構成される環境を、コントローラを使用して、自動的に電源をオンにし、プロビジョニングし、及び／または構成することによって、稼働中の本番環境をクローンしてよい。クローン環境は、実行して稼働させることができる（一方、バックアップ環境は、変更をロールバックする必要がある場合に備えて、緊急用にそのままにしておくことが望ましい）。これは、システムルール210、テンプレート230及び／またはシステム状態220を使用して、上の図1～図16Fを参照して説明したような新しいシステムまたは環境を作成、構成及び／またはプロビジョニングするために、コントローラを使用して行われてよい。後に本番環境に実装する変更をテストするために、新しい環境を開発環境として使用してよい。コントローラは、そのような環境のインフラストラクチャを、ソフトウェアで定義された構造から開発環境に生成してよい。

【0263】

本明細書で定義される本番環境とは、開発及びテスト専用の環境、すなわち開発環境とは対照的に、システムを動作させるために使用されている環境を意味する。

【0264】

本番環境がクローンされると、インフラストラクチャまたはクローン開発環境は、本番環境と同様に、グローバルシステムルール210に従ってコントローラによって構成及び生成される。開発環境の変更は、コード、テンプレート230（既存のテンプレートの変更または新しいテンプレートの作成に関連する変更のいずれか）、セキュリティ及び／またはアプリケーションもしくはインフラストラクチャ構成に対して行われてよい。開発環境に実装された新しい変更が、開発及び／またはテストを通して必要に応じて準備されると、システムは自動的に開発環境に変更を加え、その後、本番環境として稼働状態になるか、配備される。次に、新しいシステムルール210は、環境のコントローラ及び／または特定の環境に対してシステムルールの変更を適用するメインコントローラのいずれかにアップロードされる。システム状態220は、コントローラで更新され、追加または修正されたテンプレート230が実装されてよい。したがって、インフラストラクチャの完全なシステム知識は、それを再作成する機能と共に、開発環境及び／またはメインコントローラによって維持されてよい。本明細書で使用される完全なシステム知識は、リソースの状態、リソースの利用可能性及びシステム構成のシステム知識を含み得るが、これらに限定されない。完全なシステム知識は、コントローラによって、システムルール210、システム状態220から、及び／またはインバンド管理接続270（複数可）、アウトオブバンド管理接続260（複数可）S A N接続280（複数可）を使用してリソースにクエリすることによって、収集されてよい。リソースは特に、リソース、ネットワークまたはアプリケーションの使用率、構成状態または利用可能性を判断するためにクエリすることができる。

【0265】

クローンされたインフラストラクチャまたは環境は、システムルール210を介して定義されたソフトウェアであってよいが、これは必須ではない。クローンされたインフラストラクチャまたは環境は、一般に、フロントエンドまたはユーザインタフェースと、1つ

または複数の割り当てられたリソースとを備えてもよく、備えなくてもよい。割り当てられたリソースは、計算リソース、ネットワークリソース、ストレージリソース、及び／またはアプリケーションネットワークリソースを含んでもよく、含まなくてもよい。この環境は、フロントエンド、ミドルウェア及びデータベースとして構成されてもよく、構成されなくてもよい。サービスまたは開発環境は、本番環境のシステムルール 210 でブートされてよい。コントローラが使用するために割り当てられるインフラストラクチャまたは環境は、特にクローンのためにソフトウェアで定義されてよい。したがって、環境は、システムルール 210 によって配備可能であってよく、同様の手段によってクローン可能であってよい。クローン環境または開発環境は、変更が望まれる前または変更が望まれるときに、システムルール 210 を使用してローカルまたはメインコントローラによって自動的にセットアップされてよい。

10

【0266】

本番環境のデータは、開発環境が本番環境から分離されるまで、読み取り専用のデータストレージに書き込まれてよく、その後、開発環境によって開発及びテストプロセスで使用されることになる。

【0267】

ユーザまたはクライアントは、本番環境がオンラインである間に、開発環境で変更を行い、テストしてよい。データストレージ内のデータは開発中に変更されることがあり、開発環境で変更がテストされる。揮発性または書き込み可能なシステムでは、開発環境がセットアップまたは配備された後に、本番環境のデータとのホット同期も使用されてよい。システム、アプリケーション及び／または環境に対する所望の変更を開発環境に対して行い、開発環境でテストしてよい。次に、システムルール 210 のスクリプトに対して所望の変更を行い、環境またはシステム全体及びメインコントローラに対して新しいバージョンを作成する。

20

【0268】

別の例示的な実施形態によれば、新しく開発された環境は、その後、以前の本番環境が維持されるか、または完全に機能する一方で、新しい本番環境として自動的に実装されてよく、したがって、大量のデータを失うことなく、以前の状態の本番環境への復帰が可能である。次に、システムルール 210 内の新しい構成ルールで開発環境がブートされ、データベースが本番データベースと同期され、書き込み可能データベースに切り替えられる。その後、元の本番データベースを読み取り専用データベースに切り替えてよい。以前の本番環境に戻すことが望ましい場合には、以前の本番環境は、以前の本番環境のコピーとして、必要な期間そのまま維持される。

30

【0269】

環境は、物理ホスト及び／または仮想ホスト、ネットワーク及び他のリソースを含み得る単一のサーバまたはインスタンスとして構成されてよい。別の例示的な実施形態では、環境は、物理ホスト及び／または仮想ホスト、ネットワーク及び他のリソースを含む複数のサーバであってよい。例えば、負荷分散されたインターネット向けアプリケーションを形成する複数のサーバが存在してよく、それらのサーバは、複数の API / ミドルウェアアプリケーション（1 つまたは複数のサーバ上でホストされてよい）に接続されてよい。環境のデータベースは、API が環境内でクエリを通信する 1 つまたは複数のデータベースを備えてよい。環境は、静的または揮発性の形態でシステムルール 210 から構築されてよい。環境またはインスタンスは、仮想的であっても、物理的であっても、それぞれの組み合わせであってもよい。

40

【0270】

アプリケーションの構成ルールまたはシステムルール 210 内のシステムの構成ルールは、様々な計算バックエンド（例えば、ベアメタル、AMD `epyc` サーバ、`qemu` / `kvm` 上の `Intel Haswell`）を指定してよく、新しい計算バックエンド上でアプリケーションまたはサービスを実行する方法に関するルールを含み得る。したがって、例えば、テスト用のリソースの利用可能性が低下する状況がある場合には、アプリケ

50

ーションを仮想化してよい。

【0271】

本明細書に説明される例を使用し、これらに従って、テスト環境は、元の環境が物理リソースを使用する仮想リソース上に配備されてよい。図1～図18Bを参照して本明細書に説明されるコントローラを使用し、さらに本明細書に説明するように、システムまたは環境は、物理環境から、全体的または部分的に仮想リソースを備える場合も、備えない場合もある環境にクローンされてよい。

【0272】

図17Aは、システム100が、コントローラ1701と、1つまたは複数の環境、例えば、1702、1703、1704とを備える例示的な実施形態を示す。システム100は静的システムであってよい、すなわち、アクティブなユーザデータがシステムの状態を絶えず変化させたり、頻繁にデータを操作したりしないシステム、例えば、静的ウェブページのみをホストするシステムであってよい。システムは、ユーザ（またはアプリケーション）インタフェース110に結合されてよい。

【0273】

コントローラ1701は、本明細書に説明されるコントローラ200／1401／1501／1601と同様の方法で構成することができ、同様に、グローバルシステムルール210、コントローラロジック205、テンプレート230及びシステム状態要素220を含み得る。コントローラ1701は、本明細書の図14A～図16Fを参照して説明したような方法で、1つまたは複数の他のコントローラまたは環境に結合されてよい。コントローラ1701のグローバルルール210は、他のコントローラ及び／または環境を管理及び制御し得るルールを含んでよい。そのようなグローバルルール210、コントローラロジック205、システム状態220、及びテンプレート230を使用して、本明細書の図1～図16Fを参照して説明したのと同様の方法で、コントローラ1701を通してシステムまたは環境をセットアップ、プロビジョニング、及び配備してよい。各環境は、他の環境に関してを含む環境の動作を定義するグローバルシステムルール210のサブセットを使用して構成されてよい。

【0274】

グローバルシステムルール210は、変更管理ルール1711も含み得る。変更管理ルール1711は、システム100、グローバルシステムルール210及び／またはコントローラロジック205への変更が望まれ得る場合に使用され得るルール及び／または命令のセットを含む。変更管理ルール1711は、ユーザまたは開発者が変更を開発し、テスト環境で変更をテストし、その後、変更をシステムルール210内の構成ルールの新しいセットに自動的に変換することによって変更を実装することを可能にするように構成されてよい。変更管理ルール1711は、（図17Aに示すように）グローバルシステムルール210のサブセットであってよい、またはグローバルシステムルール210とは別個であってよい。変更管理ルールは、グローバルシステムルール210のサブセットを使用してよい。例えば、グローバルシステムルール210は、新しい環境を作成するように構成された環境作成ルールのサブセットを含み得る。変更管理ルール1711は、システム100の一部または全ての態様をコピー及びクローンするために、コントローラ1701によって構成及びセットアップされたシステムまたは環境をセットアップ及び使用するよう構成されてよい。変更管理ルール1711は、テスト及び実装のためにシステムのクローンを使用することによって、実装前に、システムに対して提案された新しい変更のテストを許可するように構成されてよい。変更管理1711は、以下に記載のバックアップルールを含み得る、または、使用してよい。

【0275】

図17Aに示すようなクローン1705は、特定の環境またはシステム100の一部のルール、ロジック、アプリケーション及び／またはリソースを含み得る。クローン1705は、システム100と同様の、または異なるハードウェアを備えてよく、仮想リソースを使用してもよく、使用しなくてもよい。クローン1705は、アプリケーションとして

10

20

30

40

50

セットアップされてよい。クローン1705は、システム100またはコントローラ1701のシステムルール210内の構成ルールを使用してセットアップ及び構成されてよい。クローン1705は、コントローラを備えてもよく、備えなくてもよい。クローン1705は、上でより詳細に説明したように、割り当てられたネットワーキングリソース、計算リソース、アプリケーションネットワーク及び／またはデータストレージリソースを含み得る。このようなリソースは、コントローラ1701によって制御される変更管理ルール1711を使用して割り当てられてよい。クローン1705は、ユーザがクローン1705に変更を加えることを可能にするユーザインタフェースに結合されてよい。ユーザインタフェースは、システム100のユーザインタフェース110と同じであっても、異なってもよい。クローン1705は、システム100全体、または1つまたは複数の環境及び／またはコントローラなどのシステム100の一部に使用されてよい。クローン1705は、システム100の完全なコピーであっても、そうでなくてもよい。クローン1705は、選択的に有効及び／または完全に無効にすることができ、及び／または単一方向の読み取り及び／または書き込み接続に変換し得る、インバンド管理接続270、アウトオブバンド管理接続260及び／またはSAN接続280を介してシステム100に結合されてよい。したがって、クローン環境1705がテスト中に本番環境から分離される
とき、またはクローン環境1705が新しい本番環境としてオンラインになる準備ができる
まで、クローン環境1705内のデータへの接続を変更して、クローンデータを読み取り
専用にしてよい。例えば、クローン1705に環境1702へのデータ接続がある場合、
このデータ接続は分離のために読み取り専用にされてよい。

【0276】

任意のバックアップ1706は、システム全体、または1つまたは複数の環境及び／またはコントローラなどのシステムの一部に使用されてもよく、または使用されなくてもよい。変更管理機能を行うとき、個々のサービスもバックアップされてよい。サービスのバックアップは、例えば、図21A～21Jを参照して、以下に記載するように、バックアップルールを使用して実行されてよい。バックアップ1706は、上でより詳細に説明したように、ネットワーキングリソース、計算リソース、アプリケーションネットワークリソース、及び／またはデータストレージリソースを含み得る。バックアップ1706は、コントローラを備えてもよく、備えなくてもよい。バックアップ1706は、システム100の完全なコピーであってもよい。バックアップ1706は、アプリケーションとして、
またはシステム100と同様の、または異なるハードウェアを使用してセットアップされ
てよい。バックアップ1706は、選択的に有効及び／または完全に無効にすることが
でき、及び／または単一方向の読み取り及び／または書き込み接続に変換され得る、
インバンド管理接続270、アウトオブバンド管理接続260及び／またはSAN接続280
を介してシステム100に結合されてよい。

【0277】

図17Bは、システム変更管理で図17Aのクローン及びバックアップシステムを使用するための例示的なプロセスフローを示す。ステップ1785で、ユーザまたは管理アプリケーションは、システムに対する変更を開始する。このような変更は、コード、構成ルール、セキュリティパッチ、テンプレートへの変更、ハードウェアの変更、コンポーネント及び／または依存アプリケーションの追加／削除、ならびにその他の変更を含み得るが、これらに限定されない。ステップ1786で、コントローラ1701は、図14A～図16Fに関して説明した方法で環境をセットアップして、クローン環境1705となる（ここで、クローン環境は、それ自体の新しいコントローラを有してよい、または元の環境に対して同じコントローラを使用してもよい）。

【0278】

ステップ1787で、コントローラ1701は、変更管理ルール1711を含むグローバルルール210を使用して、システムの1つまたは複数の環境（例えば、「本番環境」）の全てまたは一部を、クローン環境1705（例えば、ここでクローン環境1705は、「開発環境」として機能してよい）にクローンしてよい。コントローラ1701は、バ

ックアップルール 2104 を使用してデータを抽出してよく、抽出されたデータは、本明細書で図 21A ～ 21J を参照して記載するように、後に、バックアップルールを用いて復旧されてよい。したがって、コントローラ 1701 は、リソースを識別して割り当て、システムルール 210 を使用して、クローンリソースをセットアップして割り当て、データ、構成、コード、実行ファイル、及びアプリケーションの起動に必要な他の情報のうちのいずれかを環境からクローンにコピーする。ステップ 1788 で、コントローラ 1701 は、任意選択で、システムルール 210 内の構成ルールを使用して（コントローラの有無に関わらず）バックアップ 1706 として機能する別の環境をセットアップすることによってシステムをバックアップし、テンプレート 230、コントローラロジック 205 及びグローバルルール 210 をコピーする。

10

【0279】

クローン 1705 が本番環境から作成された後、クローン 1705 は、クローンのコード、構成ルール、セキュリティパッチ、テンプレートへの変更及び他の変更を行うことができる開発環境として使用されてよい。ステップ 1789 で、開発環境への変更は、実装の前にテストされてよい。テスト中、クローン 1706 は、本番環境（システム 100）またはシステムの他のコンポーネントから分離することができる。これは、コントローラ 1701 に、システム 100 とクローン 1706 との間の接続のうち 1 つまたは複数を選択的に無効にさせることによって（例えば、インバンド管理接続 270 を無効にすること、及び／またはアプリケーションネットワーク接続を無効にすることによって）達成することができる。ステップ 1790 で、変更された開発環境の準備ができているかどうか判定される。ステップ 1709 で、開発環境がまだ準備できていないと判定された場合（これは、通常、開発者によって行われる決定である）、プロセスフローは、クローン環境 1705 に対するさらなる変更のためにステップ 1789 に戻る。ステップ 1790 で、開発環境が準備できていると判定された場合、ステップ 1791 で、開発環境と本番環境を切り替えることができる。すなわち、コントローラは、開発環境 1705 を新しい本番環境に変更し、開発／新しい本番環境への移行が完了し良好な状態になるまで、以前の本番環境を維持してよい。

20

【0280】

図 18A は、システムの変更管理においてセットアップ及び使用され得るシステム 100 の別の例示的な実施形態を示す。図 18A の例では、システム 100 は、コントローラ 1801 と、1 つまたは複数の環境 1802、1803、1804、1805 とを備える。システムは、クローン環境 1807 及びバックアップシステム 1808 と共に示されている。バックアップ及びデータ復旧は、本明細書の他の箇所に記載のバックアップルールを用いて実行することができる。バックアップシステムの管理の例は、図 21A ～ 21J を参照して本明細書にさらに記載する。

30

【0281】

コントローラ 1801 は、本明細書で説明されるコントローラ 200 / 1401 / 1501 / 1601 / 1701 と同様の方法で構成され、グローバルシステムルール 210、コントローラロジック 205、テンプレート 230 及びシステム状態 220 の要素を含み得る。コントローラ 1801 は、本明細書の図 14A ～ 図 16F を参照して説明したような方法で、1 つまたは複数の他のコントローラまたは環境に結合されてよい。コントローラ 1801 のグローバルルール 210 は、他のコントローラ及び／または環境を管理及び制御し得るルールを含んでよい。そのようなグローバルルール 210、コントローラロジック 205、システム状態 220、及びテンプレート 230 を使用して、本明細書の図 1 ～ 図 17B を参照して説明したのと同様の方法で、コントローラ 1801 を通して、システムまたは環境をセットアップ、プロビジョニング、及び配備してよい。各環境は、他の環境に関する動作を含む環境の動作を定義するグローバルルール 210 のサブセットを使用して構成されてよい。

40

【0282】

グローバルルール 210 は、変更管理ルール 1811 も含み得る。変更管理ルール 18

50

11は、システム、グローバルルール及び／またはロジックへの変更が望まれる場合に使用され得るルール及び／または命令のセットを含み得る。変更管理ルールは、ユーザまたは開発者が変更を開発し、テスト環境で変更をテストし、その後、変更をシステムルール210内の構成ルールの新しいセットに自動的に変換することによって変更を実装することを可能にするように構成されてよい。変更管理ルール1811は、(図18Aに示すように)グローバルシステムルール210のサブセットであってよい、またはグローバルシステムルール210とは別個であってよい。変更管理ルール1711は、グローバルシステムルール210のサブセットを使用してよい。例えば、グローバルシステムルール210は、新しい環境を作成するように構成された環境作成ルールのサブセットを含み得る。変更管理ルール1811は、システム100の一部または全ての態様をコピー及びクローンするために、コントローラ1801によってセットアップ及び配備されたシステムまたは環境をセットアップ及び使用するように構成されてよい。変更管理ルール1811は、テスト及び実装のためにシステムのクローンを使用することによって実装する前に、システムに対して提案された新しい変更のテストを許可するように構成されてよい。変更管理ルール1811は、本明細書の他の箇所に記載のバックアップルールを含み得る、または、使用してよい。バックアップルールは、バックアップルール2104を使用してデータを抽出してよく、抽出されたデータは、後に、本明細書の図21A～21Jを参照して記載したバックアップルールを使用して復旧されてよい。

【0283】

図18Aに示すように、クローン環境1807は、ルール、コントローラロジック、テンプレート、システム状態データを有するコントローラ1807aと、コントローラ1801のグローバルシステムルール210及び変更管理ルール1811に従って1つまたは複数の環境に割り当てられ、セットアップされ得る割り当てられたリソース1820とを含み得る。バックアップシステム1808はさらに、ルール、コントローラロジック、テンプレート、システム状態データを有するコントローラ1808aと、コントローラ1801のグローバルシステムルール210及び変更管理ルール1811に従って1つまたは複数の環境に割り当てられ、セットアップされ得る割り当てられたリソース1821を含み得る。システムは、ユーザ(またはアプリケーション)インタフェース110または別のユーザインタフェースに結合されてよい。

【0284】

クローン環境1807は、特定の環境またはシステムの一部のルール、ロジック、テンプレート、システム状態、アプリケーション及び／またはリソースを含み得る。クローン1807は、システム100と同様の、または異なるハードウェアを備えてよく、クローン1807は、仮想リソースを使用しても、使用しなくてもよい。クローン1807は、アプリケーションとしてセットアップされてよい。クローン1807は、環境のために、システム100またはコントローラ1801のシステムルール210内の構成ルールを使用してセットアップ及び構成されてよい。クローン1807は、コントローラを備えていても、備えていなくてもよく、本番環境とコントローラを共有してもよい。クローン1807は、上でより詳細に説明したように、割り当てられたネットワークリソース、計算リソース、アプリケーションネットワークリソース及び／またはデータストレージリソースを含み得る。このようなリソースは、コントローラ1801によって制御される変更管理ルール1811を使用して割り当てられてよい。クローン1807は、ユーザがクローン1807に変更を加えることを可能にするユーザインタフェースに結合されてよい。ユーザインタフェースは、システム100のユーザインタフェース110と同じであっても、異なってもよい。

【0285】

クローン1807は、システム全体、または1つまたは複数の環境及び／またはコントローラなどのシステムの一部に使用されてよい。例示的な実施形態では、クローン1807は、環境1802のデータリソース1820に結合されたホットスタンバイデータリソース1820aを含み得る。ホットスタンバイデータリソース1820aは、クローン1

10

20

30

40

50

807のセットアップ時及び変更のテスト時に使用されてよい。ホットスタンバイデータリソース1820aは、例えば、図18Bに関して本明細書に説明されるように、変更管理の間、ストレージリソース1820から選択的に切断可能であるか、または分離され得る。クローン1807は、システム100の完全なコピーであっても、そうでなくてもよい。クローン1807は、選択的に有効及び／または完全に無効にすることができ、及び／または単一方向の読み取り及び／または書き込み接続に変換されてよい、インバンド管理接続270、アウトオブバンド管理接続260及び／またはSAN接続280を介してシステム100に結合されてよい。したがって、クローン環境1807がテスト中に本番環境から分離されるとき、またはクローン環境が新しい本番環境としてオンラインになる準備ができるまで、クローン環境1807内の揮発性データへの接続を変更して、クローンデータを読み取り専用にしてよい。

10

【0286】

古い本番環境を新しい本番環境に切り替える場合、コントローラ1801は、フロントエンド、負荷分散装置または他のアプリケーションもしくはリソースに、新しい本番環境を指すように指示してよい。したがって、ユーザ、アプリケーションリソース及び／または他の接続は、変更が行われるときにリダイレクトされてよい。これは、例えば、ip/ipoi bアドレス、infiniband GUID、dnsサーバ、infinibandパーティション/open sm構成のリスト変更、または、ネットワークングリソースに命令を送信することによって達成され得るソフトウェア定義ネットワークング(SDN)構成の変更を含むがこれらに限定されない方法によって達成されてよい。フロントエンド、負荷分散装置または他のアプリケーション及び／またはリソースは、データベース、ミドルウェア及び／または他のバックエンドを含むがこれらに限定されないシステム、環境及び／または他のアプリケーションを指してよい。このような負荷分散装置を変更管理に使用して、古い本番環境から新しい環境に切り替えてよい。

20

【0287】

クローン1807及びバックアップ1808は、システムに対する変更の態様を管理する際にセットアップされ、使用されてよい。このような変更は、コード、構成ルール、セキュリティパッチ、テンプレートへの変更、ハードウェアの変更、コンポーネント及び／または依存アプリケーションの追加／削除ならびに他の変更を含み得るが、これらに限定されない。バックアップ1808は、システム全体、または1つまたは複数の環境及び／またはコントローラ1801などのシステムの一部に使用されてよい。バックアップ1808は、上でより詳細に説明したように、ネットワークングリソース、計算リソース、アプリケーションネットワーク及び／またはデータストレージリソースを含み得る。バックアップ1808は、コントローラを備えてもよく、備えなくてもよい。バックアップ1808は、システム100の完全なコピーであってもよい。バックアップ1808は、バックアップに含まれる構成ルールからシステム／環境／アプリケーションを再構築するために必要なデータを備えてよく、全てのアプリケーションデータを含み得る。バックアップ1808は、アプリケーションとして、またはシステム100と同様の、または異なるハードウェアを使用してセットアップされてよい。バックアップ1808は、選択的に有効及び／または無効にすることができ、及び／または一方向の読み取り及び／または書き込み接続に変換してよい、インバンド管理接続270、アウトオブバンド管理接続260及び／またはSAN接続280を介してシステム100に結合されてよい。

30

【0288】

図18Bは、変更管理における図18Aのシステムの使用を示す例示的なプロセスフローであり、特に、図18Aのシステムが揮発性データを含む場合、またはデータベースが書き込み可能な場合を示す。このようなデータベースは、システム内の環境によって使用されるストレージリソースの一部である可能性がある。ステップ1870で、システムは、グローバルシステムルールを使用して配備される(本番環境を含む)。

40

【0289】

次に、ステップ1871で、クローン環境がシステムへの書き込みを無効にされる読み

50

取り専用環境を作成するために、変更管理ルール 1811 を含むグローバルシステムルール 210 と、メインコントローラ 1801 またはクローン環境内のコントローラによるリソース割り当てとを使用して、本番環境がクローンされる。クローン環境はその後、開発環境として使用することができる。

【0290】

ステップ 1872 で、ホットスタンバイ 1820a が起動され、システム 100 内で変更される任意の揮発性データを記憶するためにクローン環境 1807 に割り当てられる。クローンデータが更新され、開発環境内の新しいバージョンを更新データでテストすることができる。ホット同期データは、いつでもオフにされてよい。例えば、古い環境または本番環境から開発環境への書き込みがテストされているとき、ホット同期データはオフにされてよい。

10

【0291】

次に、ステップ 1873 で、ユーザは、クローン環境 1807 を開発環境として使用して、変更を行うことができる。次に、ステップ 1874 で、開発環境に対する変更がテストされる。ステップ 1875 で、変更された開発環境の準備ができているかどうかが判定される（通常、このような判定は開発者によって行われる）。ステップ 1875 で、変更の準備ができていないと判定された場合、ユーザが戻って開発環境に他の変更を加えることができるように、プロセスフローはステップ 1873 に戻ってよい。ステップ 1875 で、変更を稼働させる準備ができていると判定された場合、プロセスフローはステップ 1876 に進み、ここで、特定の環境に関してシステムまたはコントローラ内で構成ルールが更新され、新しい更新された環境を配備するために使用される。

20

【0292】

ステップ 1877 で、次に、開発環境（または、新しい環境）が、稼働前に、所望のリソース及びハードウェア割り当てを有する所望の最終構成で変更されて再配備されてよい。次のステップ 1878 で、元の本番環境の書き込み機能が無効にされ、元の本番環境は、読み取り専用になる。元の本番環境が読み取り専用である一方、1878 の一部として、元の本番環境（または、おそらく、新しい本番環境も）からの新しいデータはいずれも、移行データとしてキャッシュ及び識別されてよい。一例として、データは、データベースサーバまたは他の適切な場所（例えば、共有環境）にキャッシュすることができる。次に、開発環境（または、新しい環境）と古い本番環境が、ステップ 1879 で切り替えられて、開発環境（または、新しい環境）が、本番環境になる。

30

【0293】

この切り替えの後、新しい本番環境は、ステップ 1880 で、書き込み可能にされる。ステップ 1881 で、新しい本番環境が開発者が判定したように機能しているとみなされると、切り替えプロセス中のデータ損失は全て（このようなデータはステップ 1878 でキャッシュされている）は、データが新しい環境に書き込まれて、ステップ 1884 で照合されてよい。このような照合の後、変更が完了する（ステップ 1885）。

【0294】

ステップ 1881 の結果、新しい本番環境が機能していない（例えば、システムを古いシステムに戻す必要がある問題が識別される）と判定された場合、ステップ 1882 で、環境は戻されて、古い本番環境が再び本番環境になる。ステップ 182 の一部として、コントローラ 1801 の対象環境の構成ルールは、現在戻されている本番環境に対して使用されていた以前のバージョンに戻される。

40

【0295】

ステップ 1883 で、データベースの変更が、例えば、キャッシュデータを用いて判定されてよく、データは、古い構成ルールを有する古い本番環境に復旧される。ステップ 1883 をサポートするために、データベースは、データベースに行われた変更のログを維持できるので、ステップ 1883 は、戻すことが必要な可能性のある変更を判定できる。キャッシュデータを追跡及び時間を記録するバックアップデータベースを使用して、上記のようにデータをキャッシュしてよく、クロックを戻して、どの変更を行ったかを判定し

50

てよい。スナップショット及びログをこの目的のために使用してよい。

【0296】

1883でキャッシュデータを復旧した後、再び、開始したい場合、プロセスはステップ1871に戻ってよい。

【0297】

本明細書で説明した変更管理システムの例は、例えば、ハードウェアまたはソフトウェアを更新、追加、もしくは、除去するとき、ソフトウェアにパッチを適用するとき、システム障害が検出されたとき、ハードウェア障害または検出中にホストを移行するとき、動的なリソース移行のために、構成ルールもしくはテンプレートの変更のために、及び／または、任意の他のシステムに関連する変更を行う際に、使用されてよい。コントローラ1801またはシステム100は、障害を検出するように構成されてよく、障害を検出すると、コントローラが利用可能なシステムの他のハードウェアに、変更管理ルールまたは既存の構成ルールを自動的に実装してよい。使用可能な障害検出方法の例は、ホストにpingすること、アプリケーションにクエリすること、及び、様々なテストもしくはテストスイートを実行することを含むが、これらに限定されない。本明細書に記載の変更管理構成ルールは、障害が検出されると、実施されてよい。このようなルールは、障害が検出されると、バックアップ環境の自動生成、コントローラが実施しているデータもしくはリソースの自動的な移行をトリガしてよい。バックアップリソースの選択は、リソースパラメータに基づいてよい。このようなリソースパラメータは、使用情報、速度、構成ルール、ならびにデータ容量及び使用量を含み得るが、これらに限定されない。

【0298】

本明細書に記載するように、変更が生じる時はいつも、コントローラは、変更のログと、実際に実行されたことを作成する。セキュリティまたはシステムの更新のために、本明細書に記載のコントローラは、構成ルールに従って自動的にオンオフし、且つ、ITシステム状態を更新するように構成されてよい。コントローラは、リソースをオフにして電力を節約してよい。コントローラは、時間によって効率を異ならせるためにリソースをオンにしてよい、または、移行してよい。移行時は、構成ルールに従い、環境またはシステムのバックアップまたはコピーが作成されてよい。セキュリティ侵害があった場合、コントローラは、攻撃されたエリアを切り離し、遮断してよい。

【0299】

サービス依存関係の構成及び制御

図19Aは、図1～18を参照して本明細書に記載した例示的なシステム100を示し、ここでは、システム100は、1つまたは複数のリソース1910上の対応するサービスモジュール1901、1902によって示される相互に関連するサービス（または、アプリケーション）を用いて強化されている。サービスモジュール1901、1902は、認証、電子メール、ウェブメール、ウェブサービス、ミドルウェア、データベース、及び／または他のサービス等のサービスを提供するコンピュータ実行可能コードの形態をとることができる。それぞれに言及するとき、サービス1901は、サービスAと呼ぶことができ、サービス1902は、サービスBと呼ぶことができる。

【0300】

図19Aのシステムは、外部ネットワーク1980及び／またはアプリケーションネットワーク390に接続されてよく、この接続は、図13A～13Eに記載の説明に従って無効及び有効にされてよい。サービス1901、1902は、図1～18を参照して、本明細書の様々な実施形態で記載されたリソースまたはアプリケーションとして、コントローラ200によって構成される。

【0301】

サービス1901、1902は、コントローラ200によって制御することができ、共通API1903を通して相互運用してもよい。サービス1901、1902は、共通API1903を使用して、依存関係を解決することができる。例えば、ウェブアプリケーションがhttpサーバを必要としているとする。apacheまたはnginxを有す

10

20

30

40

50

るサービスが、「ウェブサーバ共通API」を有してよく、「ウェブサーバ共通API」は、そのサーバにwebappのコンテンツを出させ、情報をアプリケーションにプロキシして戻してよい。サービス1901、1902及びAPI1903は、管理ネットワークを通して直接、または、コントローラ200への管理接続（例えば、260及び／または270）、もしくはコントローラ200、サービス1901、共通API1903、及びサービス1902間の任意の他のネットワーク接続を通して結合されてよい。

【0302】

共通API1903は、サービス1901、1902、コントローラ200、またはシステム100の他のリソースのうちの1つまたは複数上で実行してよい、または、それらに応答してよい。サービスモジュール1901のサービスAは、サービスモジュール1902上の依存サービスBによって共通API1903を通して呼び出されて、1つまたは複数の機能を実行するように構成された被依存サービスであってよい。被依存サービスは、他のサービスの依存関係を満足することができるサービスである（この場合、他のサービスは、「依存サービス」である）。被依存サービスはまた、任意選択の被依存サービスであってよい。

【0303】

サービス1901、1902は、互いに安全に相互運用するようにコントローラ200によって構成またはセットアップされてよい。

【0304】

図19Aのサービス及びコントローラの相互運用の例は、図19Bのフローを参照して説明される。サービスは、例えば、コントローラ200が本明細書に記載の構成ルールを使用して開始されてよい（19.1を参照）。コントローラ200は、本明細書の図に記載の依存関係を解決する（19.2参照）。サービスは、その仕様に列挙された依存関係のセットを有し得る。一例として、これは、サービスのjson仕様を用いて行うことができる。システムはさらに、パッケージマネージャが機能する方法と類似の依存関係解決を使用し、依存関係を満足させる方法をユーザに提供し得る。別の例として、システムは、ユーザに被依存サービスのインストール、または、既存の被依存サービスの使用／選択を提案することができる。依存サービスBは、共通API1903を通して被依存サービスAを呼び出す（19.3を参照）。この呼び出しは、サービスAがサービスBをサポートするように構成するための呼び出し、または、サービスAの機能の一部を使用するための呼び出しであってよい。共通API1903は、被依存サービスA（1901）に変換し、被依存サービスAにコマンド（複数可）を実行するように指示する（19.4を参照）。変換は、サービスの1つまたはコントローラ200がAPI呼び出しを行うことによって行うことができる（及び、ここで、APIの機能は、異なるAPI上の他のAPI機能と呼ばれ出すことであってよい）。

【0305】

本明細書に記載のいくつかの例示的な実施形態によると、コントローラ200と、被依存サービス1901及び依存サービス1902とを有するシステムに追加のセキュリティを備えることができる。この追加のセキュリティは、複数のサービスが同時にインバンド管理接続270に接続され、互いに直接通信し得るときに有用である。このような追加のセキュリティは、コントローラのグローバルシステムルール210、ロジック205、テンプレート230、及び／またはシステム状態220に基づいて、及び／または、それらを使用して、構成、再構成、及び／または操作中に提供されてよい。追加のセキュリティは、サービス1901、1902がインバンド管理接続270、または他のネットワークもしくは相互接続によって通信しているいくつかの例示的な実施形態で提供されてよい。いくつかの例示的な実施形態によると、被依存サービス1901は、依存サービス1902の妥当性確認をコントローラに求めるように構成される。これは、依存サービス1902の身元を検証すること、または、API上でコマンドを実行するサービスの身元を検証することを含み得る。いくつかの例示的な実施形態によると、被依存サービスは、自身の機能、タスク、及び／または、複数のタスク、機能、またはそれらの組み合わせを特定の

10

20

30

40

50

依存サービス（複数可）のために実行する許可をコントローラ 200 に求めるように構成される。被依存サービスは、さらにまたは代替的に、被依存サービスが構成または再構成されるとき、コントローラ 200 によって許可、または許可のセットをプロビジョニングされてよい。被依存サービスの許可セットも更新されてよい。例えば、許可セットは、依存サービスが追加されるとき、更新されてよい。

【0306】

サービス間で提供される認証及び許可の例は、図 19C に示されるフローで説明される。ステップ 19.11 で、コントローラ 200 は、構成中にサービスの鍵または鍵のペアをプロビジョニングし、それによって、サービスとコントローラ 200 の間の認証を可能にする。このステップは、システムの各サービスに対して実行することができる。被依存サービスによる実行を要求する依存サービスが、被依存サービスがコントローラから妥当性確認を要求することによって妥当性を確認される。サービスの身元と、サービスに送受信されるデータとが、例えば、相互 TLS 認証、公開鍵認証、他の形態の暗号、任意のネットワークベースの妥当性確認技術（*v l a n s*、*v x l a n s*、パーティション等を含むがこれらに限定されない）、及び／または、これらの組み合わせによって、妥当性を確認されてよい。仮想のネットワーク及びパーティションを使用して、ネットワークを *I n f i n i b a n d* パーティションなどのミニネットワークに分割することができる。この結果、ポートが、例えば、パーティション 4 及び 15 上にある場合、パーティション 4 及び 15 上の物にだけ通信することができるというシナリオが可能になる。いくつかの変形形態によると、コントローラ 200 は、コントローラ 200 から独立したサービスモジュール 1901、1902 内の検証または認証を維持しながら、鍵配布センタとして使用することができる。サービスの構成中、コントローラは、サービスとコントローラ 200 との間の認証を直接または間接に可能にする公開鍵及び／またはプライベート鍵を含み得る鍵または鍵のペアをそのサービスにプロビジョニングしてよい。一例によると、コントローラ 200 は、公開鍵を保存し、プロビジョニングされたプライベート鍵を削除または無効にすることができる。追加の例として、サービスは、サービスからの公開鍵がコントローラ 200 によって識別、認識、及び／または認証できるように、サービス自体の鍵を生成してよい。この追加の例においては、コントローラ 200 がサービスの最初の公開鍵を提供したので、サービスは、現在のプライベート鍵の知識を有さないコントローラ 200 に信頼できる公開鍵を送る能力を有し、サービスは、サービスの元の鍵のペアで認証して、新しい公開鍵をコントローラ 200 と共有する。

【0307】

ステップ 19.12 で、依存サービス（サービス B）は、共通 API 1903 を通して被依存サービス（サービス A）に機能を実行するように呼び出す。被依存サービス（サービス A）は次に、コントローラ 200 によって依存サービス（サービス B）を認証する。一例によると、被依存サービスは、コントローラ 200 に連絡してよく、コントローラ 200 は、要求を行っている依存サービスを依存サービスが提供した公開鍵を使用して認証することができる。上記のように、依存サービスは、ステップ 19.11 で公開鍵を取得することができる。依存サービスはまた、新しい鍵のペア（公開鍵＋プライベート鍵）を生成してよく、（コントローラ 200 が公開鍵及びプライベート鍵を作成したとすると）コントローラ 200 が古い公開鍵を信頼することが分かっているので、古い鍵のペアを用いて、その新しい公開鍵が本物であることを証明してよい。依存サービス（サービス B）は、被依存サービスを類似の方法で認証してよい（19.13）。

【0308】

ステップ 19.14 で、被依存サービス（サービス A）はまた、依存サービス（サービス B）に対して機能を実行する許可を、機能を実行する前に確立してよい。例えば、許可は、コントローラ 200 に許可が可能かどうかを尋ねることによって、被依存サービスによって確立されてよい。別の例として、許可は、コントローラ 200 によって被依存サービス（サービス A）上にロードされた許可リストを通して確立されてよい。

【0309】

図19Dは、相互運用サービスと共に使用される強化されたセキュリティ方法の一例を示す。ステップ19.21で、依存サービスBが、例えば、本明細書に記載のコントローラ200及び／またはテンプレート230を用いて作成されてよい。ステップ19.22で、依存サービスBが実行されるとき、コントローラ200は、本明細書の様々な実施形態（例えば、図13A～13Eを参照）に関して説明したように外部ネットワーク1910及び／またはアプリケーションネットワーク390への接続を検証及び／または無効にしてよい。例えば、実施者は、サービスがインターネット等のネットワークにオープンである間、管理接続を無効にすることを望ましいとする場合がある。これは、前述のように、さらなる分離とセキュリティを提供する。このような場合、クラウドAPI（または、他の場合には、アウトオブバンド管理接続260）を使用して、インバンド管理接続270を切り替えることができる。ステップ19.23で、依存サービスBは、APIコマンドを実行して、被依存サービスAからのサービスまたは機能を要求する。このステップはまた、依存サービスBがコントローラ200に依頼することと、コントローラ200が共通APIを通してコマンドを実行することによって完了されてよい（1903参照）。ステップ19.24で、被依存サービスAは、依存サービスBの身元と、サービスBのためにサービスまたは機能を実行する許可との妥当性を確認する。例として、このステップ19.24は、サービスAがサービスBの許可の妥当性を確認することによって実行することができる。別の例として、このステップ19.24は、サービスAがサービスBにサービスを提供する許可を有することをサービスAが確認することによって、実行することができる。いずれの場合でも、他のサービスによって修正されているサービスは、他のサービスがそれらの修正を行うことを許可されていることを確認することができる。認証及び許可されている場合、被依存サービスAは、コマンドで指定されたサービスまたは機能を実行してよい（19.25を参照）。ステップ19.26で、アウトオブバンド管理260、インバンド管理270、またはSAN280などの管理接続は、任意選択で、図13A～13Eを参照して本明細書で説明したように追加のセキュリティのために切断されてよい。ステップ19.27で、外部ネットワーク（複数可）1980及び／またはアプリケーションネットワーク（複数可）390への接続は、接続が19.22で無効にされる場合、その後、再度有効にされてよい。

【0310】

図19Eは、図19A～19Dに関して記載したシステム等の例示的なシステム100を示し、クリーンアップルール1904のセットが、コントローラ200に含まれている。クリーンアップルール1904は、コントローラ200内にそれ自体のルールのセットとして具現化されてよい、または、グローバルシステムルール210、コントローラロジック205、テンプレート230、もしくはこれらの組み合わせで具現化されてよい。クリーンアップルール1904は、サービスが削除されるとき、従うべき命令及びルールのセットを含む。例として、クリーンアップルール204は、サービスのセットアップに使用されるテンプレート230に含まれてよく、その場合、サービスに関連するルールが、セットアップ中にサービスにロードされる、または、サービスに関連するルールを使用して、サービスに固有のクリーンアップルールを生成する。例えば、メールサービスによって、dns記録がdnsサービスに追加されてよい。そのメールサービスが削除される場合、dnsサービスは、dns記録をそのメールサービスから取り除くことができる。

【0311】

クリーンアップルール1904を使用して、依存サービスによって被依存サービスに行われた修正を識別し、依存サービスが削除または無効にされるとき、これらの修正の削除、除去、及び／またはアンドゥを有効にしてよい。図19Fは、クリーンアップルールを作成するための例示的なプロセスフローを示す。例えば、図19Fは、依存サービスが削除されるとき、修正が、除去するために、どのように、識別、例えば、ログ及び／または追跡され得るかを示す。

【0312】

図19Fに示すように、ステップ19.31で、依存サービス（サービスB）からAP

10

20

30

40

50

I 1903に、機能を実行する被依存サービス（サービスA）を呼び出すコマンドが発せられる。ステップ19.32で、依存サービス（サービスB）の妥当性が確認され、図19A～19Dで説明したように許可が確認される。機能を実行する際に、被依存サービス（サービスA）に対して修正が行われる、または、行われるべき場合、ステップ19.33で、対応するクリーンアップルール（複数可）、及び／またはAPIコマンドに対応するクリーンアップコマンド（複数可）が、依存サービス（サービスB）、被依存サービス（サービスA）、またはコントローラ200のうちの1つまたは複数に検索可能に追加される（または、クリーンアップルール）に関連付けられる。クリーンアップルールまたはクリーンアップコマンドは、依存サービス（サービスA）の削除時に、次のクリーンアップのための修正を識別することができる。様々な例示的な実施形態によると、依存サービスまたは被依存サービスは、関連付けられたクリーンアップルールを有してよい。クリーンアップルールはまた、サービス間の接続を修正するように構成されてよい。被依存サービスは、被依存サービスが関係を有する各依存サービスに対応し得るクリーンアップルールを有してよい。クリーンアップルールはまた、ログされたAPIコマンドから生成されてよい。個々のクリーンアップステップは、APIコマンドのログから導出されるときに1つずつ実行することができる。

【0313】

クリーンアップルールは、依存サービスが削除、変更、及び／または修正されるとき、使用することができる。クリーンアップルールは、依存サービスが被依存サービス上に変更を行ったときも、使用することができる。図19Gの例示的なプロセスフローに示すように、ステップ19.41で、サービスを削除する判定が行われる。ステップ19.42で、コントローラロジック205は、削除されるサービス（「削除サービス」）の依存サービスをチェックする。これらの依存関係は、再帰的であってよく、再帰的依存関係解決を通して発見されてよい。依存サービスは、本明細書の図2A～2Kに関して説明するように、依存サービスのサービステンプレート230で、または、システム状態220もしくは他の関連するデータベースで識別されてよい。ステップ19.43で依存サービスがあると結論付ける場合、コントローラは、（例えば、図2Kに記載のように）依存関係を満足させる代替りの方法を見つけてよい（19.44を参照）。コントローラ200が依存関係を満足させる代替りの方法を識別しない場合（19.45を参照）、ユーザ／管理者は、例えば、新しい被依存サービスを追加することによって、サービス削除をキャンセルすることによって、または、類似の方法で依存サービスを削除することによって、解決するように通知されてよい（19.46参照）。コントローラが、ステップ19.44で、依存関係を満足させる代替りの方法を識別する場合、コントローラ200は、依存関係を変更してよく、対応するコントローラコンポーネント（テンプレート230、ルール210、ロジック205、システム状態220等）と、依存サービス及び／または被依存サービスのコンポーネントとを更新及び／または再構成してよい。次に、ステップ19.47で、クリーンアップルールに従ってよい。ステップ19.43の結論として、依存サービスが無い場合、プロセスフローはまた、ステップ19.47に進んでよく、ステップ19.47で、クリーンアップルールに従ってよい。上記のように、クリーンアップルールは、依存サービスによって被依存サービスに行われた修正を識別する。従って、ステップ19.47で、これらのクリーンアップルールは、これらの修正の削除、除去、及び／またはアンドゥを有効にするように処理することができる。ステップ19.47の一部として、削除サービスによって使用中に作成されたファイルが、被依存サービスから取り除かれる。

【0314】

ストレージリソースを計算リソースにプロビジョニング

図20Aは、1つまたは複数のストレージリソース410のストレージを利用する1つまたは複数のサービスをホストする1つまたは複数の計算リソース310を有するコントローラ200を含む例示的なシステムを示す。図21Aは、物理サービスホストが、SAN280と通信するのを許可されている場合、このような物理サービスホストのみが、使

用を承認されているリモートストレージに接触することが望まれることをさらに示す。従って、悪意の当事者 2002 がストレージリソース 410 等のシステムリソースに不正アクセスするのをシステムによって防ぐことが望ましい（図 20A 参照）。

【0315】

図 20B のプロセスフローに記載される例によると、コントローラ 200 は、ストレージ認証情報を計算リソースにプロビジョニングする（20.10 参照）。例として、ストレージ認証情報は、パスワード、パスフレーズ、チャレンジハンドシェイク認証プロトコル（CHAP）鍵、暗号鍵、証明書、または、これらの組み合わせの形態をとってよいが、これらに限らない。CHAP は、iSCSI / iSER 等のリモートストレージなどの認証技術である。CHAP 鍵は、SAN のパスワードとして使用することができる。ステップ 20.10 のプロビジョニングは、多くの方法のうちのいずれかで進めることができる。例えば、サービスイメージを作成するとき、コントローラ 200 は、サービスイメージを有するストレージリソース接続情報を含み得る。計算リソースもコントローラ 200 にクエリすることができる。別の方法は、電源オンのプロセス中、コントローラ 200 が計算リソースに情報を提供することができる（これは、オンデマンドで作成される場合、ストレージリソースが作成 / プロビジョニングされた後、行うことができる）。全ての情報は、データベースに配置することができ、コントローラ 200 は、ストレージ認証情報をデータベースから引き出すことができる、または、計算リソースが、データベースクエリ、またはデータベースクエリを行う API 呼び出しを実行することによって、コントローラにストレージ認証情報を求めることができる。計算リソースは次に、ストレージ認証情報を用いてストレージリソースに接続、ログオン、または通信する（20.11 参照）。

【0316】

図 20C のプロセスフローに記載される例によると、代替でまたは追加で、計算リソース 310 とストレージリソース 410 の間の SAN 接続 280 は、本明細書の様々な実施形態で説明するように、無効にされてよい（20.20 を参照）。次に、ストレージリソース 410 は、v l a n s、v x l a n s、及び I n f i n i b a n d パーティションを含むが、これらに限らない特定の分離された接続ネットワーク上の計算リソース 310 に利用可能にすることができる（20.21 参照）。したがって、コントローラ 200 は、計算リソースをストレージリソースとペアにしてよく、それらを同じネットワークまたはファブリック上に配置してよい。例として、ポートはパーティションに割り当てることができる、または、2 つのポート間でトラフィックが通過するのを可能にするようにスイッチに伝えることができる。これは、上記のように、v l a n s、v x l a n s、及び I n f i n i b a n d パーティションを用いて行うことができる。さらに、コントローラ 200 はまた、ネットワークカードのそれぞれに鍵を与えることと、データ暗号化の F P G A を含む M e l l a n o x I n n o v a - 2 等の高性能の暗号化 / 復号能力を有するカードにデータを暗号化させることによって、高速の暗号化通信を可能にし得る。

【0317】

図 20D の例示的なプロセスフローに記載する例によると、代替的にまたは追加で、ネットワークングハードウェアは、暗号鍵を与えられてよく、送受信の前に、ワンタイムパッド（または、他のストリーム暗号）及び x o r バイトを作成し得る（20.30 を参照）。次に、これらの暗号化技術を使用して、計算リソースがストレージリソースにログオンするためのログイン認証情報を計算してよい（ステップ 20.31）。したがって、データは暗号化され、SAN にアクセスを試みるあらゆるリソースの身元がコントローラ 200 によって検証され、コントローラ 200 は、信頼できる公開鍵が分かっているので、データは、SAN 上で安全な状態を保つことができる。

【0318】

さらに、高速のデータを望むシステムに対しては、暗号化 / 復号をソフトウェアを用いて実行することができる一方、ネットワークカードを採用して、ハードウェアで暗号化 / 復号を実行することができる。コントローラ、ストレージリソース、及び / または計算リソースは、鍵または他の暗号化機構で互いに合意することができ、v e r i l o g コード

等のFPGAコードを使用して、ストレージリソース及び計算リソース上の各ネットワークカードを高速のハードウェア暗号化を用いて通信させることができる。例えば、verilogコンパイラは、特定の暗号の使用をコンパイルすることができる。FPGA以外に高速ストレージ間のデータを暗号化する他の方法もある。例えば、Intel Omni pathを使用して2つのノード間のデータを暗号化することができる。コントローラ200は、安全な送信を可能にする鍵をリソースに与え、承認されたリソースのみをリモートストレージのデータにアクセスさせるようにすることができる。

【0319】

バックアップルールのプロビジョニング及び使用

図21A-21Jは、サービスと共にバックアップルールを使用するシステム及び方法の例を示す。

【0320】

図21Aは、バックアップルール2104のセットがさらにコントローラ200に含まれる、図19A~19Dに関して説明したシステム等の例示的なシステム100を示す。バックアップルール2104は、コントローラ200内にそれ自体のルールのセットとして具現化されてよい、または、グローバルシステムルール210、コントローラロジック205、テンプレート230、もしくは、それらの組み合わせ、1つまたは複数のサービス、被依存サービス、及び／または依存サービスのサービステンプレートの一部として、具現化されてよい。さらに、サービス1901、1902のテンプレートは、バックアップルール2104のセットを含み得る。

【0321】

システム100が互いに依存関係を有する複数のサービスをサポートする例示的な実施形態では、バックアップルール2104は、様々なサービスのセットに関連付けられた様々なバックアップルールのセットの仕様を含み得る。したがって、バックアップルール2104は、各サービス（または、サービスのクラス）に個々に合わせることができる。サービスに関連するバックアップルールの使用によって、システム100は、サービス依存関係を通して効率的に追跡する自動バックアップ動作を柔軟にサポートすることができ、特定のサービスのデータだけではなく、特定のサービスの特定の依存サービスのデータの少なくとも一部もバックアップされる。特定のサービスだけではなく、その依存サービスに対するデータのこのアーカイブは、バックアップから復旧する必要がある場合、特定のサービスのより信頼できる自動回復を提供する。相互依存サービス及びバックアップ動作間の連携の例は、図21Hを参照して以下に説明する。

【0322】

また、バックアップルール2104は、サービス及びサービスに関連するデータの復旧方法に関して命令（または、命令へのポインタ）としての機能を果たす復旧情報を含む（または、指す）ことができる（例えば、図21Bの2106参照）。この復旧情報はまた、復旧の一部として抽出されるバックアップデータの位置も識別してよい。

【0323】

さらに、コントローラ200は、バックアップルール2104、コントローラロジック205、グローバルシステムルール210、及び／またはこれらの組み合わせを使用して、ストレージリソースのバックアップ動作への最適のプロビジョニング方法を決定することができる。例えば、バックアップルール2104は、バックアップデータのアーカイブに使用するストレージ空間を指定することができ、次に、コントローラ200は、バックアップデータのアーカイブに使用するシステムのストレージリソースの1つまたは複数内に適切なストレージ空間をプロビジョニングすることができる。この一部として、コントローラ200は、対象サービス（複数可）がプロビジョニングされたストレージ空間／リソースにアクセスするための認証情報を対象サービス（複数可）に提供することができる。

【0324】

バックアップルール2104は、システムまたはシステムの任意の部分のバックアップがある場合はいつでも使用し得るルール及び／または命令のセットを含み得る。例えば、

本明細書に記載のように、バックアップルール 2104 は、変更管理と共に使用されてよい。バックアップルール 2104 は、周期的またはルーチンのシステムバックアップに使用されてよい。バックアップルール 2104 は、サービス及び／またはサービスの依存関係をバックアップするとき、例えば、サービスを削除、更新、または他の方法で変更する場合、使用されてよい。バックアップルール 2104 は、データまたは他の情報を保護するために、任意の他のユーザが開始したプロセス、または自動プロセスに使用されてよい。バックアップルール 2104 は、グローバルシステムルール 210、サービステンプレート、サービスイメージの一部であってよい、及び／または、テンプレート 230 を使用して、リソース 1910、例えば、依存サービス及び被依存サービス（例えば、1901、1902）上にロードされてよい。

10

【0325】

図 21B は、バックアップルール 2104 の例を示す。バックアップルール 2104 は、バックアップ命令 2105 を含み得る。バックアップ命令 2105 は、例えば、プログラム、スクリプト、またはロジックが、サービス内で、またはコントローラ 200 から実行し得る命令の形態を取ってよい。バックアップ命令 2105 は、任意のデータ、サービス、環境、またはシステム 100 の任意の部分をバックアップするバックアッププロセスを実行するプログラム、スクリプト、またはロジックであってよい。

【0326】

バックアップルール 2104 は、復旧命令 2106 を含み得る。復旧命令 2106 は、プログラム、スクリプト、またはロジックが、任意のデータ、サービス、環境、またはシステム 100 の任意の部分を復旧するために、サービス内またはコントローラから実行し得る命令を含み得る。サービスをバックアップするとき、バックアップルール 2104 は、関連する被依存サービス、または、関連する被依存サービス内のもしくはそれに対応するデータのバックアップも指示してよい。復旧命令 2106 は、バックアッププロセスの復旧機能を実行するプログラム、スクリプト、またはロジックであってよい。

20

【0327】

バックアップルール 2104 は、戦略 2107 を含むことができ、戦略 2107 は、サービスに対応する包括的なバックアップルール 2104 から選択的に呼び出すことができるバックアップルールのサブセットの形態を取ることができる。例えば、戦略 2107 は、（ルーチンを定義し得る）選択されたグループのバックアップルールであってよい。戦略 2107 は、被依存サービスからのバックアップ方法またはバックアップデータを記述する方法、プログラム、スクリプト、ロジックを含み得る。戦略 2107 は、テンプレート、決定木、または特定のシステムまたはプログラムに関連する変数に対応する特徴に依存し得る他のプログラム、スクリプト、ロジック、もしくはユーザに基づいて、選択または呼び出すことができる。

30

【0328】

バックアップルール 2104 は、ストレージリソース情報 2108 を含み得る。ストレージリソース情報 210 は、例えば、位置、タイプ、識別、及び／または他のサービスへの関係を含む情報を識別することができる。

【0329】

データバックアップ／スタンドバイルーチンは、任意選択で、個々のサービスのバックアップルールから指示されたバックアップルーチンを実行してよい。これらのバックアップルールは、各サービスをバックアップするための具体的に指示された方法を用いて必要なデータを集める（例えば、サービスが、postgres データベースを有する場合、サービスは、postgresql バックアップルーチンを実行してよい）。これらのバックアップルールは、グローバルシステムルール 210 に含まれるバックアップルーチン／ルールも呼び出してよい。

40

【0330】

図 21C-1～21C-3 は、バックアップすることができるデータの例を示す。バックアップ動作の一部として、システムは、バックアップするデータをシステムのストレ

50

ジリソース内にコピーすることができる。バックアップルールで指定されるサービス（または、実行コードである任意のエンティティ）がデータをコピーする、またはデータをコピーする関数を呼び出す命令を実行することができる。バックアップデータはまた、複数のサービスからのデータを含み得る（または、様々なストレージリソースへのポイントの集合を含み得る）。システムはまた、データが記憶されたとき及びバックアップが行われたときを追跡し、システム状態 220 には、バックアップデータを指す情報（例えば、データベースエントリ）が存在してよく、バックアップが「進行中」か「完了」かを示すフィールドが存在してよく、偽の「完了」が無いように、また、データの完全性の検証をサポートするためにデータベースにデータのハッシュも有してよい。

【0331】

未加工のバックアップデータに加えて、バックアップデータ 2110 は、バックアップデータの識別子として機能する ID 2111 を含むことができ、この識別子は、一意の識別子の形態を取ることができる。バックアップデータ 2110 は、関連するデータ、バックアップ、またはバックアップデータへの関連付けまたは結合 2112 も含み得る。関連付け／結合 2112 の例示的なシナリオは、サービス A、B 及び C のバックアップがあつてよく、A は、依存サービスであるが、B 及び C のバックアップを必要とし、それによって、システムは B 及び C のバックアップを指すことができ、その結果、誰かが A の復旧を試みており、バックを試みており、B 及び C の復旧、もしくは、とにかくデータの取得を必要とする場合、データ復旧のために実行するルールは、関連付け／結合 2112 によって定義される連携を介してどこに進むべきかが分かる。また、A、B 及び C のバックアップは一緒に記憶することができる。

【0332】

バックアップデータ 2110 は、他のストレージリソース、フォルダ、ディレクトリ構造への結合または関連付け 2113 も含み得る。結合／関連付け 2113 は、実施者は、被依存サービスのデータを別個の場所（例えば、blob／ファイル／フォルダ／tarball し、または、アーカイブもしくは Zip／またはサブフォルダ）に置きたい場合があるので、バックアップデータがどのように他のバックアップデータに接続／関連付けることができるかに関する連携を提供する。結合／関連付け 2113 は、「このファイル共有に接続、これがパスワード」と効果的に示す命令、及びストレージ機構への関連するアクセス認証情報も含み得る。時には、被依存サービスのデータは、被依存サービスに対してバックアップルールを実行することによってバックアップされてよく、バックアップデータは、それ自体の別個のエンティティであるが、システムが復旧中のとき、システムは、そのデータに到達する必要がある（且つ、そのデータは、別のストレージリソース上に存在する場合がある）。この状況では、結合／関連付けは、必要なデータを取得するためにそのストレージリソースへの到達方法に関する命令を含み得る。

【0333】

バックアップデータ 2110 は、バックアップメタデータ 2114 も含むことができ、このようなバックアップメタデータ 2114 の例は、図 21C-2 に示される。バックアップメタデータ 2114 は、データに対応する ID 2111、バックアップが開始及び／または終了した時間、データのバックアップに使用された方法（例えば、戦略またはルール）、及び／または適切な復旧ルール（または、復旧ルールへのポイント）等の情報を含み得る。復旧ルールは、バックアップルールの一部として記憶することができ、バックアップルール自体は、「使用した方法 2116」、または他の復旧情報 2115 もしくは復旧プロセスをガイドするのに使用され得る情報の一部として記憶されてよい。バックアップメタデータ 2114 は、サービス名、サービスタイプ、または他のサービス識別子も含み得る。他の情報、例えば、結合、関連するバックアップ、ストレージリソース位置、ポイント、戦略、及び方法はバックアップメタデータ 2114 に含むことができる。バックアップメタデータ 2114 も ID または一意の識別子と共に、システム状態 220 に記憶することができる（図 21C-3 を参照）。

【0334】

図 2 1 C - 4 は、復旧情報 2 1 1 5 の例を示す。復旧情報 2 1 1 5 は、復旧ルール、または復旧命令 2 1 0 6 へのポインタを含み得る。復旧情報は、コントローラ 2 0 0、グローバルシステムルール 2 1 0、システム状態 2 2 0、及び／またはこれらの組み合わせから生じ得る関連するシステム情報 2 1 1 8 も含み得る。復旧情報 2 1 1 5 は、この特定のバックアッププロセスからの特定のバックアップデータ 2 1 1 0 からの復旧を支援することができる情報 2 1 1 9 も含み得る。これは、被依存サービスの位置に関する情報 2 1 2 0、または被依存サービスに関する情報についてコントローラ 2 0 0、システム状態 2 2 0、もしくはグローバルシステムルール 2 1 0 へのクエリの仕方に関する情報を含み得る（システム状態 2 2 0 またはシステムルール 2 1 0 が変わった場合、または、復旧時にシステム 1 0 0 への既知の変更がある場合）（2 1 2 1 を参照）。リソース入手可能性、ストレージリソース位置変更等のシステム状態 2 2 0 の他の情報 2 1 2 2 も含み得る。予測されるファイルサイズ、バックアップデータのハッシュ、及びシステム状態 2 2 0、システムルール 2 1 0、または他のコントローラコンポーネントに記憶されていない場合がある任意の他の情報等、復旧情報に記憶された被依存サービスに関連していない他の情報も任意選択で存在してよい。

【 0 3 3 5 】

図 2 1 D は、例えば、図 2 1 E に示すように、バックアッププロセス中、コントローラ 2 0 0、ストレージリソース 1 9 1 0 a、及びサービス 1 9 0 1 a（依存、または被依存）の関係を示す。図 2 1 E のプロセスフローは、バックアッププロセスの開始で始まる（2 1 . 1）。ステップ 2 1 . 1 でバックアップルールを利用するプロセスが、ユーザまたは自動プロセスによって開始されてよい。例えば、ユーザが、バックアップルーチンまたはスタンドバイルーチンを要求する関数を実行するようにシステムに要求してよい（例えば、ステップ 2 1 0 . 7 の図 2 M を参照）、または、コントローラ 2 0 0 が、システムルール 2 1 0、テンプレート 2 3 0、またはコントローラロジック 2 0 5 に従って、バックアップルーチンまたはスタンドバイルーチンを要求するタスクを実行してよい。コントローラ 2 0 0 は、バックアップルーチンを呼び出すサービスまたは他のプログラムにバックアップ要求を送信する（図 2 1 E のステップ 2 1 . 2 を参照）。コントローラ 2 0 0 は、任意選択で、ストレージリソースをプロビジョニングすることができる（2 1 . 3 を参照）。ステップ 2 1 . 4 で、サービスは、バックアップルールに従って、バックアッププロセスを開始する。ステップ 2 1 . 5 で、バックアップされるデータが、コントローラ 2 0 0 または指定されたストレージリソースに送信される。バックアップルールは、バックアップするデータをディレクトリ、ボリューム、または他のタイプのストレージリソースに移動または配置するように（コントローラロジック 2 0 5 または他のストレージプロセスを用いて）コントローラ 2 0 0 に指示する（2 1 . 6 を参照）。次に、適切な情報が、システム状態 2 2 0 にログされる（2 1 . 7 を参照）。例えば、このような情報は、バックアップメタデータ 2 1 1 4、適切なストレージリソースにアクセスするための情報（例えば、図 2 1 C - 1 の 2 1 1 3 を参照）、バックアップ ID 2 1 1 1、または任意の他の識別情報を含み得る。あるいは、ステップ 2 1 . 1 でバックアッププロセスを開始した後、コントローラ 2 0 0 は、必要なストレージリソースに結合してよい（2 1 . 8 を参照）。コントローラ 2 0 0 は、次に、バックアップルール 2 1 0 4 からバックアップルーチンを実行する（2 1 . 9 を参照）。システム状態 2 0 0 は次に更新される（2 1 . 1 0 を参照）。

【 0 3 3 6 】

図 2 1 F は、バックアップデータ、構成、及び他の情報の復旧を記載する例示的なプロセスフローである。ステップ 2 1 . 2 0 で、復旧が要求される。具体的に識別されたバックアップデータ、構成、及び他の情報が、システム状態 2 2 0 から見つけれられる（2 1 . 2 1 を参照）。データ、構成、または他の情報を含むストレージリソース（複数可）が、復旧サービスに、または関連するストレージリソースに結合される（2 1 . 2 2 を参照）。例えば、このような他の情報は、バックアップデータ 2 1 1 0 にはない情報を含むことができ、システム状態 2 2 0 またはシステムルール 2 1 0 は、変更されている可能性があ

り、その場合、構成及び／または他の情報は、例えば、「システム状態でこの事を見る」と効果的に示すことによって、復旧プロセスに知らせることができる。復旧プロセスがステップ 21.23 で実行される。この実行の完了時、ストレージリソースへのサービスまたはデータ復旧が、完了されたと報告される（21.24 を参照）。システム状態 220 は次に更新される（21.25 参照）。

【0337】

図 21G は、サービスが破損され、削除または訂正の必要がある状況において、バックアップルール 2104 の使用を記載する例示的なプロセスフローを示す。この状況では、プロセスは、（1）サービスをバックアップし、（2）サービスを削除し、且つ、（3）サービスを復旧する。システム 100 内でサービスのセットアップ及び／または使用中、サービス及び全ての依存関係データがバックアップされる（21.30 を参照）。システム使用中、サービスは、破損しているとして識別される（21.31 を参照）。例によると、ユーザは、バックアップからのサービスのリロードを望む、あるいは、自動回復プロセスが開始される（21.32 を参照）。サービスを削除して、バックアップから復旧する準備をする（例えば、サービスのバックアップデータ 2104 から）（21.33 を参照）。ステップ 21.34 で、クリーンアップルールが、例えば、本明細書の図 19A ～ 19G に関して記載されるように実行される。クリーンアップルールを実行することによって、システムは、復旧中に、何らかの情報を二重に書くことを避けることができる。例えば、サービスが（ldap／ウィンドウズアクティブディレクトリバインドアカウント等の）ログオン認証情報を必要とし、復旧ルールが、再び、そのアカウントを作るとする。クリーンアップルールは、再作成される前に、全てクリーンに削除することを確実にすることができる。これによって、復旧データがシステムに既にあるデータと衝突するのを回避する。復旧プロセスは次に、例えば、本明細書の図 21F に関してより詳細に記載するように、ステップ 21.35 で開始する。バックアップルール 2104 は、サービスの被依存サービス上で実行される復旧ルーチンを指示する。ステップ 21.36 で、これらの復旧ルーチンは、被依存サービス（複数可）上で実行される。サービスの依存関係は、サービス作成時に定義することができ、システム状態 220 及び／またはグローバルシステムルール 210 に記憶することができる。依存関係は、別の時にも満足させることができる、または、任意選択の依存関係であってよい。この情報は、サービステンプレート、グローバルシステムルール 210、サービスイメージ、システム状態 220（または、これらの組み合わせ）にあってよい。サービス及び被依存サービス（複数可）のデータ、情報、及び構成が、次に復旧される（ステップ 21.37 を参照）。

【0338】

図 21H は、サービスがどのようにバックアップルールのセットを有することができるかと、これらのバックアップルールが被依存サービス上のバックアップルールをどのように呼び出すことができるかを示す。この連携をバックアップ動作の一部として使用することは、困難であり得るが、コントローラ 200 と、これらの依存関係をマッピングできるシステム状態 220 とをシステムが使用することによって、バックアップから復旧するときに、サービスの依存関係の適切なバックアップに拡張する非常に効果的なサービス固有のバックアップ動作を可能にして、信頼できる復旧動作を可能にする。バックアップが実行されている間、コントローラ 200 がこれを管理していない場合でさえ、サービスが作成されていると、コントローラ 200 は、依存関係と通信し、他のバックアップルールを呼び出す方法を知るために必要な情報をサービスに提供する。システム 100 が、全てが互いに依存するアプリケーション／サービスのグループを有するとき、たとえデータを復旧したとしても、システムの残りは、異なっている場合があるので、サービスをバックアップするのが困難であり、図 21H のアーキテクチャが、問題の解決を助ける。

【0339】

図 21H では、システムは、上記のコントローラ 200 を有し、バックアップルール 2104 及びシステム状態 220 内に常駐し得るポインタ 220p、サービステンプレート、グローバルシステムルール 210、または依存サービス 1902 を含み得る。ポインタ

220pは、プロビジョニングされたストレージ、バックアップまたは復旧中のサービス、及び／または、使用中のバックアップルールを指すことができる。コントローラ200、または（サービス等、コントローラに結合することができる）リソース2100は、特に、依存サービス1902と通信して、バックアップルール2104を使用してバックアップを要求または呼び出すことができる。依存サービス1902は、サービス1902に関連付けられた、依存サービス1902（または、特定のタイプの依存サービスに）固有であってよい、または固有でなくてよい自身のバックアップルール2104aのセットを有してよい。被依存サービス1901a、1901b、1901cは、図19A～19Gを参照して本明細書で説明するように依存サービスに関連付けられてよい。被依存サービス1901cは、被依存サービス1901cに対して依存サービスであり得るサービス1901bに関連付けられてよい。被依存サービス1901a、1901b、1901cは、図21Hに示すようにバックアップルール2104b、2104c、2014d、2014eのうちの1つまたは複数のセットを有してよい。被依存サービスは、バックアップルールの複数のセットを有してよい、例えば、被依存サービス1901aは、バックアップルール2104b、2104cを有するとして示される。これらのバックアップルールは、被依存サービスのデータを部分的または完全にバックアップしてよく、バックアップされている依存サービスを復旧するために要求されたデータをバックアップすることができる。依存サービスバックアップルールは、そのサービスの特定のバックアップルールを呼び出してよい。例えば、2401eは、2401dに関連付けられたロジックに詳細に記載されてよい。バックアップルール2401aは、サービスまたは複数のサービスのデータをバックアップすることができるプログラム、命令、API呼び出し、及び、他のプログラムへの呼び出しを含み得る。バックアップルールは、被依存サービスのバックアップルールをロードまたは作成することができる、または、バックアップルールは、依存サービスの必要なデータを任意選択でバックアップする既存のバックアップルールを呼び出すことができる。これらの命令の一部は、データベースバックアップ、ファイルバックアップ、ファイルコピー、被依存サービス（複数可）からのデータ抽出を含むが、これらに限定されない機能を含み得る。データのアーカイブ、ストレージリソースへの接続、及び、ファイルシステム複製の実行は、図9Bから一部を含み得る（例えば、揮発性イメージ954がコピーされてよい）。

【0340】

図21I及び21Jは、1つまたは複数の被依存サービスを有する依存サービスのバックアッププロセスを記載する例示的なプロセスフローである。図21I及び21Jの方法は、バックアップルール2104及び／またはクリーンアップルール1904を使用して、任意選択で、サービスを削除し、サービスを復旧する、依存サービスのバックアップ及び復旧を可能にする。バックアップルール2104を用いたバックアッププロセスは、例えば、ユーザ要求によって、または、バックアップスタンドバイルーチンによってトリガされる（図21Iの21.40を参照）。ステップ21.41で、依存サービスのバックアップルールが呼び出される。依存サービスのバックアップルールは、被依存サービスのバックアップルールのセットをトリガし（21.42を参照）、それによって、関連する、選択された、または適切な被依存サービスデータが、ステップ21.50で開始する、より詳細に記載されたバックアップのために識別される（図21Jを参照）。

【0341】

ステップ21.43で、依存サービスデータは、ステップ21.50以下の前、中、または後にバックアップされる。依存サービス削除（または、更新）は、ステップ21.44で開始される。ステップ21.45で、コントローラロジック205は、例えば、本明細書の図19E～19Gに関して記載したように、クリーンアップルール1904を実行する。依存サービス、または更新された依存サービスを復旧する場合、ユーザまたはロジック、スクリプトまたはプログラムが、削除されたサービスの復旧を開始する（21.46を参照）。バックアップルールの復旧機能は、被依存サービスのバックアップルールから復旧動作をトリガする（21.47を参照）。復旧機能は、バックアップデータから復

旧動作を実施するルールまたはコードであってよく、復旧機能は、2115の一部であってよい（図21C-2を参照）、または、復旧機能は、それ自体のルールの一部であってよい（例えば、システムが、バックアップルールを有し、2つの別個のファイルとしてルールを復旧する場合）。ステップ21.48で、被依存サービスデータは、被依存サービスのバックアップルールによって定義されたトリガされた復旧動作に基づいて、または、ユーザによって定義された別の復旧プロセスによって復旧することができる。1つまたは複数のステップが、各被依存サービスが必要なデータを復旧されるように繰り返されてよい。そして、依存復旧プロセスが終了する（21.49を参照）。

【0342】

図21Jは、依存サービスが複数の被依存サービスに関連付けられているとき、バックアップルールを使用する例を記載する。依存サービスのバックアップルール（複数可）（または、戦略（複数可））は、どの情報が各被依存サービスからバックアップされる必要があるかを識別してよい（これは、各被依存サービスに関するバックアップルールの別個のセットがあってよい）（21.51を参照）。これは、例えば、本明細書の図19A～19Fに関して記載したサービス相互関係情報を使用して行うことができる。被依存サービスのバックアップルールは、被依存サービスデータをバックアップする方法を含むバックアップ戦略を含み得る。ステップ21.52で、被依存サービスバックアップ戦略が選択される。被依存サービスに対応する他のバックアップルールを使用して、バックアップルーチンがステップ21.53で実行される。先行するステップ前、ステップ中、またはステップ後に、ストレージリソースがプロビジョニングされてよい、または、ポイントが既存のストレージリソースを指すように設定されてよい（21.54を参照）。例えば、新しいストレージリソース（新しいi s e rもしくはn v m e o fターゲット、またはn f s等の共有フォルダ等）をプロビジョニングすることができる、または、既存のリソースへのリンクを接続することができる。（時には、相互依存関係を有し得る複数のサービスのバックアップが、物事をよりクリーンにするために1つのリソース上に全てのデータを有することが望ましい）。プロビジョニングされたストレージリソースが、バックアップルールを実行するサービス（または、バックアップルールを実行するコントローラ200）が、そのバックアップデータを記憶するために作成されてよい。これは、コントローラ200によって、または、サービス内のバックアップルールによって呼び出されたプログラムによって行うことができる。ストレージリソース（複数可）は、ステップ21.55で、任意選択で結合されてよい、または、ステップ21.55は、ストレージリソース（複数可）をバックアップされているサービスに結合することを伴ってよい、または、コントローラ200がサービスから適切なデータをフェッチしてよい。データは次に、例えば、図21C-1に示すように、バックアップデータに結合することができる。先行するステップを、各被依存サービスに対して繰り返すことができる。ステップ21.55はまた、呼び出された及び記憶された様々な被依存サービスのデータブロックのそれぞれに関して、ストレージリソース、ストレージリソース内のリソースもしくは位置が結合されることを含み得る。ステップ21.56で、バックアップルールを用いた被依存サービスのバックアップが、被依存サービスに対して完了され、ステップ21.57で、システム状態220が更新される。ステップ21.58で、被依存サービスに対する必要なバックアップが完了されたことを依存サービスに通知する。各追加の被依存サービスに対してバックアッププロセスが繰り返される（21.59を参照）。順番は、依存関係追跡の既知の方法を用いて決定することができる。被依存サービスが依存サービスである場合、その被依存サービスと共に依存サービスとして同様にバックアップされる（21.60を参照）。

【0343】

図21Kは、図19A～19D及び21Aに関して記載したシステム等の例示的なシステム100の依存サービスの被依存サービスのバックアップルールを生成またはロードする例を示す。ここでは、依存サービスが、被依存サービス（複数可）のためのバックアップルールを作成、ロード、提供し、依存サービスのバックアップルールが適切なバックアップルールをその被依存サービスに対して実行するのを可能にする。依存サービスをプロ

10

20

30

40

50

ビジョニングすることができ、その依存関係は、図 2 K を用いて記載したように解決または作成することができる。図 2 1 K は、ステップ 2 1 . 6 1 で依存サービスが作成されることを示す。依存関係は、図 2 K で、新しいサービスイメージを作成することによって、または、被依存サービスとして機能する既存のサービスに結合することによって解決される。次に、テンプレート、システムルール、及び／または、任意選択でサービスイメージ（複数可）の作成に使用できる他の情報が、コントローラロジックまたは他のプロセスが任意選択の依存関係と依存関係を満足させる複数の方法とを考慮して被依存サービスにロードする適切なバックアップルールを選択するために使用されてよい（2 1 . 6 3 を参照）。したがって、サービス A がサービス B 1 または B 2 との依存関係を満足できる場合、サービス A は、どちらを選択するかに応じて、正確なバックアップルールを B 1 または B 2 にポピュレートする。被依存サービスのバックアップルールは次に、システム状態 2 2 0、サービスイメージ、または、システム 1 0 0 の他の位置にロードすることができる（2 1 . 6 2 を参照）。

【0 3 4 4】

図 2 1 L - 1 及び図 2 1 - L 2 は、図 9 B に示す *Overlay fs* レイアウトを使用して、図 1 9 A ~ 1 9 D 及び 2 1 A に関して記載したシステム等のシステム 1 0 0 のサービス 1 9 0 1 の例示的なバックアップ方法及び復旧方法をそれぞれ、示す。図 2 1 L - 1 は、*Overlay fs* のレイヤが *lvm* クローニングまたは他のファイルシステムクローニング方法等のツールを用いてクローンされるバックアップルールの例を示す。バックアップルールがトリガされ、バックアップルールは、揮発性イメージ（複数可） 9 5 4 のバックアップを指示する（2 1 . 6 4 参照）。このデータは、次に、バックアップデータ 2 1 1 0 にパッケージ化される（2 1 . 6 5 参照）。図 2 1 L - 2 は、図 2 1 L - 1 に示すバックアッププロセスから対応する復旧プロセスを示す。復旧は、2 1 L - 1 に示すバックアップルールからトリガされる（2 1 . 6 6 参照）。復旧されたサービスの組み立てが開始される。ベースイメージ 9 5 2、サービスイメージ 9 5 3、及び、他の可能なイメージが任意選択で *Overlay fs* を用いて結合される（2 1 . 6 7 参照）。揮発性イメージ（複数可） 9 5 4 がバックアップデータ 2 1 1 0 から抽出される（2 1 . 6 8 参照）。揮発性イメージ 9 5 4 と、任意選択でサービスイメージとは次に、任意選択で *Overlay fs* を用いてベースイメージ 9 5 2 及び／またはサービスイメージ 9 5 3 に結合される（2 1 . 6 9 参照）。次にサービスが復旧される（2 1 . 7 0 参照）。

【0 3 4 5】

更新ルールのプロビジョニング及び使用

図 2 2 A は、図 1 9 A ~ 1 9 D 及び 2 1 A に関して記載したシステム等の例示的なシステム 1 0 0 の例示的な更新ルール 2 2 0 4 を示す。ここでは、更新ルール 2 2 0 4 のセットが、コントローラ 2 0 0 にさらに含まれる。更新ルール 2 2 0 4 は、コントローラ 2 0 0 内にそれ自体のルールセットとして具現化されてよい、または、グローバルシステムルール 2 1 0、コントローラロジック 2 0 5、テンプレート 2 3 0、もしくはそれらの組み合わせで、または、1 つまたは複数のサービス、被依存サービス、及び／または依存サービスのサービステンプレートの一部として具現化されてよい。さらに、サービス、例えば、サービス 1 9 0 1、1 9 0 2 のテンプレートは、更新ルール 2 2 0 4 のセットを含み得る。

【0 3 4 6】

更新ルール 2 2 0 4 は、（テンプレートの形態を取ってよい）ソフトウェアの 1 つまたは複数のサポートされた古いバージョン 2 2 0 5、またはそれらへのポインタを含み得る。更新ルール 2 2 0 4 は、（テンプレートの形態を取ってよい）ソフトウェアの 1 つまたは複数のサポートされた新しいバージョン 2 2 0 6、またはそれらへのポインタをさらに含み得る。さらに、更新ルール 2 2 0 4 は、被依存サービスの方法／ルールを更新するポインタ 2 2 0 7 を含み得る。更新は、変更管理ルールを用いることによって、及び／または更新時にバックアップルール 2 1 0 4 を用いたバックアップと必要なまたは選択されたデータを復旧した後に古いバージョンを削除することによって、システム 1 0 0 にロード

することができる。依存サービス及び被依存サービスのバックアップの例を図 2 1 A ～ 2 1 J を参照して説明する。更新ルール 2 2 0 4 はまた、（復旧命令を含む）推奨されたバックアップルール 2 2 0 8 も含み得る。推奨されたバックアップルールは、本明細書に記載のバックアップルール 2 1 0 4 （または、そのサブセット）、例えば、バックアップ戦略を含み得る。ポインタ（複数可） 2 2 0 7 は、グローバルルール 2 1 0、テンプレート 2 3 0、及び／またはリソース、例えば、サービス内のバックアップ／復旧ルールを指すことができる。

【 0 3 4 7 】

図 2 2 B は、更新ルールを用いた例示的な更新プロセスの例示的なプロセスフローである。このプロセスでは、1 つまたは複数の依存サービスは、クリーンアップルール 1 9 0 4 を用いて削除することができ、その後、復旧することができる。図 2 2 B のプロセスフローは、更新プロセスが開始されるとき、開始する（2 2 . 0 1 を参照）。適切なルールがロード／呼び出される（2 2 . 0 2 を参照）。ルールは、被依存サービス上で処理される（2 2 . 0 3 を参照）。例えば、サービスを削除する場合、本明細書の他の箇所に記載のバックアップルール 2 1 0 4 を呼び出すことができる（2 2 . 0 4 を参照）。次に、サービスが削除される（2 2 . 0 5 を参照）。クリーンアップルール 1 9 0 4 は、本明細書の他の箇所に記載のように実行することができる（2 2 . 0 6 を参照）。次に、選択した情報データを復旧するために、バックアップルール 2 1 0 4 を呼び出して、実行することができる（2 2 . 0 7 を参照）。

【 0 3 4 8 】

図 2 2 C は、図 1 9 A ～ 1 9 D、2 1 A、2 1 L - 1、2 1 L - 2、及び 9 B に関して記載したシステム等の例示的なシステム 1 0 0 の更新ルール 2 2 0 4 を用いた例示的な更新プロセスを示す。図 2 1 L - 2 は、図 2 1 L - 1 に示すバックアッププロセスから対応する復旧プロセスを示す。ステップ 2 2 . 0 8 で、ユーザまたはコントローラロジック（または、これらの組み合わせ）が更新をトリガする。2 2 0 5 及び 2 2 0 6 を使用して更新ルールのバージョン互換性をチェックする（2 2 . 0 9 を参照）。次に、更新ルールを開始する（2 2 . 0 2 を参照）。これらの更新ルールを用いて、揮発性データは、新しいサービスファイルシステムイメージに結合され、任意選択で更新ルール 2 2 0 4 で任意選択でサポートされたより新しいベースイメージに結合される。サービスの揮発性データは、バックアップデータ 2 1 1 0 から抽出することができる（2 2 . 1 0 を参照）。ベースイメージ 9 5 2 は、任意選択で Overlay fs を使用して、新しいサービスイメージ 9 5 3 と結合され（2 2 . 1 1 を参照）、揮発性データ 9 5 4 は、新しいイメージを有するサービス内に結合され（2 2 . 1 2 を参照）、次にサービスが、まだ損なわれていない全てのデータを用いて更新される（2 2 . 1 3 を参照）。

【 0 3 4 9 】

クラウド上にインスタンスをプロビジョニング及び接続

図 9 F は、クラウド上のインスタンス 3 1 0 a を含むリソースへの追加接続を有する、図 9 D に示すシステム及びコントローラ 2 0 0 を示す。この文脈で本明細書で使用する場合、「クラウド」という用語は、口語では「クラウド」として知られ、Amazon、Microsoft 等のクラウドプロバイダから入手可能なネットワーク化されたコンピュータシステムなど、サービスをシステムに提供するために利用可能なコンピュータシステムの外部を指す。クラウドインスタンス 3 1 0 a は、アクセス及び仕様が望まれるシステム 1 0 0 の外部のコンピュータ上にある任意のリソースの形態を取ることができる。例えば、任意の仮想プライベートサーバ（VPS）、Amazon Elastic Compute Cloud（EC2）インスタンス、及び／または、Azure インスタンスが、クラウドインスタンス 3 1 0 a として機能してよい。接続 2 6 5 は、クラウド API 9 8 0 に結合され、API 9 8 0 は、接続 2 6 5 を通して依頼または購入されると、インスタンス 3 1 0 a をプロビジョニングする。接続 2 6 5 は、クラウドリソースをプロビジョニング、修正、及び／または破壊するための機構であってよい。接続 2 6 5 は、クラウドインスタンス 3 1 0 a からアウトオブバンドのままであることができる。すなわち、接続

10

20

30

40

50

265は、クラウドインスタンスのオペレーティングシステムに接続されない。したがって、クラウドインスタンスのオペレーティングシステムは、接続265を見なくてよい。実施者が望む場合、接続265はまた、クラウドインスタンスのオペレーティングシステムの範囲外にある操作を提供してよい。コントローラ200を通してシステム100に一旦プロビジョニングされると、インスタンス310aは、コントローラ200と通信し、任意選択でインバンド管理270もしくはクラウドインスタンス310aへの他の接続によって、及び、任意選択のVPN990を通して、システム100に計算機能または他の機能を提供してよい。

【0350】

図9G-1は、クラウド上にストレージリソースを作成するための図9Fのシステム及びコントローラの使用を記載する例示的なプロセスフローである。ステップ900.11で、コントローラは、クラウドAPI980を通してインスタンスをプロビジョニングする。例えば、このプロビジョニングは、AWS Lambda等のサーバーレスクラウドAPI、またはクラウドインスタンスを購入するための任意の技術を通して実行することができる。ステップ900.12で、コントローラ200は、ストレージバケット900.12を作成する。ストレージバケットがクラウドベースである場合、ストレージバケットは、クラウドプロバイダのコンピュータに配置することができる。ストレージバケットは、ストレージにリモートでアクセスするための機構を提供する。そのストレージは、承認されたユーザ／マシンのみがストレージバケット内のファイルにアクセスできるのを確実にするためのアクセス制御及び認証を有してよい。例として、Amazonシンプルスストレージサービス(S3)は、一種のストレージバケットであってよい。コントローラ200は、ストレージバケットをプロビジョニングされたインスタンスと整合してよい、及び／または計算インスタンスを作成し（且つ、それをストレージにも使用）してよい。コントローラ200は次に、クラウドストレージリソース接続情報をシステム状態220に保存する（900.13参照）。

【0351】

図9G-2は、プロビジョニングされたクラウドストレージリソース（310aを参照）をシステムの計算リソースに接続するために図9Fのシステム及びコントローラ200の使用を記載する例示的なプロセスフローである（900を参照）。コントローラ200は、例えば、図9G-1に記載するように（900.14を参照）、ストレージリソースをクラウド上に作成する、または、既にクラウド上にプロビジョニングされたストレージリソースを識別する（900.15を参照）。コントローラ200は次に、プロビジョニングされたクラウドストレージリソースに（または、クラウドAPI980に）情報を提供して、新しい計算リソースをストレージリソースに接続するのを可能にする（900.16を参照）。この情報は、認証情報、ネットワークアドレス、リソースの位置、接続情報、接続の仕方に関する命令、暗号鍵、証明書、公開鍵、証明機関、及び／またはストレージリソースへの接続に必要な、または接続を容易にし得る任意の他の情報を含み得るが、これらに限らない。ステップ900.17で、計算リソースは次に、作成またはセットアップされて、サービスイメージの命令と共にクラウドストレージリソースに結合される。この関連で、サービスイメージは、構成の構築、及び／またはクラウドインスタンス（複数可）上へのサービス（複数可）のプロビジョニングを可能または容易にする命令、構成情報、及び／またはコンピュータプログラムを含み得る。あるいは、ステップ900.17で、計算リソースを、作成またはセットアップすることができ、コントローラ命令と共にクラウドストレージリソースに結合することができる。この関連で、コントローラ200は、APIエンドポイントを呼び出すことによって、命令を送信することが多い。したがって、ステップ900.17で、コントローラ200は、リモートでインスタンスにログインすることによって、または、APIコマンドを実行することによって、作業を行うことができる、または、命令及び情報をクラウドプロバイダ／クラウドAPIに提供することができ、それによって、クラウドプロバイダが全ての命令を実行する。

【0352】

図9G-3は、コントローラ200によって管理されるクラウドリソースプールをプロビジョニングするためのコントローラ200の使用を示す例示的なフローである。ステップ900.30で、クラウドインスタンス310aは、クラウド計算リソース対応ホストとして、システムに追加される。ステップ900.31で、コントローラ200は、ホストのセットアップを実行する。このセットアップは、サービスイメージをホストに提供することによって、または、オペレーティングシステムバージョンを指し、ブート時に実行するスクリプトを提供することによって、実行することができる。あるいは、ホストのオペレーティングシステムが、プリロードされる、及び／またはホストが予め構成される（ステップ900.32を参照）。例えば、ハードドライブイメージをクラウドにアップロードすることができ、次に、サーバまたは仮想サーバを、そのハードドライブイメージを用いてスピンアップすることができる。ステップ900.33で、クラウドリソースが、計算リソースプールとしてシステム状態220に追加される。クラウドリソースプールを追加する場合、ステップ900.33の前に追加のステップを実行することができ、その追加のステップで、コントローラ200は、対象のクラウドプロバイダに接続するための認証情報を与えられる。ステップ900.34で、VPN接続990が、コントローラ200とホスト間のインバンド管理270を通して作成される。例えば、ハードドライブイメージをクラウドにアップロードすることができ、次に、サーバまたは仮想サービスをそのハードドライブイメージを用いてスピンアップすることができる。そうすると、クラウドリソースは、システム100によって仮想のまたはクラウドリソースプールとして使用することができる（900.35を参照）。例えば、クラウドリソースは、コントローラ200に利用可能にされたコンテナホスト（または、仮想マシン等の計算リソースの他のバージョン）として使用することができ、コントローラ200は、このようなコンテナまたは他のサービスをこのクラウドリソースにプロビジョニングしてよい。

【0353】

図9G-4は、クラウドリソースをシステムに追加するための、またはクラウド上のシステムにさらにリソースをプロビジョニングするための技術を示す例示的なプロセスフローを示す。ステップ900.40で、コントローラロジック205は、クラウドAPI980に接続し、認証情報を使用して、クラウドAPI980を介して認証、通信、使用、及び／または購入を行う。このような認証情報は、ユーザがクラウドプロバイダに有し得るアカウントに対してユーザによって提供されてよい。支払いもこのようなアカウントを通して行われてよい。クラウドAPI980は、インスタンス310aの購入、及び／またはコントローラ200のインスタンス310aの接続を可能にする（900.41を参照）。図9Fに示す接続265を用いて、コントローラ200は、リソースのインスタンスのタイプ、及びその構成を識別する（900.42を参照）。コントローラ200は、グローバルシステムルール210を使用して、特定のインスタンスを自動的に購入及び／または追加すべきかどうかを判定する（900.43を参照）。例えば、ルール210は、必要なリソースタイプを指定することができる（例えば、ルール210は、amazon m4. largeが必要であると示してよい）。ルール210はさらに、サービスの電源がオンになると、コントローラ200が必要とされるリソースタイプに対応するインスタンスを購入及び構成することを指定してよい。それが自動的に行われない場合、コントローラ200は、インスタンスを使用及び／または購入するための承認を待つ（900.44を参照）。それが自動的に行われる場合、コントローラ200は、購入及び／または自動的セットアップのためにそのルール210を使用する（900.45を参照）。購入及び／またはセットアップ後、コントローラ200は、ステップ900.46に進み、テンプレート（複数可）230を使用して、インスタンス310aをリソースとしてシステムに追加する。ステップ900.47で、コントローラ200は、システムルール210に従って、接続265を通して、リソース／インスタンス310aの電源をオンにする、及び／または有効にする。システムルール210を用いて、コントローラ200は、リソースのタイプに基づいて、テンプレート（複数可）230からリソース／インスタンス310aのブートイメージを見つけて、ロードする（900.48を参照）。このロード

10

20

30

40

50

は、接続 265 を通して実行することができる。サービス、アプリケーションまたはリソース 310 a は、サービスまたはアプリケーションのイメージからブートされ、クラウドインスタンス 310 a は、有効にされる、電源をオンにされる、及び／または接続を有効にされる（900.49 を参照）。あるいは、イメージは、既存のインスタンステンプレート（例えば、230）、及び構成スクリプト（例えば、Ubuntu12.10 及びシェルスクリプト）から生じてよい。このような場合、テンプレート 230 は次に、変更されてよい（または、図 9A 及び付属書類 B を参照して前述した Overlayfs システムを用いて重ねられたファイルを有してよい）。リソース／インスタンス 310 a に関する情報は、インバンド管理接続 270 または接続 265 を通してリソース／インスタンス 310 a からコントローラ 200 によって受信されてよい（900.50 を参照）。新しいリソース情報は、インバンド管理 270 を通してクラウドインスタンスから提供され、システム状態 220 に追加される（400.51 を参照）。インスタンス 310 a は、リソースプールに追加され、割当の準備が行われる。

【0354】

図 9H は、クラウド上に追加のインスタンス 310 b を有する図 9F に示す例示的なシステムを示し、その追加のインスタンス 310 b は、クラウド API 980 に接続され、インバンド管理接続 270 によって VPN 990 を通してコントローラ 200 にも接続される。インスタンス 310 a 及び 310 b は、図 19A～19G を参照して記載したように共通 API 1903 を通して接続されてよく、インスタンス 310 a、310 b は、非依存サービスと依存サービスとして対話してよい。共通 API 1903 もクラウド上にあってよい。インスタンス 310 a、310 b はまた、図 19A～19G で説明したサービスの通信と類似の方法で、インスタンス 310 a、310 b 間の通信のために任意選択の VPN 990 a を通して結合されてよい。

【0355】

図 9I は、クラウド上にストレージリソース／インスタンスを作成するための図 9H のシステム及びコントローラ 200 の使用を説明する例示的なプロセスフローである。ステップ 900.60 で、コントローラ 200 は、サービスイメージを作成する。このサービスイメージは、クラウド上の計算リソースまたはストレージリソースでサービスまたはイメージを実行するのに必要な情報を含むオペレーティングシステムサービスイメージであってよい。ステップ 900.61 で、コントローラ 200 は、VPN 接続情報を提供してよい。この VPN 接続情報は、インスタンスが VPN を介して相互に通信するのを可能にする。この一部として、コントローラ 200 は、鍵／認証情報と、VPN を作成するのに必要な任意の他の情報とを提供することができる。VPN は、2 点間のデータを暗号化することができるが、これは、オープンなインターネットに公開されるポートを少なくできることを意味する。ステップ 900.62 で、コントローラ 200 は、クラウド API 980 を使用して、インスタンス 310 a 及び／または 310 b 等のインスタンスを作成する。ステップ 900.63 で、コントローラ 200 は、310 a 及び／または 310 b 等のクラウドインスタンスを（任意選択で、サービスイメージを用いて）作成する。ステップ 900.64 で、コントローラ 200 は、コントローラ 200 とインスタンス（例えば、310 a、310 b）間のインバンド管理 270 を通して通信のために（または、クラウド API 990 を通してサポートされた安全な接続のために）VPN 990 を使用する。

【0356】

図 9J も、クラウド上に追加のインスタンス 310 b を含む図 9F に示す例示的なシステムを示し、インスタンス 310 b は、クラウド API 980 に接続され、インバンド管理接続 270 によって VPN 990 を通してコントローラ 200 にも接続される。インスタンス 310 a、310 b は、任意選択の VPN 990 a を通して結合されてよい。コントローラ 200 はまた、（図 19A～19G に記載するように）インバンド管理接続 270 によって、インスタンス 310 a、310 b に共通 API 1903 を通して結合されてよく、インスタンス 310 a、310 b は、非依存サービス及び依存サービスとして対話してよい。共通 API 1903 は、コントローラ 200 の一部であってよい、または、別

個であってよい。共通API 1903は、インスタンス310a、310bによって使用されて、図19A～19Gを参照して説明したように、依存サービスに対して被依存サービスによって実行される機能を求めてよい。

【0357】

図9Kは、被依存サービス及び依存サービスとして働く複数のインスタンスを含むストレージリソースをクラウド上に作成するための図9Jのシステム及びコントローラ200の使用を説明する例示的なプロセスフローである。システムルール210、または、ユーザアクションは、依存サービスを配備するために必要なとき、被依存サービスを配備するようにコントローラ200に指示する(900.70を参照)。コントローラ200は、クラウドAPI 980を使用して、被依存サービス及び依存サービスを配備する(900.71を参照)。ステップ900.72で、クラウド計算リソースまたはインスタンスを配備する。この配備は、コントローラ200がサービスイメージをクラウドAPIに提供することによって実行することができる。コントローラ200は、命令(例えば、シェルスクリプト(複数可)、スクリプト(複数可)、コンピュータプログラム(複数可)、オペレーティングシステムテンプレート(複数可)、及び/またはクラウドホスト上で稼働するサービスイメージを作成するために必要な他の情報)も提供してよい。ステップ900.73で、コントローラ200は、サービスの共通API 980への接続の仕方(コントローラ200を通して、またはVPN 990を通して)に関する命令を提供する。コントローラが共通APIに接続できる多くの方法がある。例えば、最少のセキュリティを提供する1つのオプションは、インターネット上のオープンなAPIであってよく、http要求などの要求を行ってよい。他のオプションは、追加のセキュリティを提供することができる。例えば、別のオプションは、クラウドプロバイダを通して提供されたツールを通してクラウドインスタンス(または、インスタンスがインターネットに公開されていないその一部)にアクセスする。別のオプションは、クラウドインスタンスは、VPNをインターネットに公開することができ、コントローラ200が、VPNに接続できる。さらに別のオプションは、インスタンスがVPNに接続してコントローラと通信することであってよい。インスタンスがプロビジョニングまたは作成されると、このような接続を容易にするために必要な構成情報及びコンピュータプログラムがプロビジョニングされる。ステップ900.74で、依存サービスは、依存関係を満足させる被依存サービスを(任意選択で、インスタンス間のVPN接続990aを通して)構成する。

【0358】

別の例示的な実施形態では、システム100は、クラウドホストシステムとしてクラウドに配置されてよい。このようなクラウドホストシステムは、実施者が望めば、クラウドから分離されて、ローカルにホストされたシステムに移動させることができる。例えば、図9Lは、システムをローカルホスト上、または、ローカルにホストされたシステムにクラウドから分離するための例示的なプロセスフローを示す。このような実施形態を用いて、実施者は、クラウド上で自分たちの作業を実行、または自分たちのシステムを運転し、その後、システムを1つまたは複数のローカルホスト(複数可)に分離することによって取り除くことができる。実施者はまた、クラウドに固定されていないITシステムを構築することができ、後日、そのクラウドから取り除くことができる。これに関して、システムルールを含むシステム100は、クラウドに存在することができる(ステップ900.80を参照)。これらのシステムルールは、新しくローカルにホストされたシステムにコピーすることができる(ステップ900.81を参照)。コントローラロジックは次に、これらのコピーされたシステムルールをローカルにホストされたシステム上で処理して、システムを効果的にクラウドから分離し、システムをローカルホスト上に移動させることができる。例として、ステップ900.82は、図2A-20を参照して前述した技術を用いて実行することができる。

【0359】

発明を例示的な実施形態に関して記載してきたが、発明の範囲内にある様々な修正が発明に対して行われてよい。発明に対するこのような修正は、本明細書の教示を考慮して認

10

20

30

40

50

識可能である。

【0360】

【表1-1】

付属書類A：例示的なストレージ接続プロセス

複数のシステム間でストレージリソースを共有することに関する例示的なプロセスと例示的なルールを記載する。これは、ストレージ接続プロセスのほんの一例にすぎず、計算リソースをストレージリソースに接続する他の技術を使用できることを理解されたい。別段の記載のない限り、これらのルールは、ストレージ接続の開始を試みる全てのシステムに当てはまる。

10

この付属書類Aの定義

ストレージリソース：ストレージトランスポートを介して共有できるブロック、ファイル、または、ファイルシステム

ストレージトランスポート：ストレージリソースをローカルにまたはリモートに共有する方法。例は、iSCSI／iSER、NVMe over Fabrics、NFS、Sambaファイル共有である。

システム：指定されたストレージトランスポートを介してストレージリソースに接続を試みることができる任意のもの。システムは、任意の数のストレージトランスポートをサポートしてよく、どのトランスポートを使用すべきかをシステム自体が決定できてよい。

20

読み取り専用：読み取り専用ストレージリソースは、そのストレージリソースが含むデータの修正を許可しない。この制約は、ストレージトランスポートでストレージリソースのエクスポートを処理するストレージデーモンによって課される。さらに保証するために、幾つかのデータストアが、データをバックアップするストレージリソースを読み取り専用を設定してよい（例えば、LVM L Vを読み取り専用に設定）。

読み書き（または、揮発性）：読み書き（揮発性）ストレージリソースは、そのコンテンツがストレージリソースに接続するシステムによって修正されてよいストレージリソースである。

30

ルール：システムが所与のストレージリソースに接続し得るか否かをコントローラが判断するとき、従わなければならないルールのセットがある。

1. 読み書きストレージリソースは、1つのストレージトランスポートでのみエクスポートされるものとする。
2. 読み書きストレージリソースは、1つのシステムによってのみ接続されるものとする。
3. 読み書きストレージリソースは、読み取り専用として接続されてはならない。
4. 読み取り専用ストレージリソースは、複数のストレージトランスポートでエクスポートされてよい。
5. 読み取り専用ストレージリソースは、複数のシステムから接続されてよい。
6. 読み取り専用ストレージリソースは、読み書きとして接続されてはならない。

40

50

【表 1 - 2】

プロセス

接続プロセスを関数と考える場合、接続プロセスは、2つの独立変数を取る。

1. ストレージリソース ID
2. サポートされたストレージトランスポートのリスト（順序によって優先付け）

最初に、要求されたストレージリソースが、読み取り専用か読み書きかを判断する。

読み書きである場合、読み書きストレージリソースは1つの接続に制限されるので、ストレージリソースが既に接続されているか否かをチェックしなければならない。既に接続を有する場合、ストレージリソースを要求しているシステムが、現在接続されているシステムであることを確認する（これは、例えば、再接続の場合に生じる場合がある）。そうでない場合、複数のシステムは同じ読み書きストレージリソースに接続できないので、エラーを出す。

10

要求しているシステムが、このストレージリソースに接続されているシステムである場合、利用可能なストレージトランスポートのうちの1つが、このストレージリソースの現在のエクスポートに一致することを確認する。一致する場合、接続情報を要求しているシステムに伝える。一致しない場合、複数のストレージトランスポートで読み書きストレージリソースを供給できないので、エラーを出す。

20

読み取り専用ストレージリソース及び接続されていない読み書きストレージリソースに関しては、供給されたストレージトランスポートのリストを順に処理し、そのトランスポートを用いて、ストレージリソースのエクスポートを試みる。エクスポートに失敗すると、エクスポートに成功するまで、または、ストレージトランスポートが無くなるまで、リストの順にエクスポートの試みを続ける。ストレージトランスポートが無くなると、要求しているシステムに、ストレージリソースは接続できなかったことを通知する。エクスポートに成功すると、接続情報と、新しい（リソース、トランスポート）=>（システム）関係とをデータベースに記憶する。次に、要求しているシステムは、ストレージトランスポート接続情報を伝えらえる。

システム：ストレージ接続は、現在、通常動作中、コントローラと計算デーモンによって行われる。しかしながら、将来の繰り返しは、サービスが、ストレージリソースに直接接続し、計算デーモンをバイパスしてよい。これは、例示的なサービス物理的配備の要件であってよく、仮想マシン配備に対しても同じプロセスを使用することは理に適う。

30

40

50

【表 2 - 1】

付属書類 B : 例示的な OverlayFS への接続

サービスは、OverlayFSを利用して、共通ファイルシステムのオブジェクトを再利用し、サービスパッケージサイズを低減する。

この例のサービスは、3つ以上のストレージリソースを含む。

1. プラットフォーム。これは、ベースのLinuxファイルシステムを含み、読み取り専用でアクセスされる。
2. サービス。これは、サービスのオペレーションに直接関連する全てのソフトウェア（NetThunder Service Daemon、OpenRCスクリプト、バイナリ等）を含む。このストレージリソースは、読み取り専用でアクセスされる。
3. 揮発性。これらのストレージリソースは、システムに対する全ての変更を含み、（物理的、コンテナ、及び、仮想マシン配備のために）サービス内からLVMによって管理される。

10

仮想マシンで実行する時、サービスは、以下を行うロジックを含むinitramfsと共にカスタムのLinuxカーネルを用いて、Qemuでカーネルから直接ブート（Direct Kernel Boot）される。

1. 利用可能な読み書きディスクからLVMボリュームグループ（VG）を組み立てる
*このVGは、サービスのための全ての揮発性ストレージデータを含む1つのロジカルボリューム（LV）を含む。
2. プラットフォーム、サービス、及び、LVをマウント
3. ユニオンファイルシステム（ここでは、OverlayFS）を用いて、3つのファイルシステムを結合

20

同じプロセスが、物理的配備に使用できる。1つの選択肢は、PXEBootまたはIPMI ISO Bootを介してブートされた軽量OSにリモートでカーネルを提供し、次に、新しい実際のカーネルにkexecを使用する。または、軽量OSをスキップし、カーネル内に直接、PXEブートする。このようなシステムは、ストレージリソースに接続するためにカーネルinitramfsの追加のロジックを必要とする場合がある。

30

40

50

【表 2 - 2】

OverlayFS構成は、次のようになり得る。

```

／-----＼
| 揮発性レイヤ (LV) (RW) |
+-----+
| サービスレイヤ (RO) |
+-----+
| プラットフォームレイヤ (RO) |
＼-----/

```

10

OverlayFSのいくつかの制限によって、特別なディレクトリ ‘／data’ を「アウトオブツリー」としてマーク付けすることが可能になる。このディレクトリは、サービスパッケージを作成するとき、‘／data’ ディレクトリを作成すると、サービスに利用可能になる。この特別なディレクトリは、‘mount --rbind’ を介してマウントされて、OverlayFS内にはないサブセットの揮発性レイヤへのアクセスを可能にする。これは、OverlayFSの一部である共有ディレクトリをサポートしていないNFS（ネットワークファイルシステム）等のアプリケーションに必要である。

カーネルファイルシステムレイアウト：

20

```

／
+-- platform／
  +-- bin／
  +--... ／
+-- service／
  +-- data／ [optional]
  +-- bin／
  +--...
+-- volatile
  +-- work／
  +-- root／
    +-- bin／
    +-- data／ [if present in service／]
    +--...
+-- new_root／
+--...

```

30

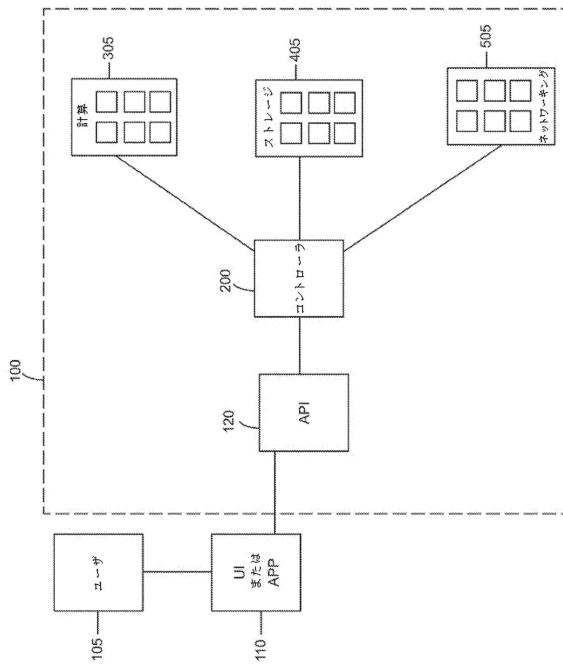
／new_rootディレクトリを作成し、そのディレクトリをOverlayFSを構成するためのターゲットとして使用する。OverlayFSが構成されると、／new_ディレクトリ内にexec_rootを行うと、システムは、全ての利用可能なリソースと共に正常に開始する。

40

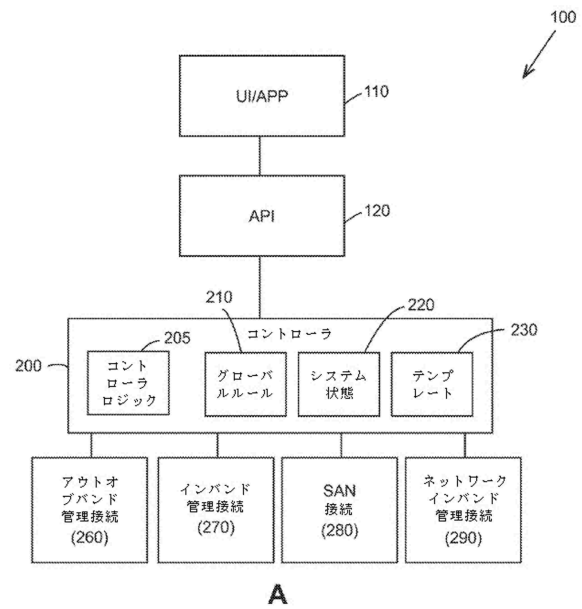
50

【図面】

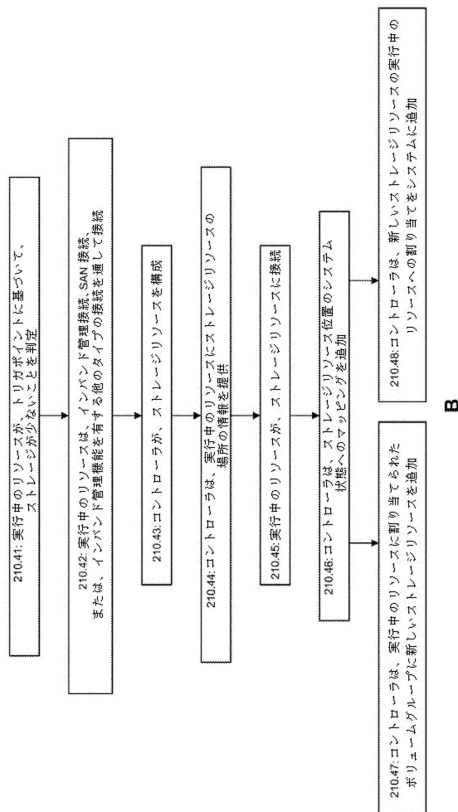
【図 1】



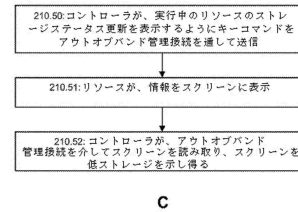
【図 2 A】



【図 2 B】



【図 2 C】



10

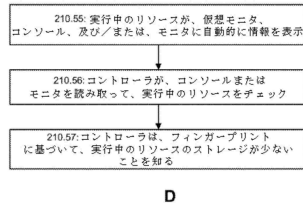
20

30

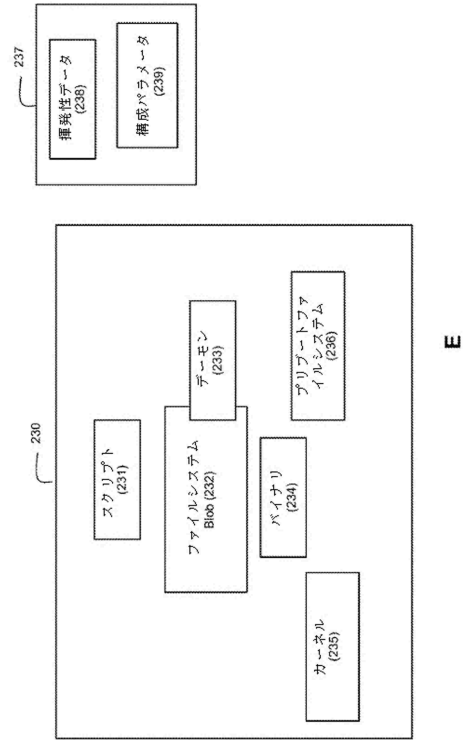
40

50

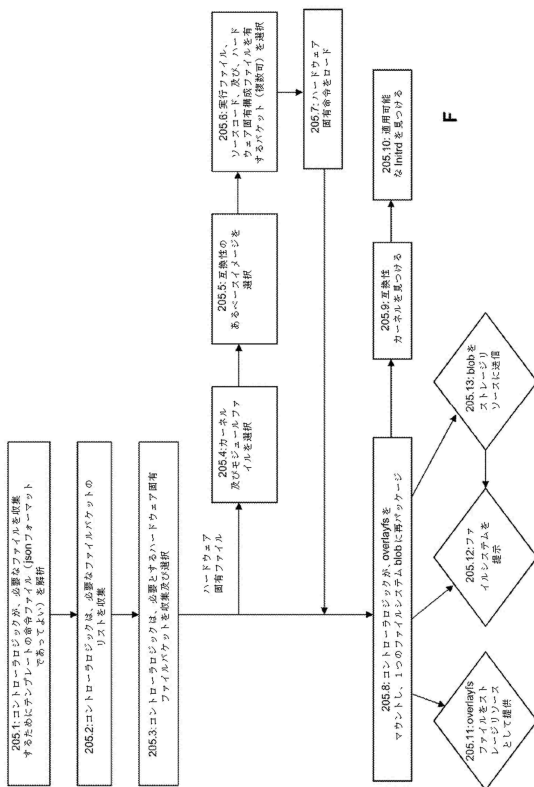
【図 2 D】



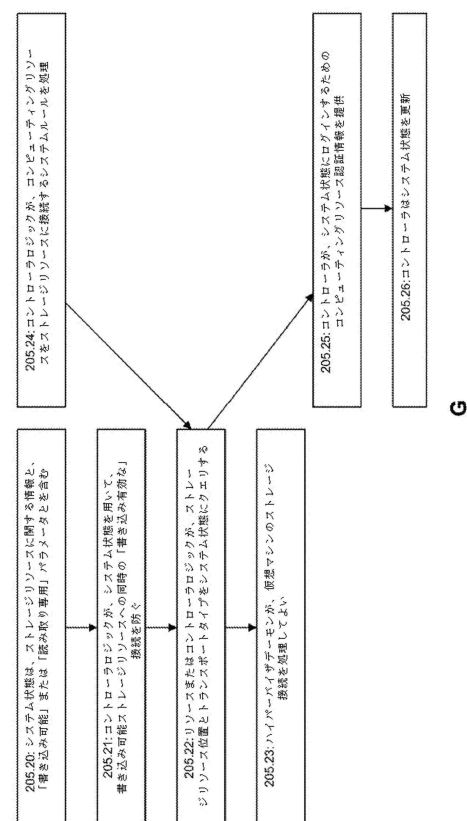
【図 2 E】



【図 2 F】



【図 2 G】



10

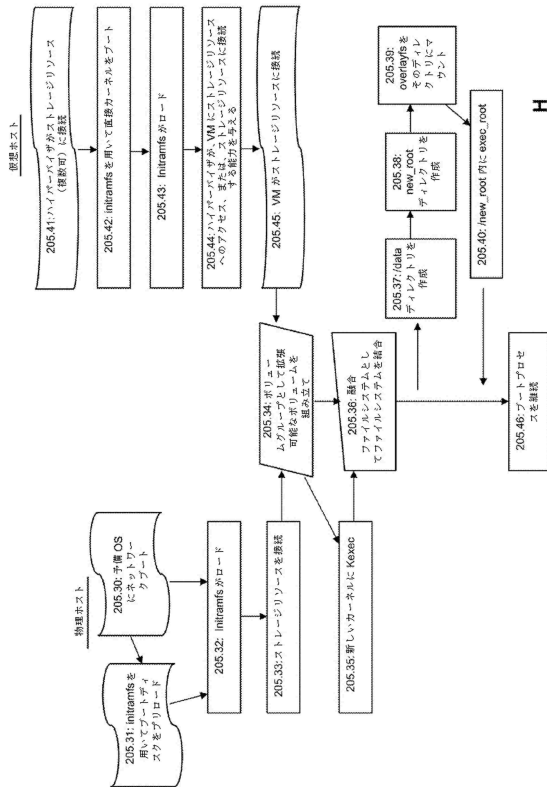
20

30

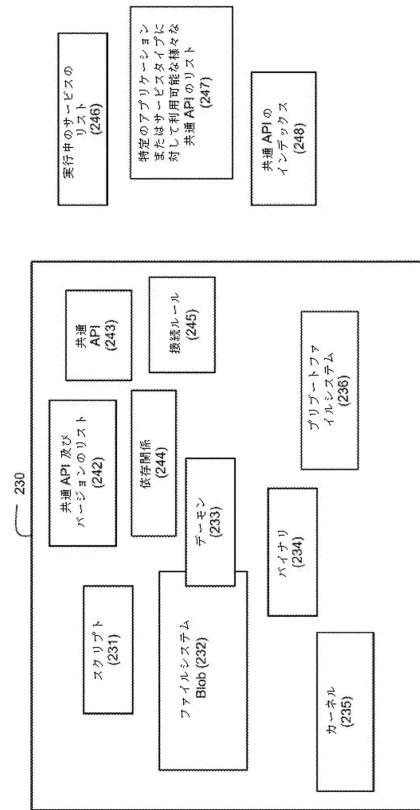
40

50

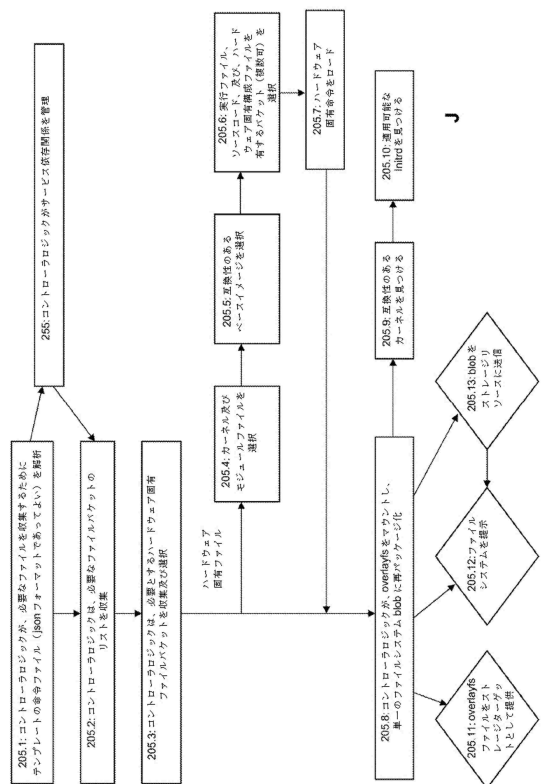
【図 2 H】



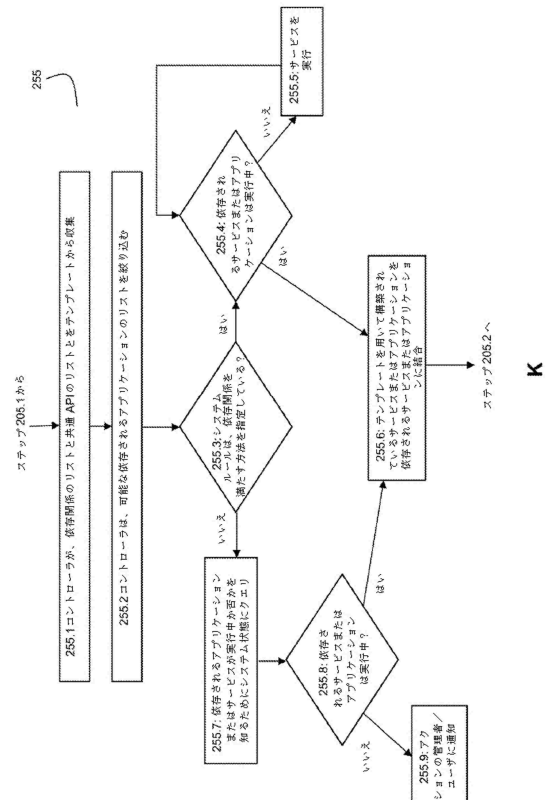
【図 2 I】



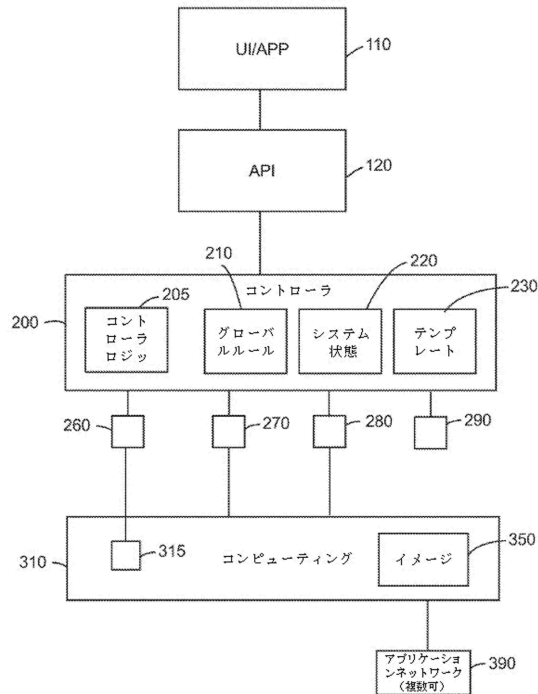
【図 2 J】



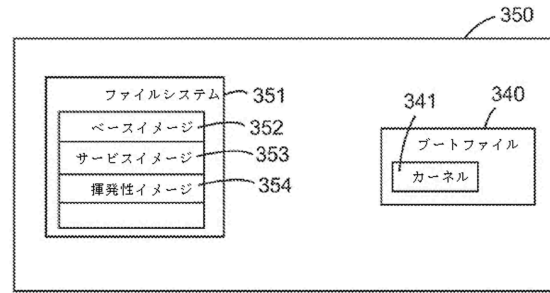
【図 2 K】



【図 3 A】



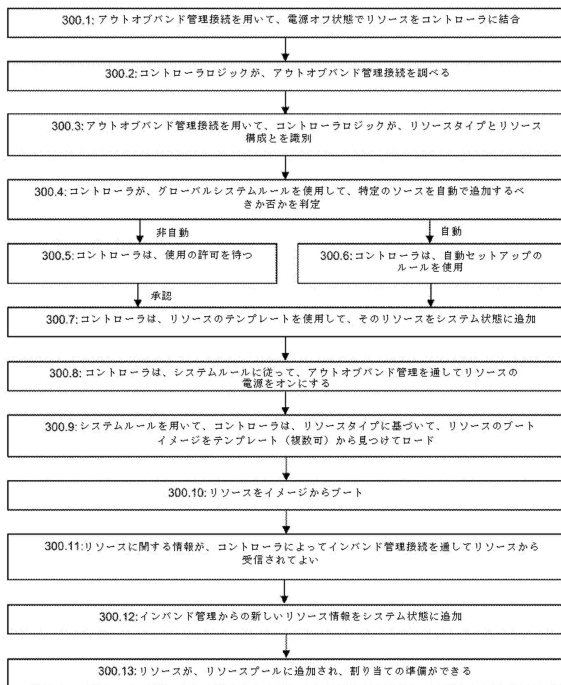
【図 3 B】



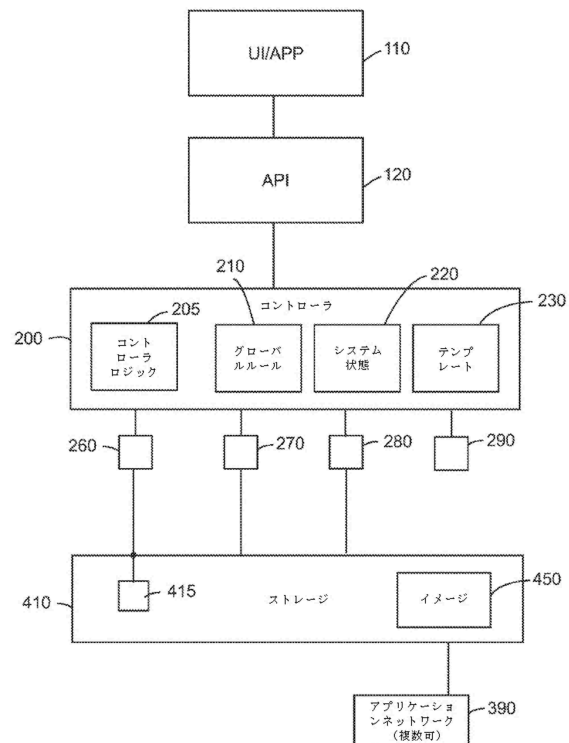
10

20

【図 3 C】



【図 4 A】

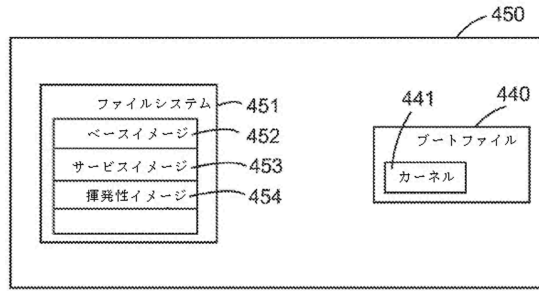


30

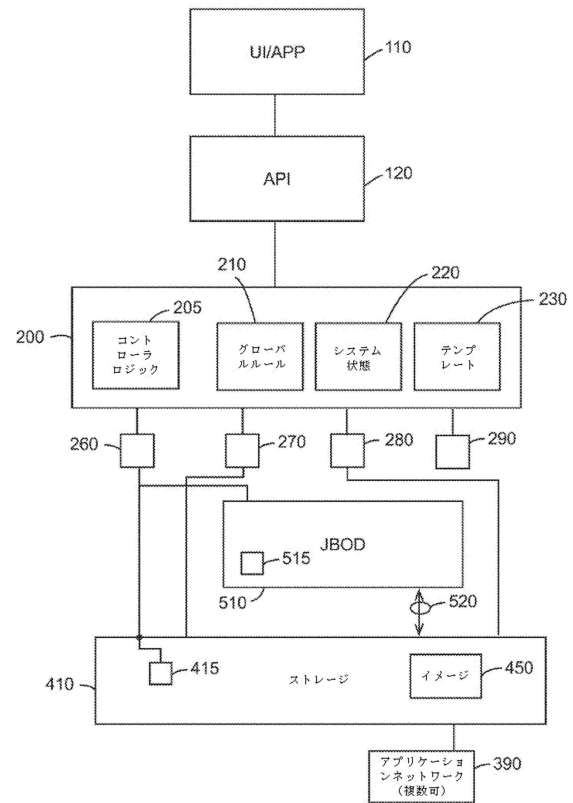
40

50

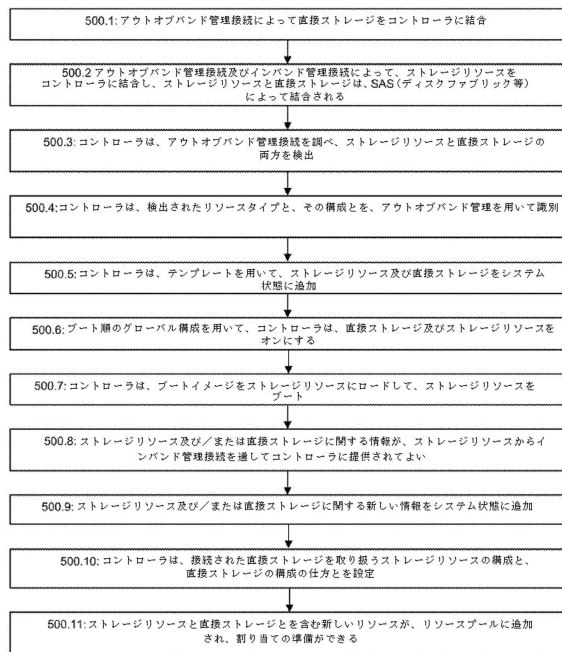
【図 4 B】



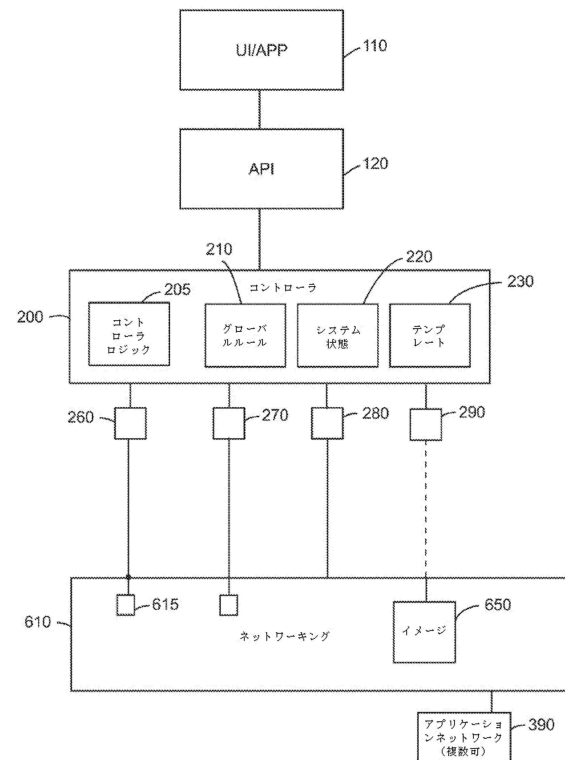
【図 5 A】



【図 5 B】



【図 6 A】



10

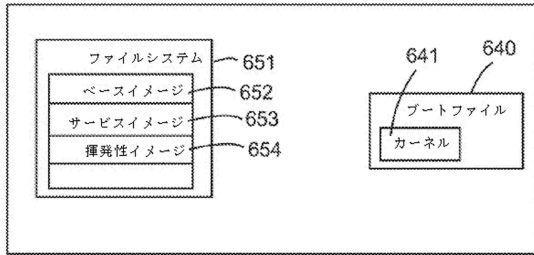
20

30

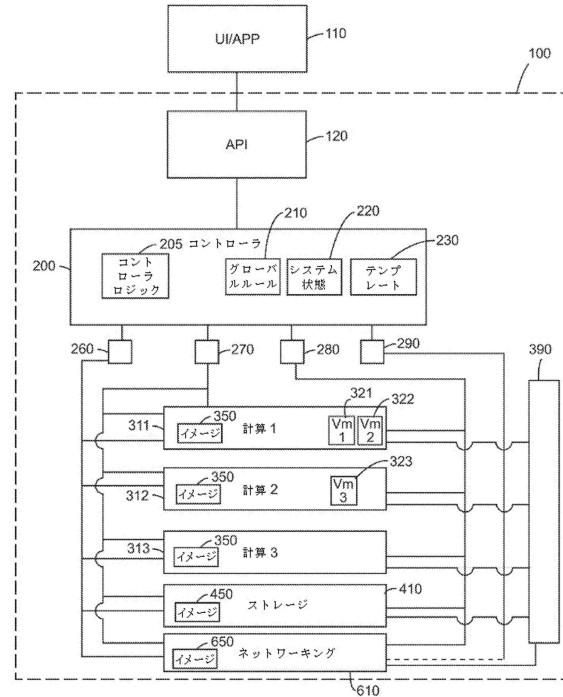
40

50

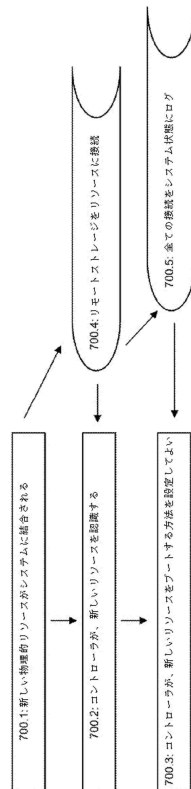
【図 6 B】



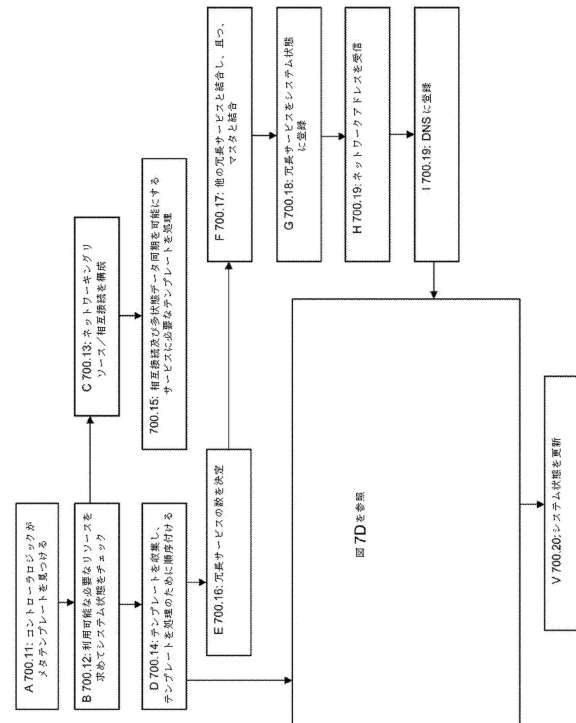
【図 7 A】



【図 7 B】



【図 7 C】



10

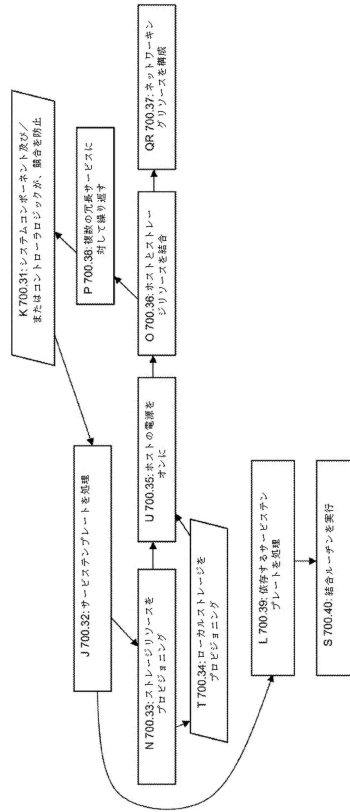
20

30

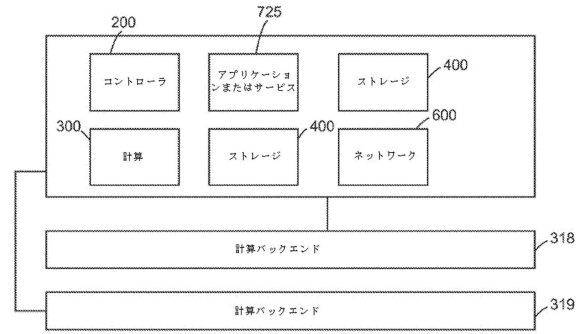
40

50

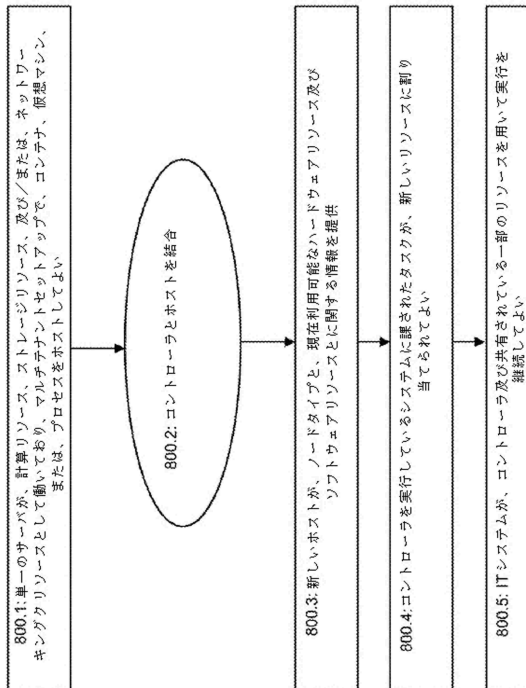
【図 7 D】



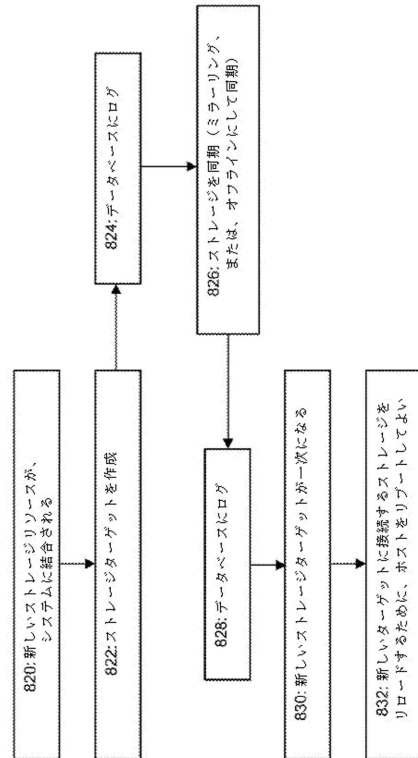
【図 8 A】



【図 8 B】



【図 8 C】



10

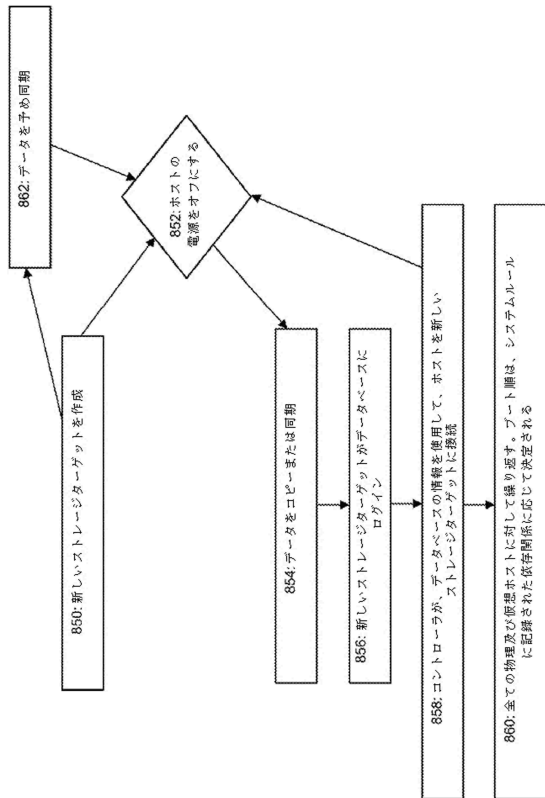
20

30

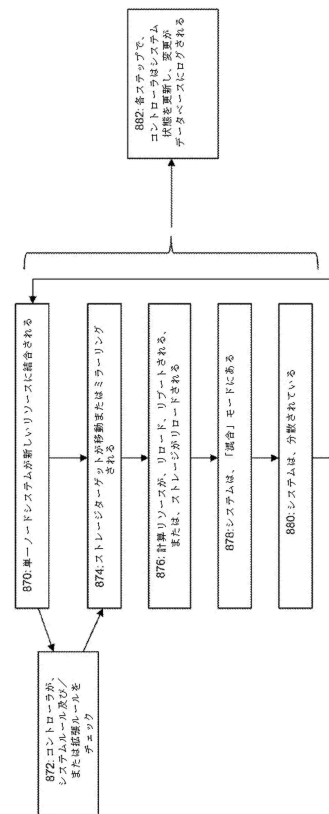
40

50

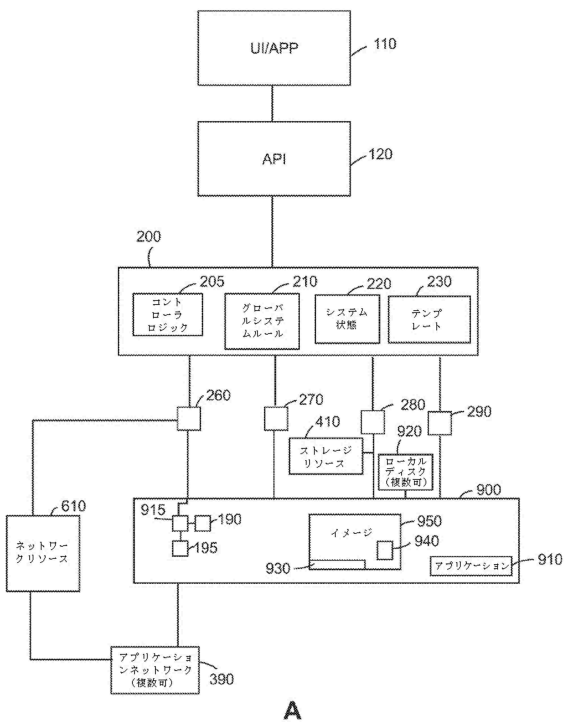
【図 8 D】



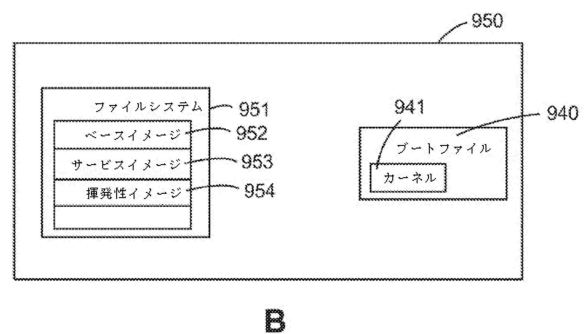
【図 8 E】



【図 9 A】



【図 9 B】



10

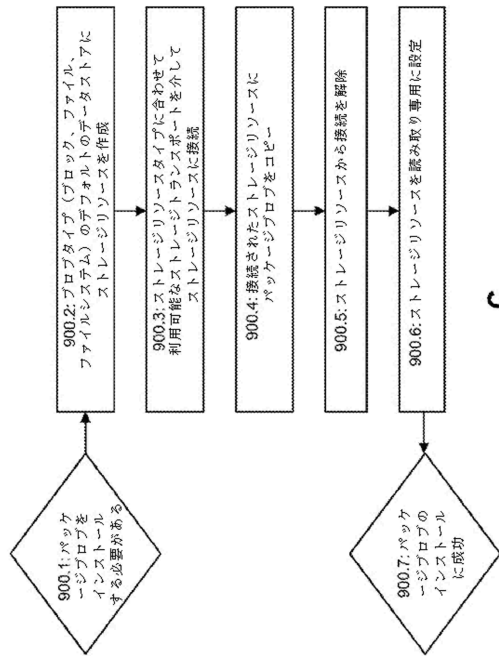
20

30

40

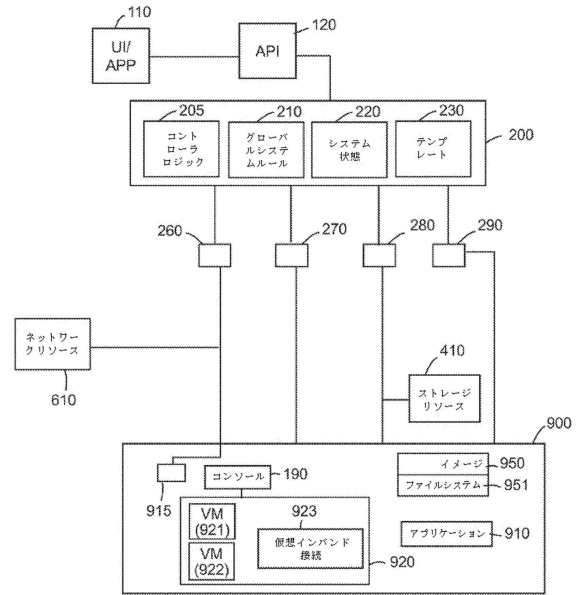
50

【図 9 C】



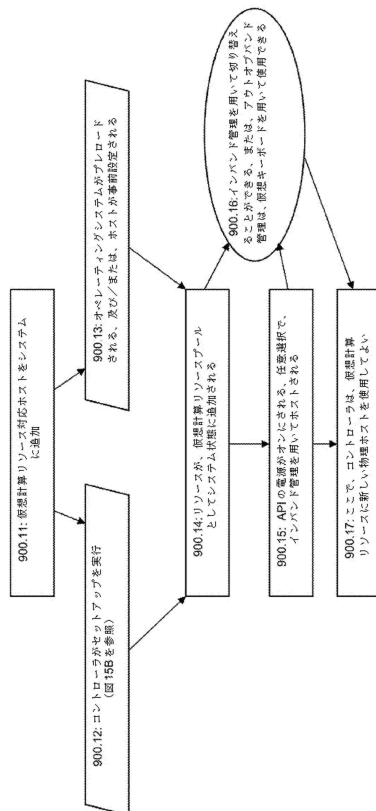
C

【図 9 D】



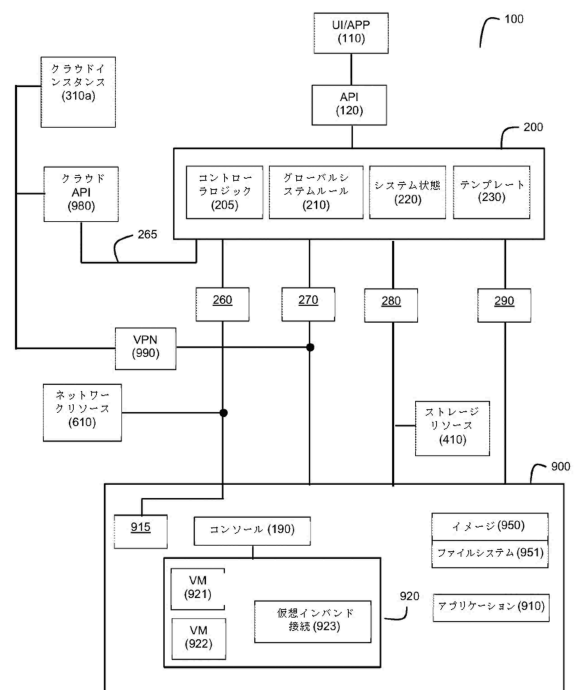
D

【図 9 E】



E

【図 9 F】



F

10

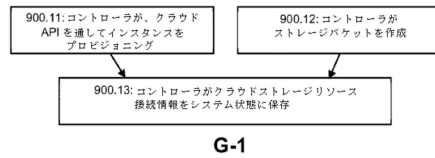
20

30

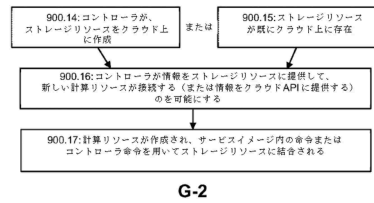
40

50

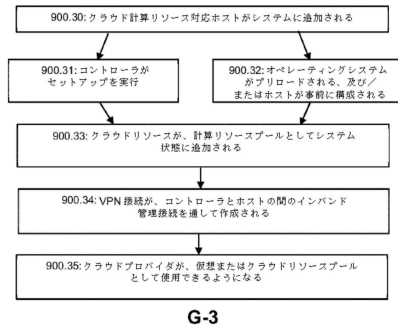
【図 9 G - 1】



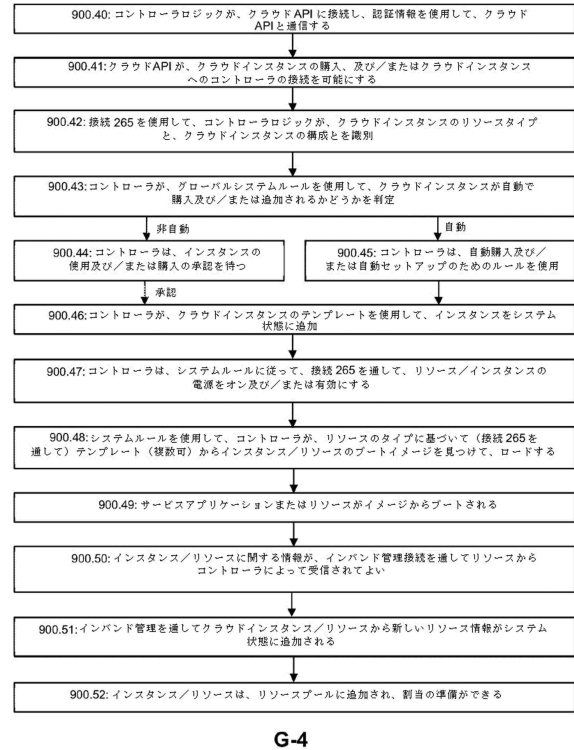
【図 9 G - 2】



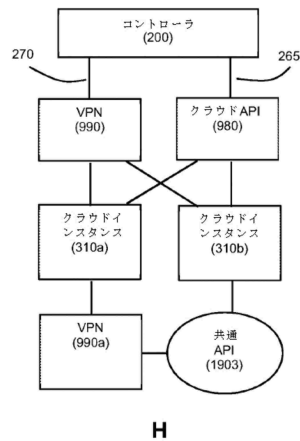
【図 9 G - 3】



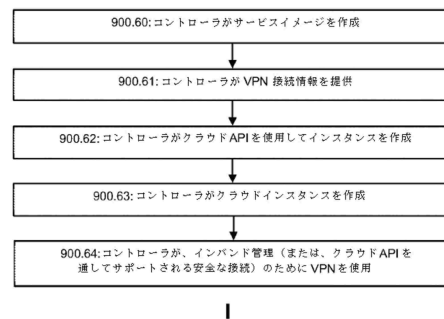
【図 9 G - 4】



【図 9 H】



【図 9 I】



10

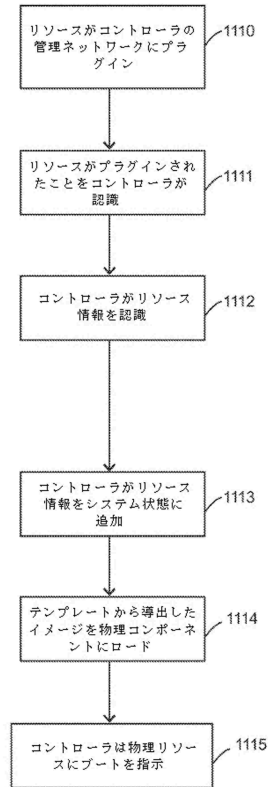
20

30

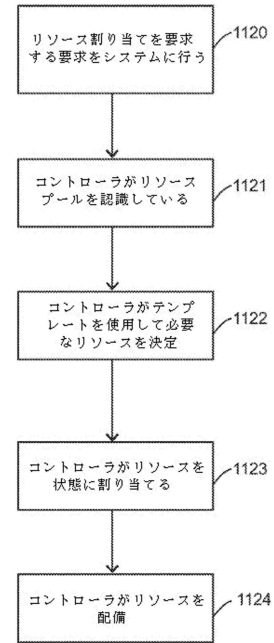
40

50

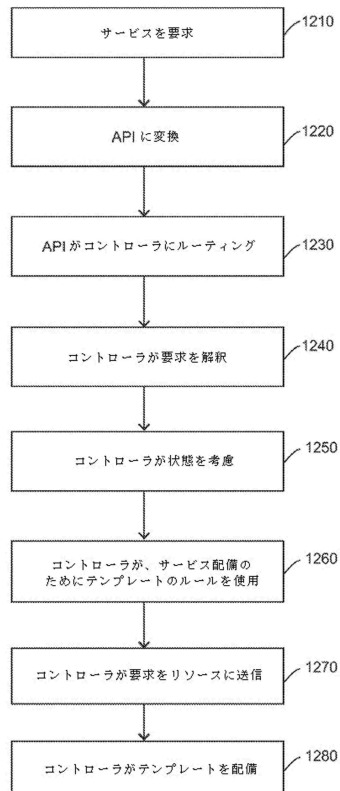
【図 1 1 A】



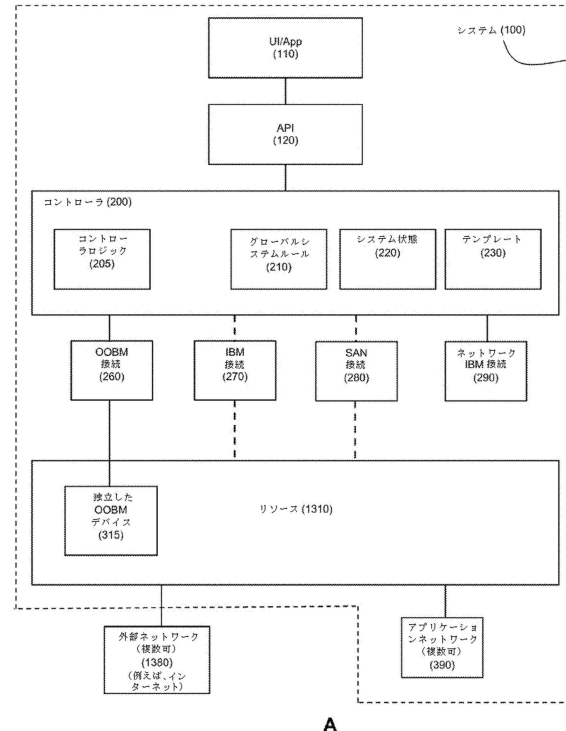
【図 1 1 B】



【図 1 2】



【図 1 3 A】



10

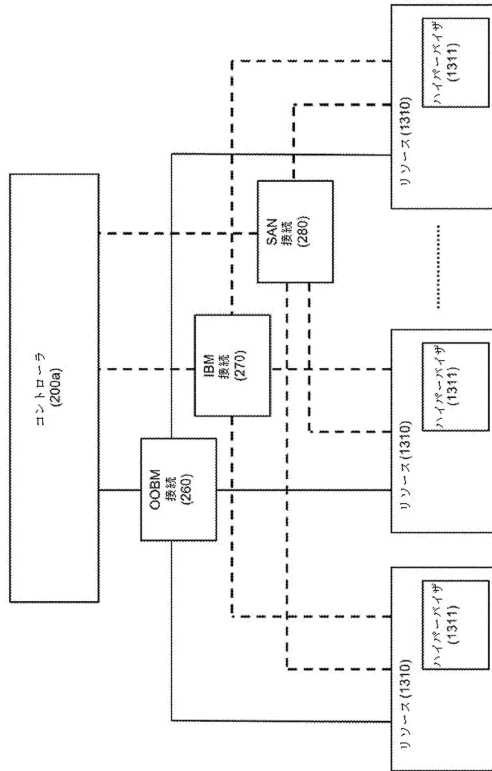
20

30

40

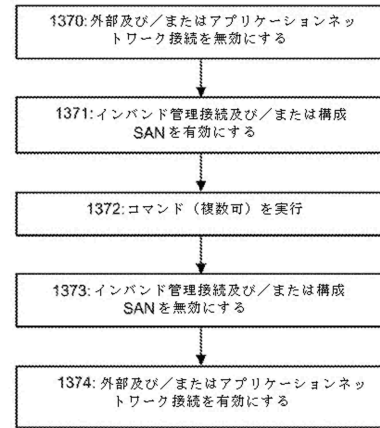
50

【図 13 B】



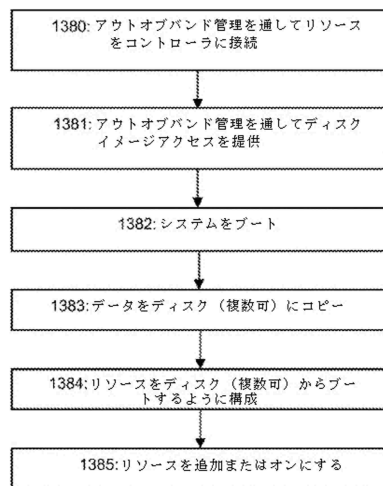
B

【図 13 C】



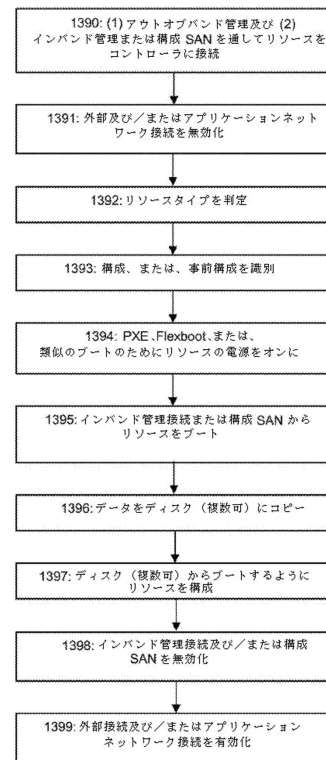
C

【図 13 D】



D

【図 13 E】



E

10

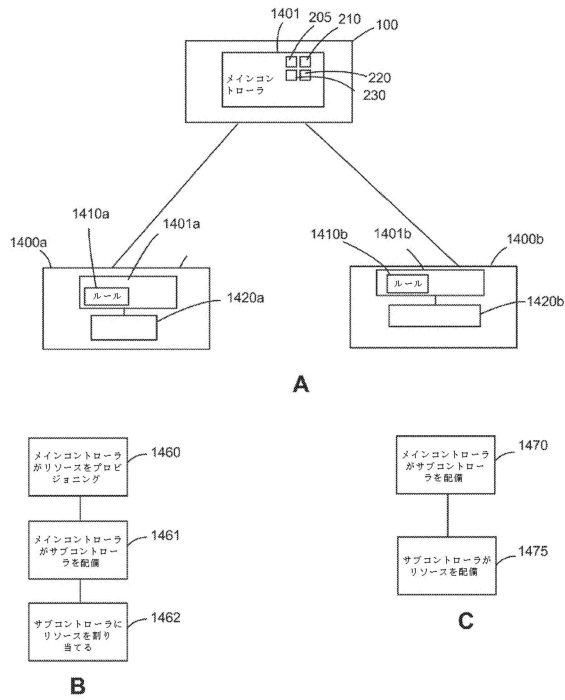
20

30

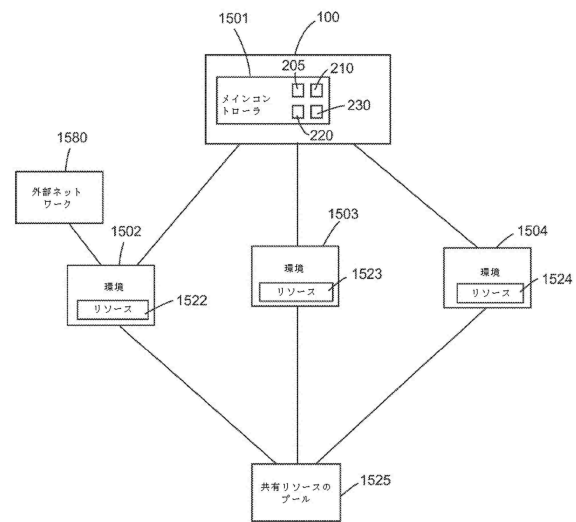
40

50

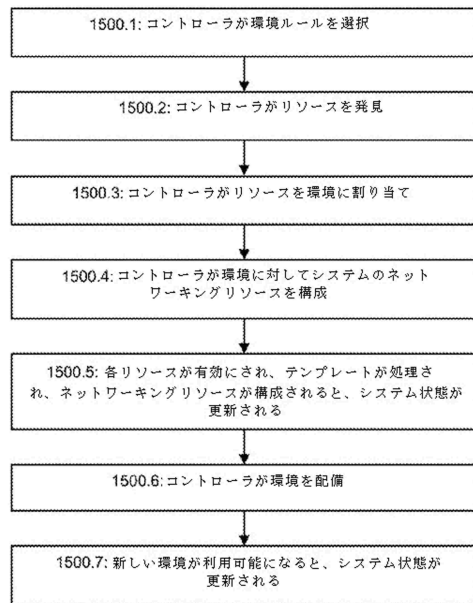
【図 14】



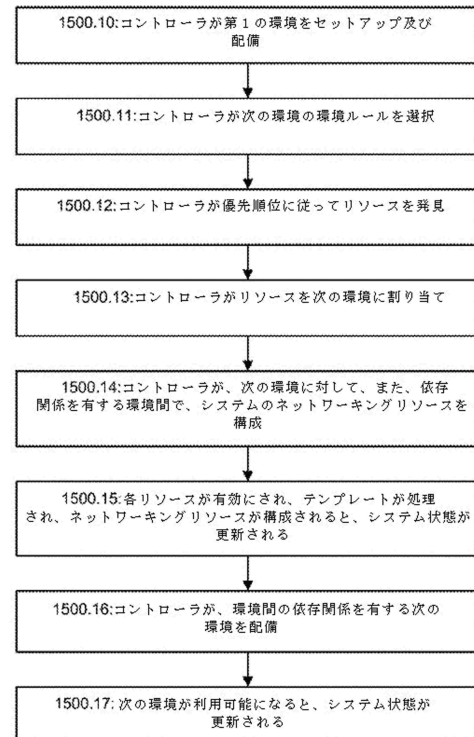
【図 15 A】



【図 15 B】



【図 15 C】



10

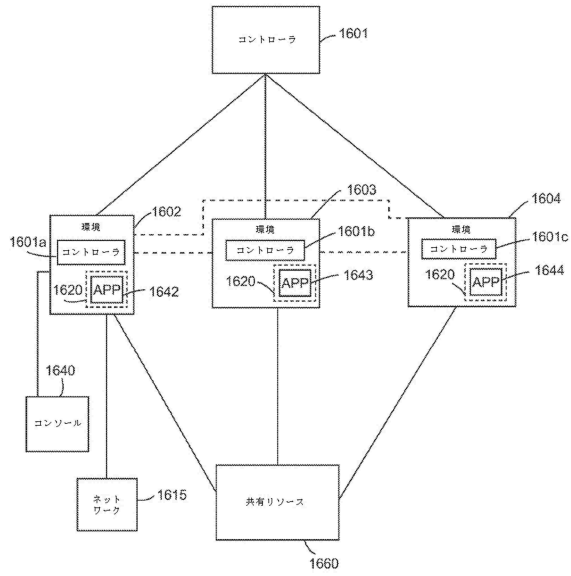
20

30

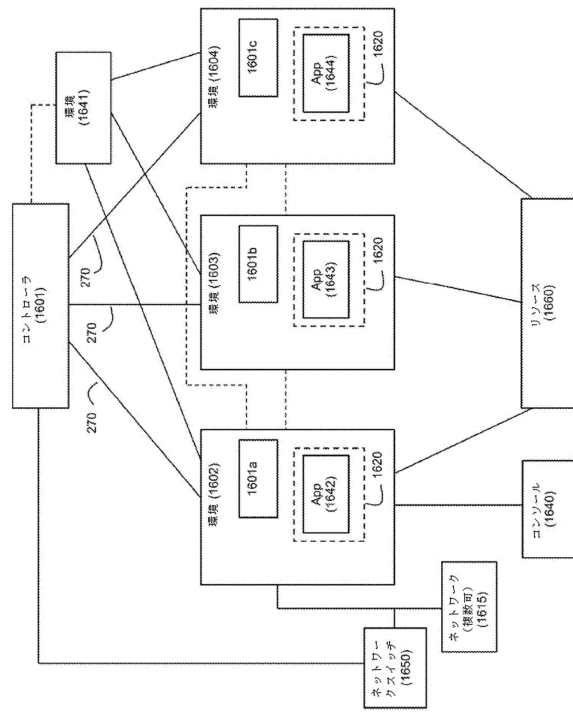
40

50

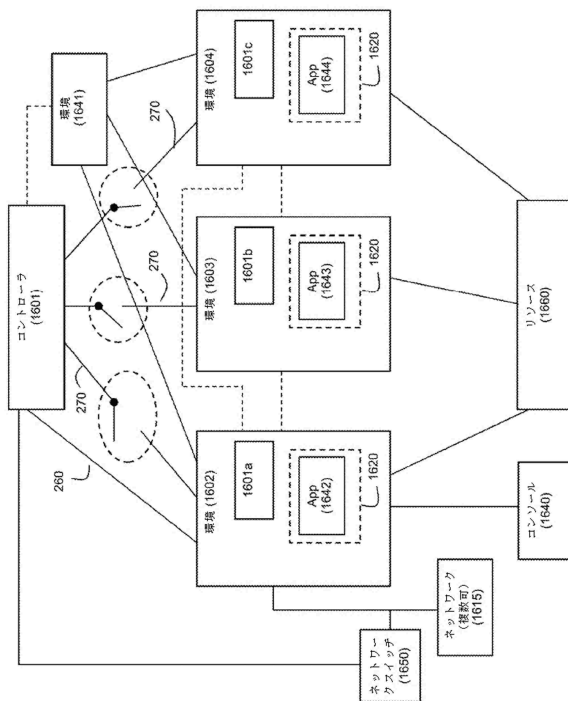
【図 16 A】



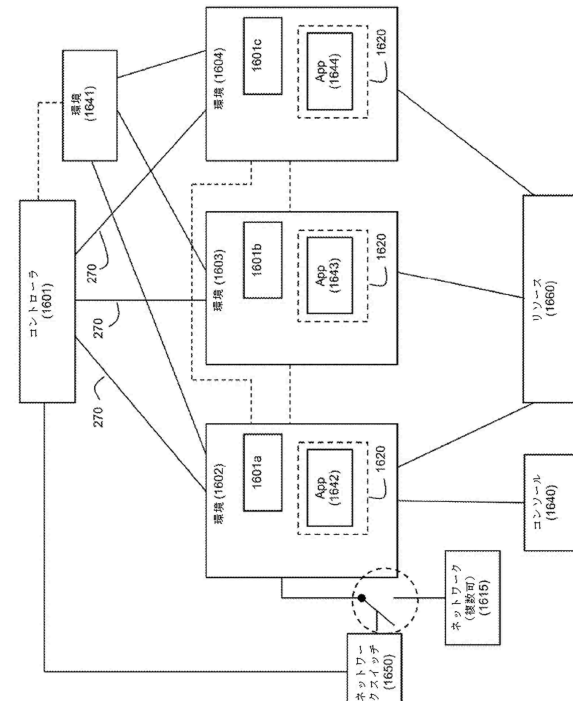
【図 16 B】



【図 16 C】



【図 16 D】



10

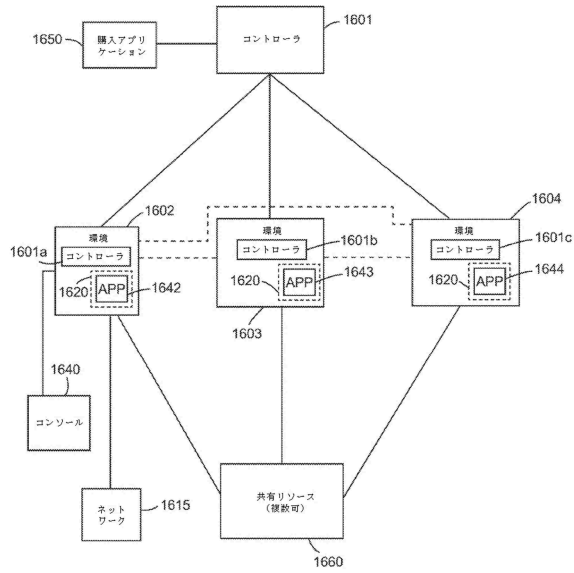
20

30

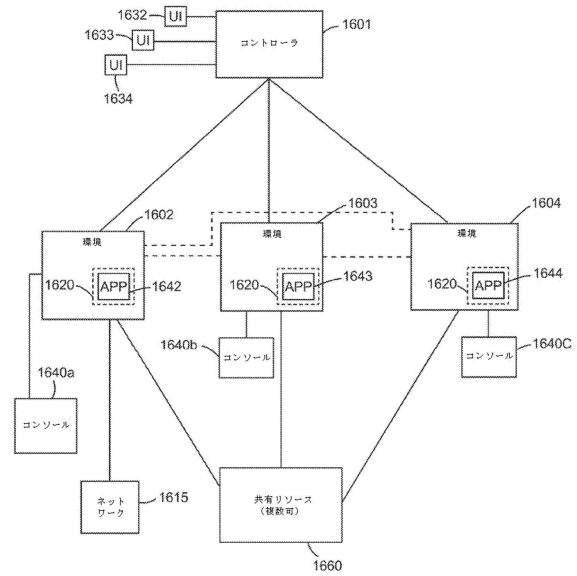
40

50

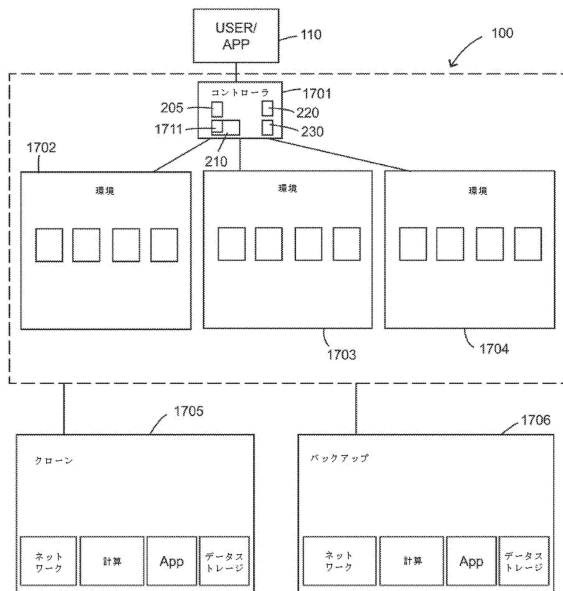
【図 16 E】



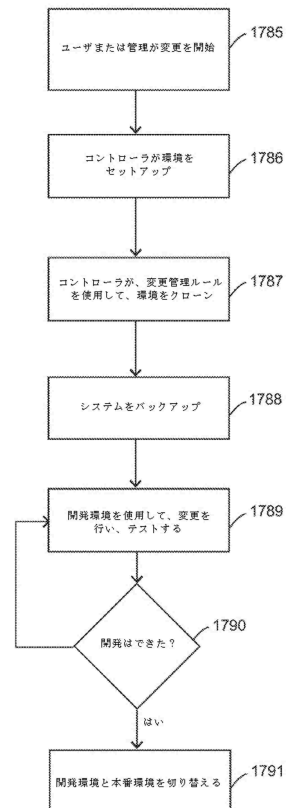
【図 16 F】



【図 17 A】



【図 17 B】



10

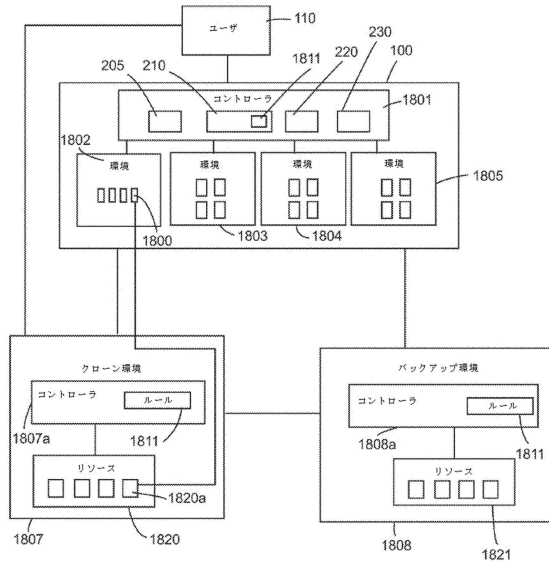
20

30

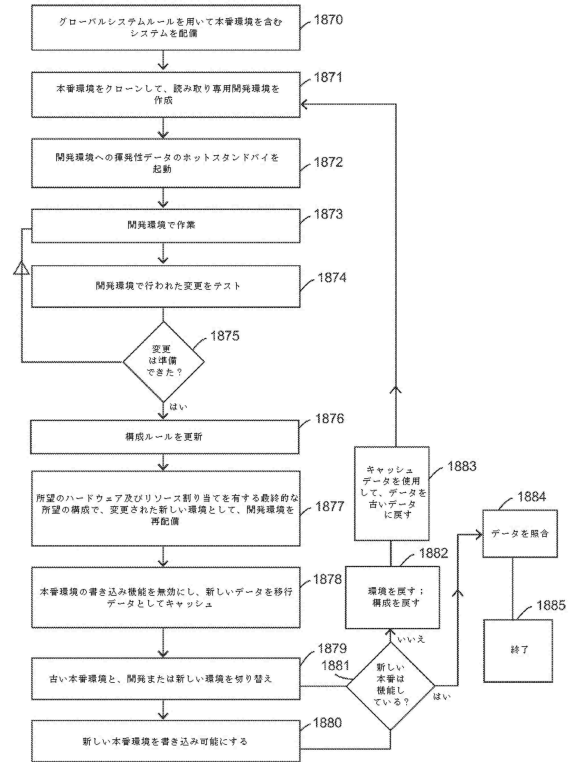
40

50

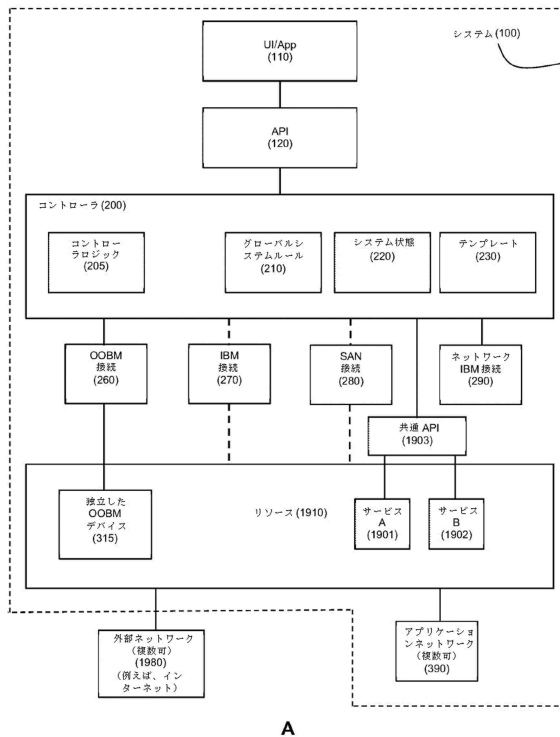
【図 18 A】



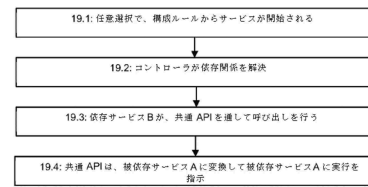
【図 18 B】



【図 19 A】

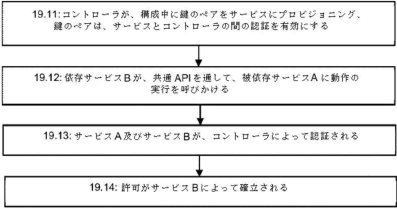


【図 19 B】



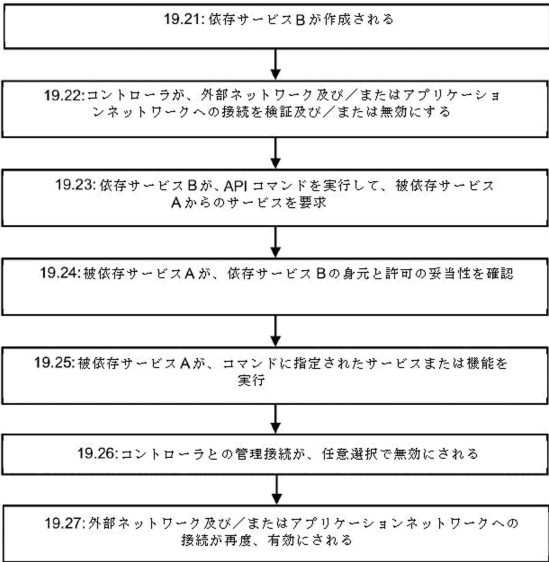
B

【図 1 9 C】



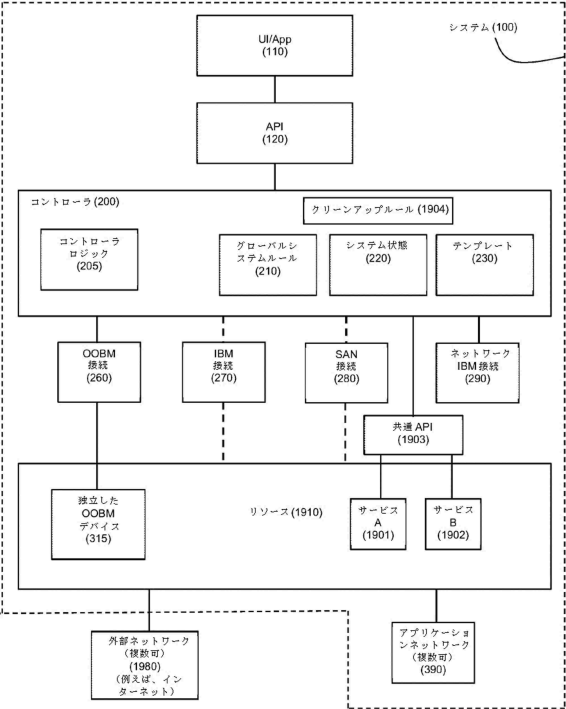
C

【図 1 9 D】



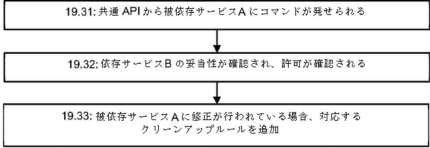
D

【図 1 9 E】



E

【図 1 9 F】



F

10

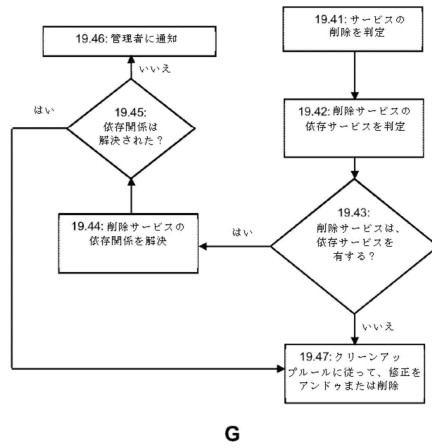
20

30

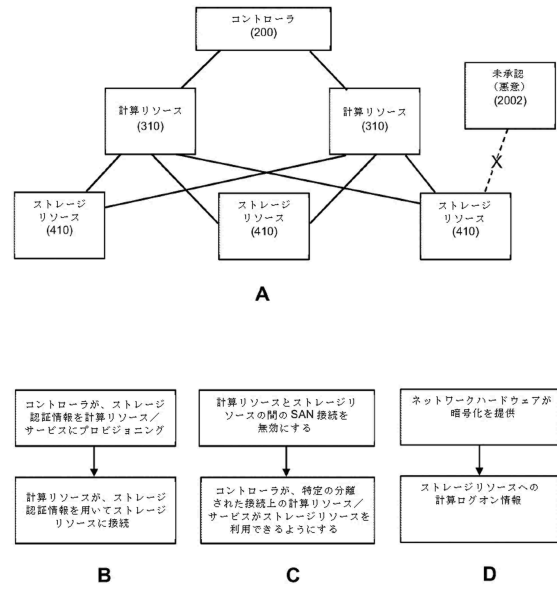
40

50

【図 19 G】

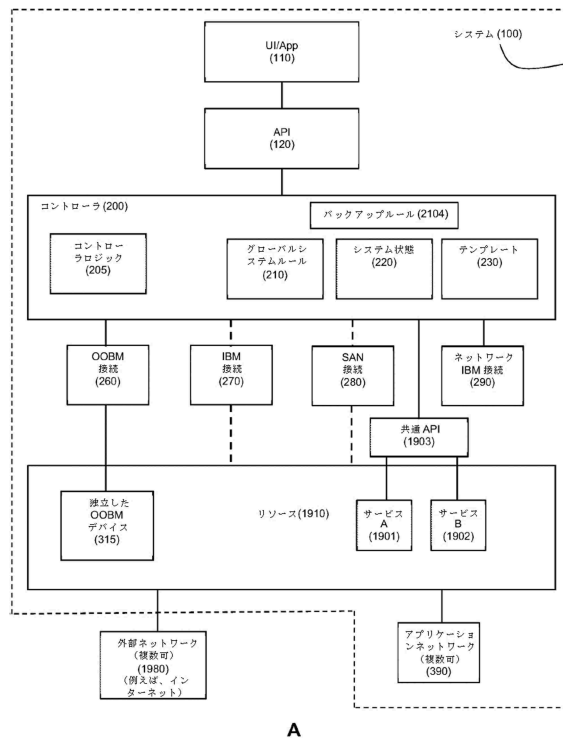


【図 20】

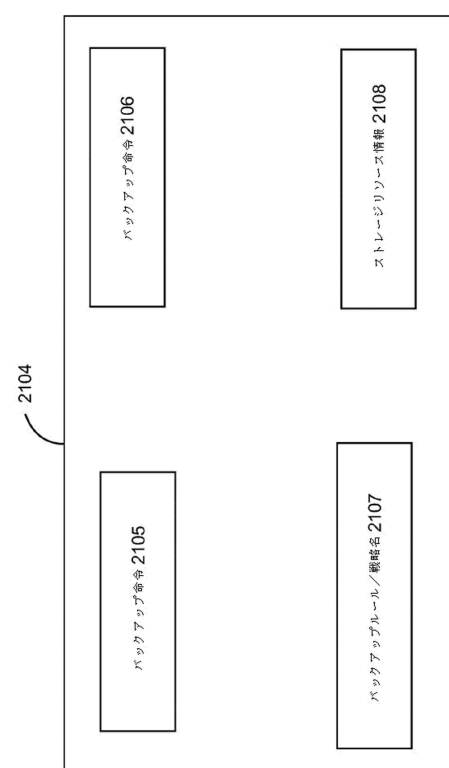


10

【図 21 A】



【図 21 B】



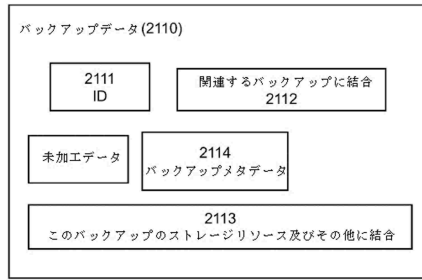
20

30

40

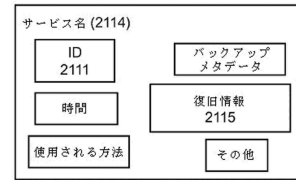
50

【図 2 1 C - 1】



C-1

【図 2 1 C - 2】



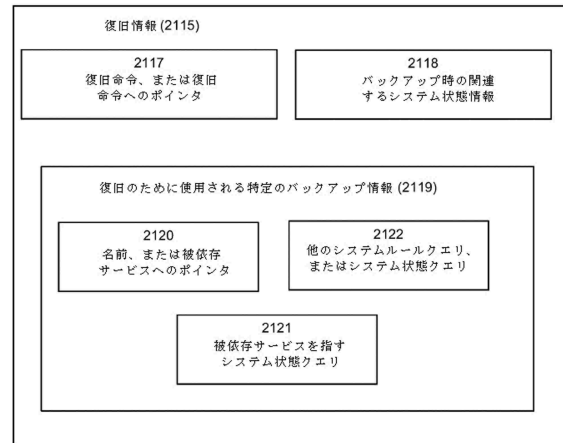
C-2

【図 2 1 C - 3】



C-3

【図 2 1 C - 4】



C-4

10

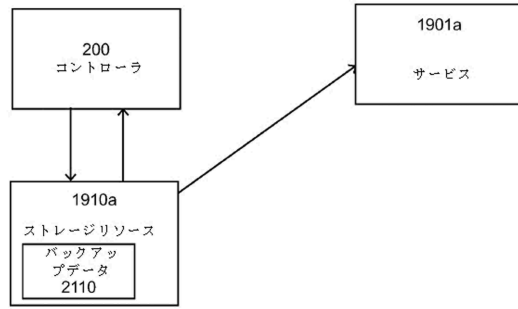
20

30

40

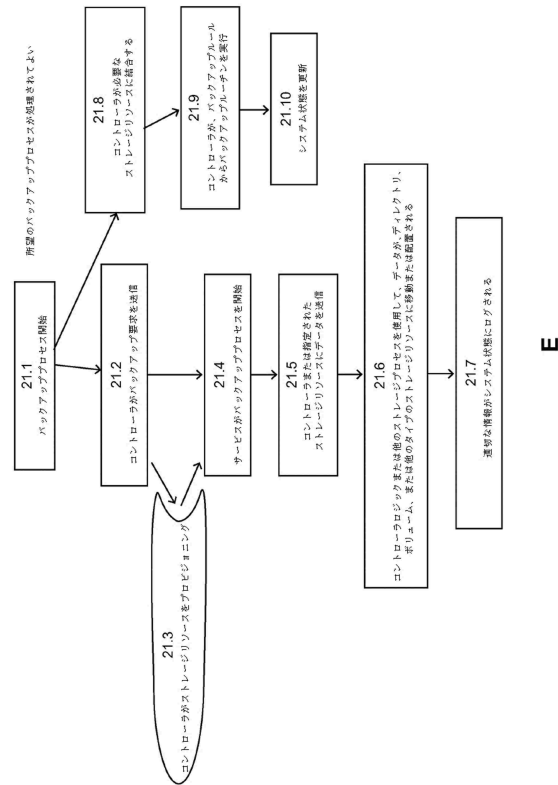
50

【図 2 1 D】



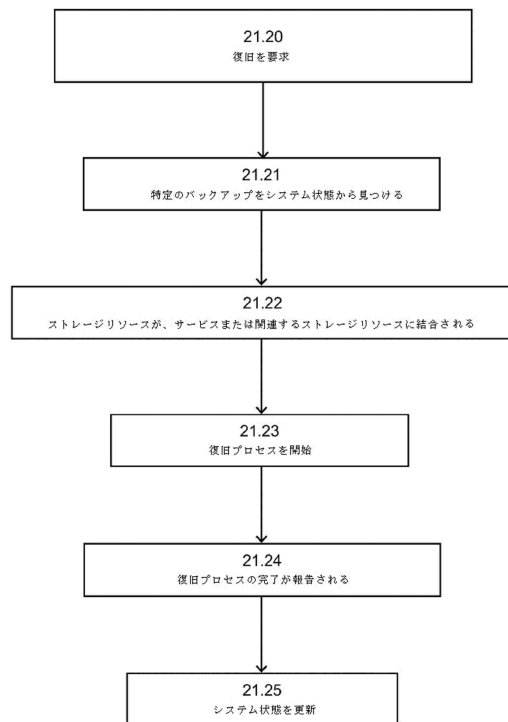
D

【図 2 1 E】



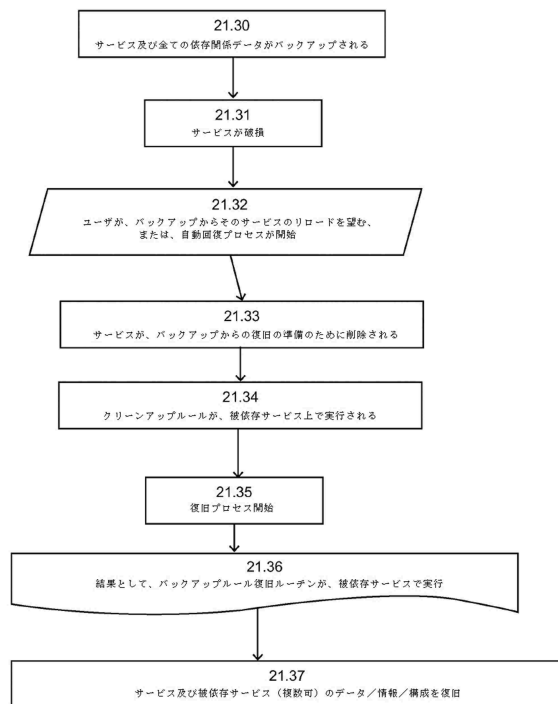
E

【図 2 1 F】



F

【図 2 1 G】



G

10

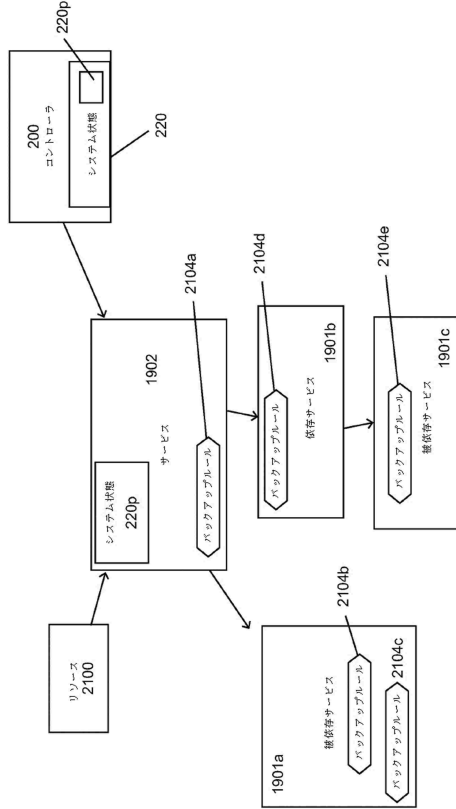
20

30

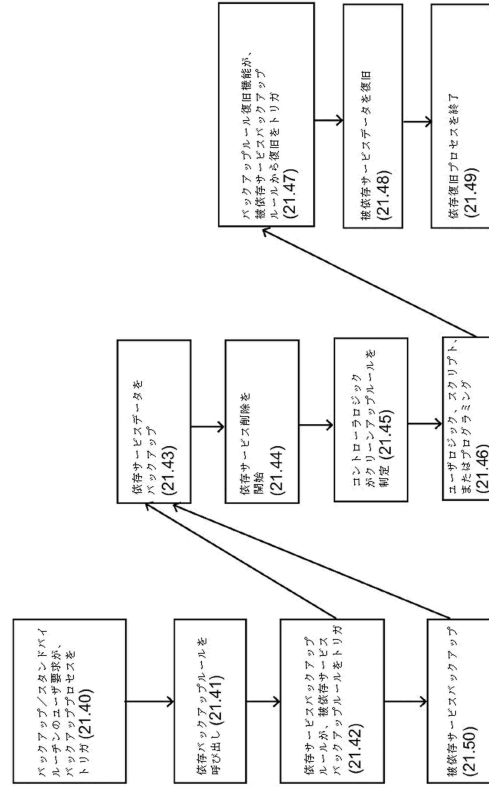
40

50

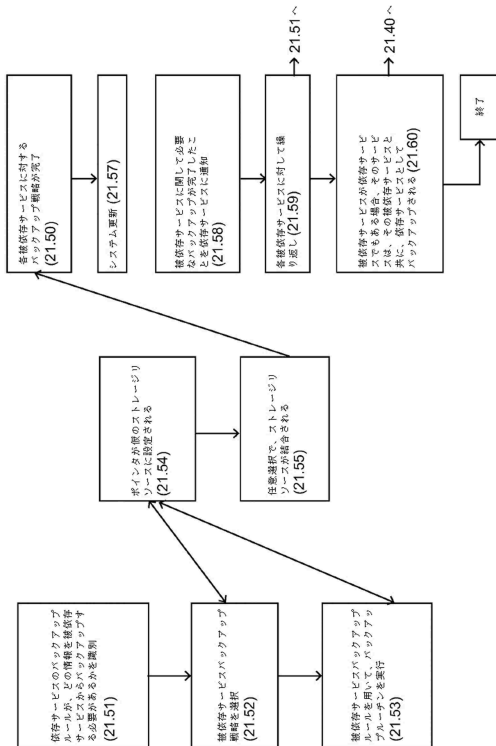
【図 2 1 H】



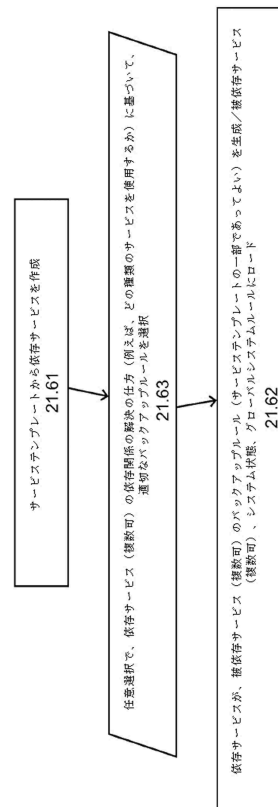
【図 2 1 I】



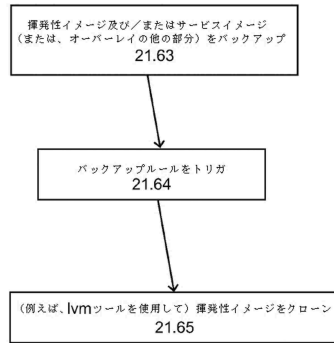
【図 2 1 J】



【図 2 1 K】

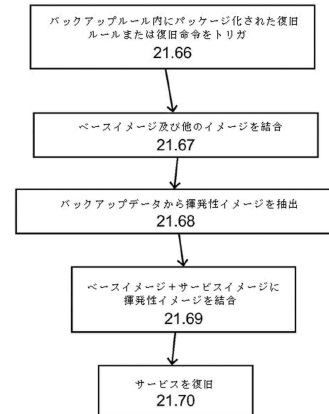


【図 2 1 L - 1】



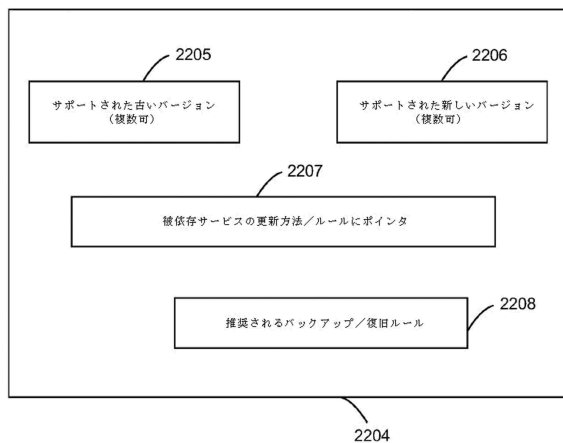
L-1

【図 2 1 L - 2】

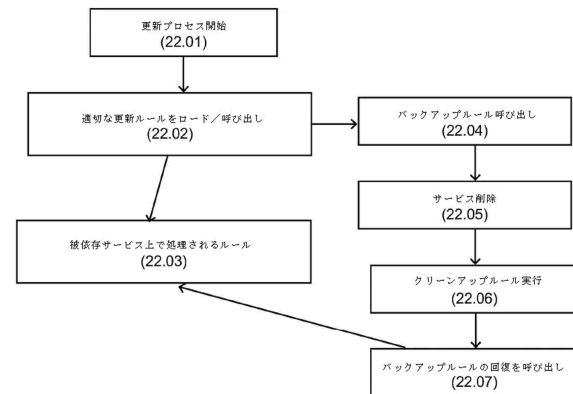


L-2

【図 2 2 A】



【図 2 2 B】



10

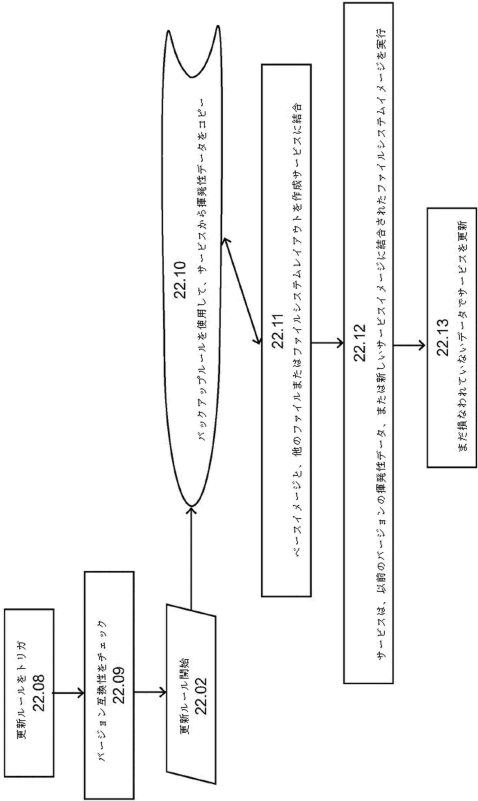
20

30

40

50

【図 22 C】



10

20

30

40

50

フロントページの続き

- (74)代理人 100116403
弁理士 前川 純一
- (74)代理人 100134315
弁理士 永島 秀郎
- (74)代理人 100162880
弁理士 上島 類
- (72)発明者 パーカー ジョン シュミット
アメリカ合衆国 6 0 0 9 0 イリノイ ウィーリング イースト マークワート ドライブ 1 1 1
- (72)発明者 ショーン マイケル リチャードソン
アメリカ合衆国 6 0 0 9 0 イリノイ ウィーリング イースト マークワート ドライブ 1 1 1
- (72)発明者 ニール ベンジャミン セメル
アメリカ合衆国 6 0 0 9 0 イリノイ ウィーリング イースト マークワート ドライブ 1 1 1
- (72)発明者 キャメロン タイラー スプライ
アメリカ合衆国 6 0 0 9 0 イリノイ ウィーリング イースト マークワート ドライブ 1 1 1
- (72)発明者 アリアナ オウサー
アメリカ合衆国 6 0 0 9 0 イリノイ ウィーリング イースト マークワート ドライブ 1 1 1
- 審査官 今川 悟
- (56)参考文献 特開 2 0 1 8 - 0 4 1 2 5 1 (J P , A)
米国特許出願公開第 2 0 1 9 / 0 0 1 8 7 6 0 (U S , A 1)
国際公開第 2 0 0 6 / 0 4 0 8 1 0 (W O , A 1)
- (58)調査した分野 (Int.Cl., D B 名)
G 0 6 F 8 / 6 0
G 0 6 F 2 1 / 4 4